OH, JIYOUNG, M.S. Stabilizing RED Queue Oscillation using the Logistic Map in AutoRED Mechanism. (2009)
Directed by Dr. Shan Suthaharan. 108 pp.

Active queue management (AQM) is one of the ways to control congestion at Internet Routers. One of the widely used AQM's is the random early detection (RED) scheme. The RED scheme suffers from chaotic queue oscillation problem particularly in a highly congested network. It causes jitter, high queuing delay when the queue size stays high, and underutilization when the queue size is low. Recently AutoRED algorithm has been proposed as a solution to the chaotic queue oscillation problem in that AutoRED calculates the weight, w_q, continuously as opposed to a constant value set by a user [1]. AutoRED displays the reduction of the chaotic queue oscillation by network performance metrics and queue behavior graphs, but there has been no metric known to measure the degree of queue oscillation in terms of its effect on the Quality of Service (QoS).

The purpose of the present study is twofold. Firstly, the possibility of an improvement by modifying AutoRED using a Logistic Map is investigated. This new technique introduces a user control parameter that can contribute to further improvements. Secondly, a new metric is proposed to show the degree of queue oscillation with regards to its effect on the QoS. The experiments are done by applying the new technique to network simulations in TCP only and TCP and UDP combined traffic environments. The results are compared with RED and AutoRED with regards to the proposed metric coupled with the network performance measurements and the statistical measurements of the queue behavior.

STABILIZING RED QUEUE OSCILLATION

USING THE LOGISTIC MAP IN

AUTORED MECHANISM

by

Jiyoung Oh

A Thesis Submitted to
the Faculty of The Graduate School at
The University of North Carolina at Greensboro
in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Greensboro
2009

Approved by

_____
Committee Chair

To my heavenly Father for His unfailing love

and

To my earthly parents for their patience and support

APPROVAL PAGE

This thesis has been approved by the following committee of the Faculty of The Graduate School at The University of North Carolina at Greensboro.

Committee Chair _____

Committee Members _____

_____

_____

Date of Acceptance by Committee

_____

Date of Final Oral Examination

ACKNOWLEDGMENTS

A great deal of appreciation goes to Dr. Shan Suthaharan for his continual support and guidance. I am also thankful to Rev. Yoochan Choi and the congregation of the Korean First Presbyterian Church for their financial support during this semester. My appreciation also goes to Jungho Pak for proofreading and Gloria Fang for supporting it financially.

Last but not least, I am grateful for the support and encouragement of John C. Chang and his family, and Jonghee Shim.

TABLE OF CONTENTS

Page

CHAPTER I

INTRODUCTION

Overview

The past two decades have seen the Internet growing from a network of a group of researchers to a worldwide communication channel and information reservoir. As more technologies are developed, many new facets have been added to the textual, wired-based communication. On top of that, crossover between data networks and traditional PSTN channels and cellular networks are increasingly more available essentially creating an integrated communication environment.

High quality of service coupled with minimal delay has been the expectations of Internet users. The realization of such expectations has been made more difficult by the exponential growth of the number of users as the applications and services offered through the Internet become more versatile. Given that the existing resources in the network infrastructure would not be able to meet all of the needs of service providers and end users, the network channels are often congested. This happens particularly at what is known as a bottleneck which is a network link whose bandwidth is not sufficient enough to meet the aggregate demand of the network links that are directly connected with. Intermediate devices known as routers or gateways interfaced with such bottleneck where

traffics from end users and from other gateways pass through are most likely to experience high levels of congestion.

The control of such congested traffic has become one of the main research topics in the field of computer networks. The goal is to maximize the throughput while minimize the delay and loss at the same time. Researchers have approached solving this problem from largely two angles. One approach looks into controlling data sending rate at the end user level whereas the other considers the congestion control mechanism at the gateway level.

The first approach incorporates a mechanism that reacts to congestion level by controlling data sending rate at the end user level. Simply put, when resources are available end user's data gets transmitted at a higher rate and when the network congestion is sensed the end user data transmission goes into an abeyance and slowly resumes as it probes the congestion level of the network. This congestion avoidance mechanism was first introduced by Nagle [2] triggered by the "Internet meltdown" which occurred in the mid 80's during the early growth of the Internet. Transmission Control Protocol (TCP) employs this mechanism in its variants that increases and reduces the transmission rates of end user data based on the congestion level. Such variations include Tahoe [3], Vegas [4], and Reno [5]. TCP schemes "sense" the congestion level by absence and delay of acknowledgement packets from recipient based on the time-out determined either by a fixed timer or an adaptive retransmission timer [6]. This works as a feedback from the network. TCP then applies its response to a window size which works as a leash to the packet transmission rate. TCP Reno, for example, reduces its

window size to half when congestions is sensed and then increase it one-by-one in recovery as the mechanism receives acknowledgement packets.

While this governs the behavior of the transmission at the end user level, the Internet gateways or routers also have to be controlled because packets do not travel from one end to the other on a single hop but each packet may travel different routes consisted of various bandwidths and network conditions and a number of gateways that interconnects such diverse network routes. As a packet is received at a router, the router reads the packet, determines an appropriate port based on the destination address, and then sends the packet on the port onto the next hop. To improve router performance and control the congested traffics, mainly two approaches have been made.

One approach, known as "scheduling", deals with the order of processing packets by their types or priorities. This provision is to improve the QoS based on traffic types and priorities which gateways employing only the best effort mechanism cannot adhere to. In this mechanism, packets have indications for their types and priorities recognized by the scheduling policy which in turn allocates bandwidth accordingly.

The other group of researchers have worked on mechanisms categorized as "queue management" algorithms [7]. Routers are equipped with a queue or buffer that holds incoming packets while processing one packet at a time. Queue management algorithms determine when and how to drop packets. Dropped packets become lost and the end user will sense the network congestion by the absence of acknowledgement packets for the lost packets. The most widely deployed mechanism of managing the queue is known as "Drop Tail" mechanism. When packets arrive faster than a router can

process them, the router queue becomes full and the packets arrived while the queue is full are dropped.

Drop Tail is easy to understand and simple to implement. However it has exhibited serious disadvantages. Reference [7] notes them as the "lock-out" phenomenon and "full-queues" problem. Certain network situations such as traffic bursts often activate a "drop tail" queue to drop correlated packets from a segment of network flows which then responds by slowing down the transmission rate. Then the network flows unaffected by this monopolize router resources. Often the case is the group of flows that have slowed down suffers from synchronization while the unaffected group gets to use queue space exclusively. This is called a lock-out phenomenon created by timing effects of TCP congestion avoidance scheme.

The full-queues problem stems from the fact that in a congestive network drop-tail queue is unable to "inform" end nodes of the congestion until it becomes full and subsequently drops packets. Considering the Internet witnesses burst traffics, ideally, a queue should maintain room to absorb short bursts of traffics for overall high throughput and low end-to-end delay [7]. Other similar mechanisms such as "drop front on full" or "random drop on full" may well alleviate the lock-out phenomenon but the full-queues problem still remains unresolved.

As a response to these issues, a group of new mechanisms have been introduced. These mechanisms are called "Active Queue Management (AQM)" techniques. Random Early Detection (RED) [8], Virtual Queue (VQ) [9], Random Early Marking (REM) [10], and Adaptive Virtual Queue (AVQ) [11] are examples.

The main goal of AQM is to keep the network traffic volume under control in such a way that a router queue does not overflow while maintaining a steady flow of throughput. To achieve this, AQM mechanisms detect the incipient congestions and signal responsive network sources to reduce their transmission rates. As a part of this, AQM mechanisms maintain the current size of a queue at a desired level below its full capacity in a steady state. The space between the desired level and the maximum capacity are to be occupied when congestion or a bursty traffic occurs.

The question is then what should be an appropriate queue level in a steady state and a drop probability that maintains the queue level and allows it to increase when needed?

Employing the algorithms addressing these questions, RED has been recommended by ITEF as a congestion control policy at gateways for the Internet [7]. Conceptually, congestion control at gateways that can adapt to dynamically changing network traffics is a logical development. However, designing algorithms that can actually perform well in real network environment has been a very difficult task. Although the algorithms in RED are relatively simple and easy to understand, its implementation has not been as successful as anticipated due to the sensitivity of its control parameter values resulting in unexpected outcomes. General consensus has been that the interaction between RED gateways and TCP/IP mechanisms is not well understood. For this reason, reference [12] reported more research on the understanding of the dynamics of RED and TCP/IP should precede the deployment of RED. Since then,

many researches have been conducted on understanding RED theoretically and empirically.

As a result, a group of RED variants have been suggested as well, notably FRED [13], stabilized RED (SRED) [14], Blue [15], adaptive RED (ARED) [16], and Exponential-RED [17] and AutoRED [1].

Particularly, AutoRED incorporates an additional algorithm into RED that allows one of the control parameters in RED to be calculated dynamically based on the current network characteristics. This addresses the observation made by [12] that a static RED does not produce better result than that of Drop-tail in dynamic and heterogeneous network environments.

Research Problem

One of the unexpected behaviors exhibited by RED is non-deterministic chaotic queue oscillation. This along with the sensitive nature of its control parameters has been reported [12], [18]-[20]. In addition to the fact that the chaotic queue oscillation contributes to low throughput and high packet loss, it is problematic largely on two accounts. Firstly, a large queue delay for one cluster of packets and almost no delay on next increases a chance of jitter at a receiving end. Secondly, in a valley of oscillation where the current queue size is below the average queue size, gateway resources are underutilized while in a peak period packets are dropped due to the overflow. To address this issue, AutoRED has been proposed such that it reduces the chaotic queue oscillation by continuously calculating the weight parameter based on network characteristics [1]. Simulation results and illustrations reported in [1] exhibit visible improvements in the

behavior of the current queue size over time of an AutoRED scheme than that of RED. However, in a highly congested network where AutoRED appears to reduce the chaotic queue oscillation the most, it also shows high packet loss rate and a higher overall average queue size resulting in larger queuing delays. This finding has prompted the further study on AutoRED for possible improvements.

As noted earlier, chaotic queue oscillation is problematic because of low performance, jitter, and under-utilization of queue resources. By plotting current queue size over time, one can visibly identify a region of chaotic queue oscillation. However, other than the graphical representation, there has been no definitive metric that measures the degree of queue oscillation with regards to its effect on the Quality of Service (QoS). While the instantaneous queueing delay is indirectly measured by the queue size itself at the moment, the queueing delay alone cannot provide the information regarding how long the queue size stays above the overall average queue size or below the average. This information is important particularly for the QoS in that end users would be less satisfied due to high delays as the queue size stays above the average. Similarly, network service providers would be less satisfied due to the under-utilization of the resources if the queue size stays below the average. In both cases, the longer the queue size stays as it has, the less satisfied end users and service providers would be.

In this respect, chaotic queue oscillation has poor QoS because it typically exhibits longer periods of queue size staying above or below the average than non-chaotic queue oscillation does. Therefore, an indicator of such queue behavior should exist that represents its effect on the QoS.

Structure of Thesis

The composition of the remainder of this thesis is as following. In the second chapter RED mechanism is presented with the original RED and Auto RED described in more detail. The third chapter contains the fundamentals of the logistic map and how it is applied to the proposed mechanism called L_{map}-RED. The fourth chapter presents the definition of the new metric. In the fifth chapter, the experimental results comparing RED, AutoRED, and L_{map}-RED in terms of the new metric are given. The methodology of the simulation experiments is also included in this chapter followed by discussions in the sixth chapter. Lastly the seventh chapter concludes with the suggestion for future research.

CHAPTER II

PRELIMINARIES

RED Algorithm

AQM mechanisms strive to maintain the average queue size in a router at a certain level while leaving a room to absorb bursty network traffics. Authors of RED list four design goals in [8]. The first and main goal is to control the average queue size in order to provide congestion avoidance instead of reaction to congestion. By keeping the average queue size at around a region of high throughput and low delay, the queuing delay is lowered and therefore the end to end delay is reduced. Additionally, by reducing the packet transmission rates preemptively through congestion avoidance mechanism of TCP, the router queue should be able to process incipient bursts in network traffics and is unlikely to overflow. Then, transmitted packets are hardly dropped, hence, minimizing the packet loss rate. Second in the list is global synchronization avoidance and the third is avoidance of a bias against bursty traffic. Global synchronization occurs when congestion is noticed globally and all flows retreat from sending packets at the same time. This can be avoided at gateways by choosing appropriate flows and notify to back off at the onset of congestion. This way, only a portion of connections responds with congestion avoidance mechanisms while other flows transmit in a normal fashion. In addition, unlike

Drop Tail in which bursty traffics are unfavorably treated, RED can decide which flows to notify of congestion in a way that it avoids a bias against bursty flows.

In fact, RED employs a randomized algorithm in order not to penalize particular sources. Lastly, RED should be capable of maintaining an upper bound on the average queue size even without network sources with congestion avoidance mechanisms. This can be done by actually dropping incoming packets when the average queue size exceeds the imposed maximum threshold. It should be noted that original RED "marks" incoming packets instead of dropping to notify of the congestion if a gateway queue is not full.

With these goals, RED employs a relatively simple mechanism composed of two main parts with a set of control parameters. The first part is calculating an average queue size and the other is deciding whether to discard an incoming packet or not. Based on the estimation of the average queue size, the algorithm determines how to handle an incoming packet.

Upon arrival of each packet in the queue, an average queue size is calculated by a method known as exponentially weighted moving average (EWMA) as in (1) using the notation presented in [1].

$$q_{avg,t} = (1 - w_q) \times q_{avg,t-1} + w_q q_t \qquad (1)$$

where w_q is the weight parameter chosen by a user. The average queue size is used instead of the actual queue size in order to filter out incipient congestion. It should be noted here that the weight, w_q, determines how much portion of the actual queue size at time t, q_t, to be accounted in calculating the average queue size, $q_{avg,t}$. When the weight is small, the average queue size sustains its previous level without closely following the

movement of the instantaneous queue size. This is a desirable setting so that the average queue size does not rapidly react to transient congestion.

After the average queue size is calculated, if the average queue size is under the minimum threshold, the packet is queued. No incoming packet is dropped or marked if the average queue size is lower than the minimum threshold. Likewise, if any packet with the average queue size on or over the maximum threshold is dropped or marked. Now the incoming packets with the average queue size being on or over the minimum and under the maximum threshold are determined to be queued or dropped/marked based on the maximum probability. The probability of being dropped/marked increases as the average queue size becomes closer to the maximum threshold. This is the second component which is further described in the next paragraph. All of these parameters, w_q, minimum and maximum thresholds, and the maximum probability are chosen by a user. Authors of RED mechanism in [8] provide three guidelines as to choosing parameters. First, $w_q \geq 0.001$ and their choice of value is 0.002. Second, the minimum and the maximum thresholds to be set high enough to maximize the network power, and third, the maximum threshold to be at least as twice as the minimum threshold.

The second part of deciding packet discard is essentially determining how frequently incoming packets are marked or dropped. When the average queue size is in a critical region which is between the minimum threshold and the maximum threshold, the original RED algorithm applies a packet-marking probability increased linearly as the average gets closer to the maximum threshold. The packet-marking probability is also increased when the count of consecutively queued packets is increased. This way, a RED

queue does not wait too long before marking a packet and packets are marked at evenly spread out intervals. It is to ensure the fair and random treatment of various sources to avoid biases and global synchronization. Mathematical representation is presented in three steps. First, the fraction, *Fr*, of the region less than the average queue size is as following:

$$F_r = \frac{q_{avg,t} - \min_{th}}{\max_{th} - \min_{th}} \qquad (2)$$

Then, a packet-marking probability, P_b, is computed by

$$P_b = P_{max} \times Fr \quad \text{where } 0 \le Fr \le 1 \qquad (3)$$

(note that $Fr = 1$ when $q_{avg,t} = max_{th}$)

On top of this, in order to incorporate the count factor, the final packet-marking probability, Pa, is computed by

$$P_a = \frac{P_b}{(1 - count \times P_b)} \qquad (4)$$

Although RED mechanism seems superior to that of Drop Tail theoretically and experimentally, its parametric sensitivity makes it difficult for wide deployment. One of the abnormal behaviors of a RED queue is its chaotic queue oscillation illustrated in Fig. 1 resulting in unexpected network delays, jitter, and underutilization of router resources.

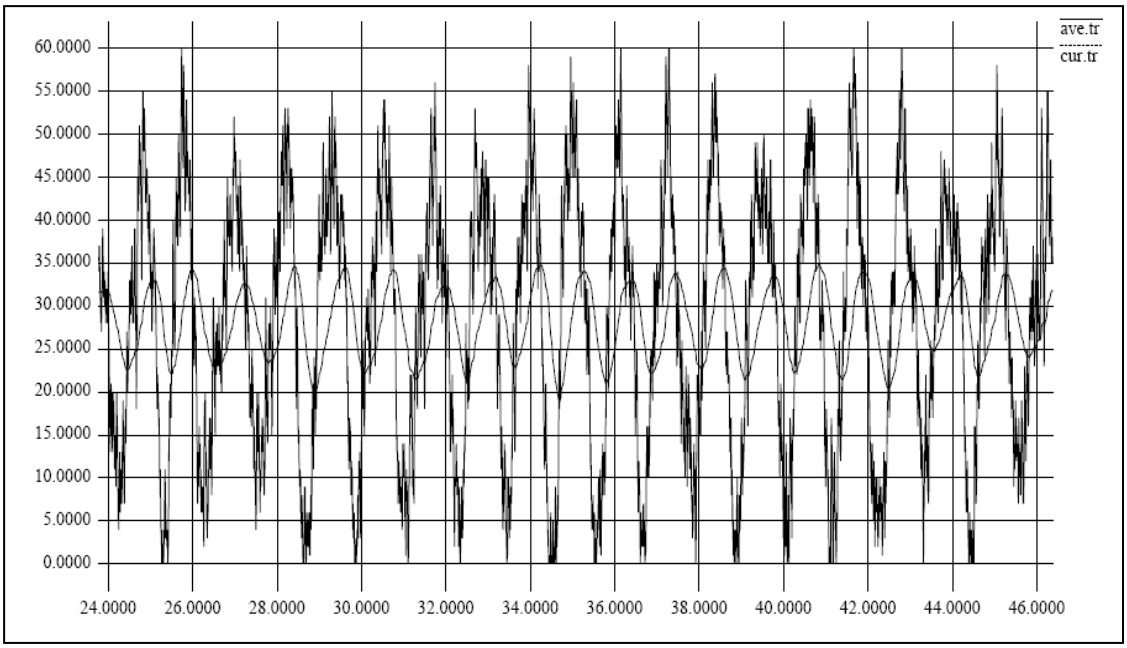

Figure 1. An example of chaotic queue oscillation with a queue size of 60 packets on 30 TCP flows

AutoRED Algorithm

Within the frame of RED mechanism, AutoRED modifies the way the weight, w_q, in EWMA is obtained, thereby affecting the calculated value of the average queue size. Therefore, it may be viewed as calculating the average queue size in two steps: first calculate w_q and then calculate the average queue size using the outcome of the first step. Instead of w_q being a constant value chosen by a user, AutoRED continuously calculates the value of w_q by a product of three mathematical functions modeling congestion characteristics, traffic characteristics, and queue normalization. Therefore, the mathematical notation of the average queue size at time t becomes (5) where w_q from the original RED algorithm is replaced by a time dependent $w_{q,t}$. And the total product of the three functions for calculating $w_{q,t}$ is denoted in (6).

$$q_{avg,t} = (1 - w_{q,t})q_{avg,t-1} + w_{q,t}\, q_t \qquad (5)$$

$$w_{q,t} = p_t(1 - p_t) \times \frac{2(5.923 + |\, q_t - q_{avg,\,t}\,|)}{\ln(5.923 + |\, q_t - q_{avg,\,t}\,|)} \times \frac{1}{qs} \qquad (6)$$

Equation (6) is broken down into the following three components;

$$\text{Congestion characteristics} = p_t(1 - p_t) \qquad (7)$$

$$\text{Traffic characteristics} = \frac{2(5.923 + |\, q_t - q_{avg,\,t}\,|)}{ln(5.923 + |\, q_t - q_{avg,\,t}\,|)} \qquad (8)$$

$$\text{Queue normalization} = \frac{1}{qs} \qquad (9)$$

It should be noted that the queue normalization is based on the queue size, *qs*, which is a fixed value. Therefore, the congestion characteristics and the traffic characteristics are contributing factors to varying $w_{q,t}$ values over time.

Modeling the congestion characteristics in (7) is derived from the probability law of Geometric distribution as following:

$$P = p_t(1 - p_t)^{n-1} \text{ where } n = 2 \qquad (10)$$

Reference [1] describes a random variable Y_t as a pointer for congestion at time *t* in this way:

$$Y_t = \begin{cases} 1 & \text{if } q_t - q_{avq,t-1} > 0 \\ 0 & \text{if } q_t - q_{avg,t-1} \leq 0 \end{cases} \qquad (11)$$

In other words, if the current queue size at time *t* is greater than the average queue size at time *t - 1*, it is regarded as an indicator that the network is heading for congestion at time *t + 1* and if the queue size is less, the network is not heading for congestion. Mathematically this sequence of trials whose outcome is either 1 or 0 can be construed as

Bernoulli trials. In this case, the outcome 1 means congestion and 0 means no congestion. From this, assuming p_t represents the probability that the network may head for congestion at time $t + 1$ based on Y_t at time t, it is stated that the probability law of Geometric distribution governs the number of trials required to formulate the first indication of congestion as in (10). The equation means that the queue status at time t represents the probability that the network is heading for congestion at time $t + 2$, which is in two steps. This notion is important in the new proposed scheme because the congestion characteristics value serves as a seed value in the logistic map used in the new scheme. More details are covered in the third chapter.

Now, p_t is calculated based on Y_t in the following way:

p_t = the number of outcome of Y_t being 1 over the total number of trials

Therefore, $(1 - p_t)$ = the number of outcome of Y_t being 0 over the total number of trials

This has been the process of calculating the congestion characteristics in AutoRED and its values can range from 0 to 0.25 mathematically.

The function (8) for the traffic characteristics is also time-dependent. Reference [1] states the mathematical function models the dynamics of the traffic characteristics in that it increases as the current queue size increases. Not only this aspect but it also satisfies what is called the "1- unit packet increase" effect in [1]. The 1-unit packet increase means when $q_t - q_{avg,t-1} = x_t$ and $q_{t+1} - q_{avg,t} = x_t + 1$. The effect of the number of unit packet increase at time t when there is 1-unit packet increase at time $t + 1$ is 50% at maximum which comes from $x_t = 1$ at time t. Then at time $t + 1$, $q_{t+1} - q_{avg,t}$ gives $x_t + 1$

which is 2. Therefore, the effect of 1-unit increase at time $t + 1$ over the total increase, $x_t + 1$, at time $t + 1$ (2 in this case) is 50%. Since x_t ranges $1 \leq x_t \leq qs$, the effect of the 1-unit packet increase diminishes as x_t increases. Based on this, [1] presents a mathematical function $f(x_t)$ that satisfies the condition of $|f(x_t + 1) - f(x_t)| \leq 0.5$. Function (8) is chosen as it satisfies these attributes mentioned above.

Reference [1] shows that AutoRED with the automatic calculation of the weight w_q produces adequate results of reducing chaotic queue oscillation that are visible in illustrations especially in a highly congested network. However, further research shows that AutoRED exhibits higher packet loss rates and tends to increase overall average queue size thereby increasing queuing delays as shown in Table I. While controlling chaotic queue oscillation by AutoRED shows positive results, the mechanism appears to be less than ideal in terms of packet loss rates and queuing delays. This may leave a room for further improvements particularly when fine tuning may be necessary.

As looking further into this matter, the possibility of improvements may be found by introducing a control parameter. In addition, currently no specific metric exists in measuring the effect of chaotic queue oscillation with respect to the QoS. This has been incorporated in the fourth chapter.

Table I. Comparison of network performance of RED and AutoRED in highly congested networks: RED exhibits the chaotic queue oscillation from 30 TCP flows and up. AutoRED values greater than those of RED are in bold.

	RED			**AutoRED**		
Number of TCP flows	*Link utilization*	*Packet loss rate*	*Overall average queue size*	*Link utilization*	*Packet loss rate*	*Overall average queue size*
10	99.60%	0.30%	16.28	99.41%	0.30%	16.35
20	99.81%	0.36%	24.08	99.76%	**1.29%**	22.98
22	99.72%	0.60%	26.58	99.69%	**1.99%**	24.31
23	99.49%	1.91%	27.60	**99.55%**	**2.64%**	25.39
25	99.72%	2.69%	28.16	**99.83%**	**3.14%**	27.04
30	99.22%	4.94%	28.83	**99.79%**	4.50%	28.09
40	99.36%	8.08%	30.67	**99.79%**	8.00%	30.63
50	99.64%	10.98%	32.32	**99.84%**	**11.68%**	**32.51**
60	99.73%	13.85%	33.64	**99.88%**	**14.06%**	33.57
70	99.82%	15.32%	34.14	**99.90%**	**15.86%**	**34.35**
80	99.83%	16.88%	34.75	**99.91%**	**17.38%**	**34.98**
90	99.87%	19.12%	35.59	**99.90%**	**19.27%**	**35.75**
100	99.88%	20.02%	36.05	**99.90%**	19.79%	35.97

CHAPTER III

A NEW PROPOSED TECHNIQUE

The logistic map is looked into mainly because the mathematical function of the congestion characteristics used in AutoRED displays a very similar pattern to that of the logistic map. Therefore, a new proposed technique called L_{map}-RED utilizes a logistic map function in place of the function of the congestion characteristics in AutoRED mechanism while the traffic characteristics and the queue normalization remain the same in computing the weight, $w_{q,t}$, in EWMA. Before describing further of the mechanism of L_{map}-RED, first the fundamentals of the logistic map are described.

Fundamentals of the Logistic Map

Logistic map is first presented by Robert May in his 1976 paper, analogous to the logistic equation which Pierre François Verhulst introduced in 1838 as modeling biological population growth [21]. The difference equation is following:

$$X_{n+1} = rX_n(1 - X_n) \qquad (12)$$

where $0 \leq X_n \leq 1$, X_n is the population size in the nth generation and hence X_0 is the initial population. As a model for biological population, the equation is designed to represent population growth as well as reduction due to overcrowding and limited resources in the environment. r value is meant to capture this combination of growth and reduction rate.

Depending on r value, this seemingly simple nonlinear dynamic map exhibits complex and chaotic behavior as following [21][22]:

- For $0 < r < 1$, the population goes extinct, regardless of the initial population.
- For $1 \leq r < 2$, the population rapidly reaches a steady state at $1 - \frac{1}{r}$.
- For $2 \leq r < 3$, the population first oscillates around $1 - \frac{1}{r}$ and eventually stabilizes on the value. When $r = 3$, the rate of convergence is very slow.
- For $3 \leq r < 1 + \sqrt{6}$ (approximately 3.449), as a population model, it alternates between a large population and a small population in one generation and another oscillating around $1 - \frac{1}{r}$. This is a period-2 cycle which repeats every two generations.
- For $3.449 \leq r < 3.544$ (approximately), the population oscillates between 4 values, repeating every four generations, a period-4 cycle. At around 3.544, the period doubles to 8, and then to 16 at around 3.564, to 32 at 3.568, and so on. Incidentally, this observation fits the period-doubling cascade with the ratio of two successive periods being 4.669…., the Feigenbaum Constant [21].
- Around $r = 3.5699$ is the onset of chaos and r values beyond that exhibits characteristics pertained to chaos.

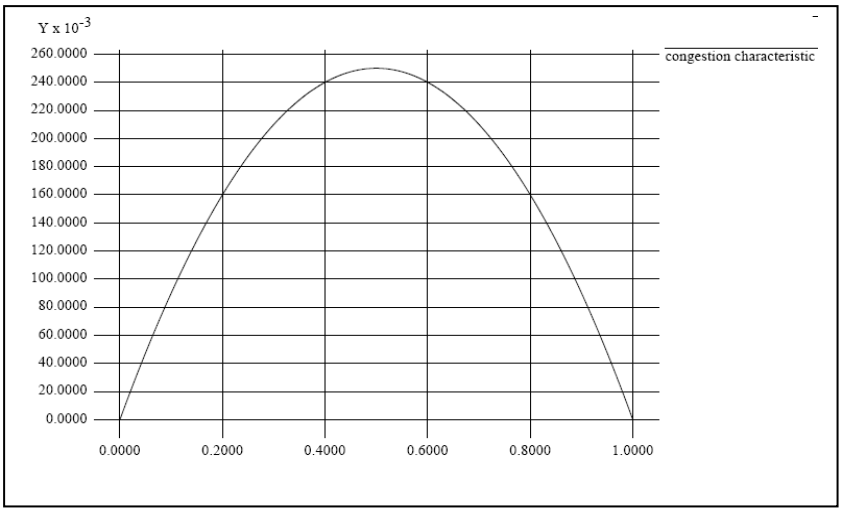

Figure 2. Congestion characteristics = $p_t(1 - p_t)$

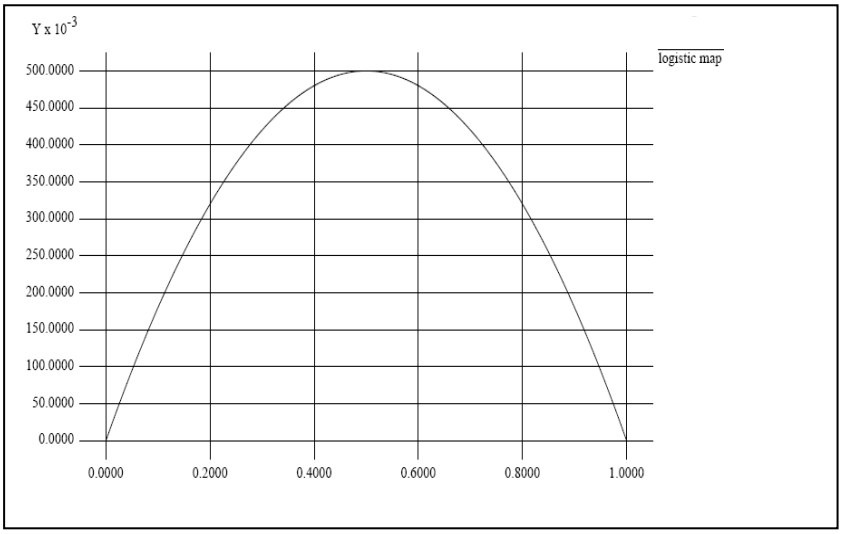

Figure 3. Logistic map $X_{n+1} = rX_n(1 - X_n)$ where $r = 2$

In [1], a mathematical model for the dynamics of the congestion characteristics is shown as a graph in Fig. 2. And a simple plot of (X_n, X_{n+1}) in the logistic map when $r = 2$ illustrated in Fig. 3 displays very similar pattern to that of the congestion characteristics. In the case of the logistic map, however, it should be noted that as r value increases and

the equation goes into the bifurcation region, the pattern changes. The pattern shown in Fig. 3 holds up to when $r = 3$. For $r > 3$, bifurcation starts. In addition, the maximum value at $X_n = ½$ is a quarter of r value. Hence, r value should be $0 \leq r \leq 4$ in order to meet the condition of $0 \leq X_n \leq 1$.

A New Technique: L_{map}-RED

Equation (13) shows the mathematical model of L_{map}-RED in which the function modeling the congestion characteristics in the AutoRED algorithm is replaced by the logistic map equation, X_t in (14).

$$w_{q,t} = X_t \times \frac{2(5.923 + | q_t - q_{avg,\,t} |)}{\ln(5.923 + | q_t - q_{avg,\,t} |)} \times \frac{1}{qs} \qquad (13)$$

$$X_t = r\, X_{t-1}(1 - X_{t-1}) \qquad (14)$$

where X_{0-1} is the initial value of the logistic map for calculating $w_{q,t}$ at time 0. In fact, in the network simulation setting for experiments covered in the fifth chapter, experiments start with the AutoRED algorithm and the moment when the AutoRED algorithm is switched over to L_{map}-RED is viewed as the start of the L_{map}-RED algorithm and the initial value of (14) is X_{0-1} and the time, t, at that moment is 0. Incidentally, it is at 5 seconds into the simulations when AutoRED is switched to L_{map}-RED in the experiments in the fifth chapter.

Therefore, equation (14) can be written as the following for the moment of the switch-over:

$$X_0 = r\, X_{0-1}(1 - X_{0-1}) \qquad (15)$$

Since L_{map}-RED does not start until time 0, time *0 – 1* as in X_{0-1} in (15) represents the previous step still in the AutoRED algorithm and its value is the congestion characteristics value at that time. Based on the meaning of the congestion characteristics in AutoRED defined by the "probability that the system can lead to congestion in two steps [1]," Fig. 4 illustrates this concept showing at the current time *t* denoted by a solid black dot predicts the congestion characteristics at time *t + 2*.

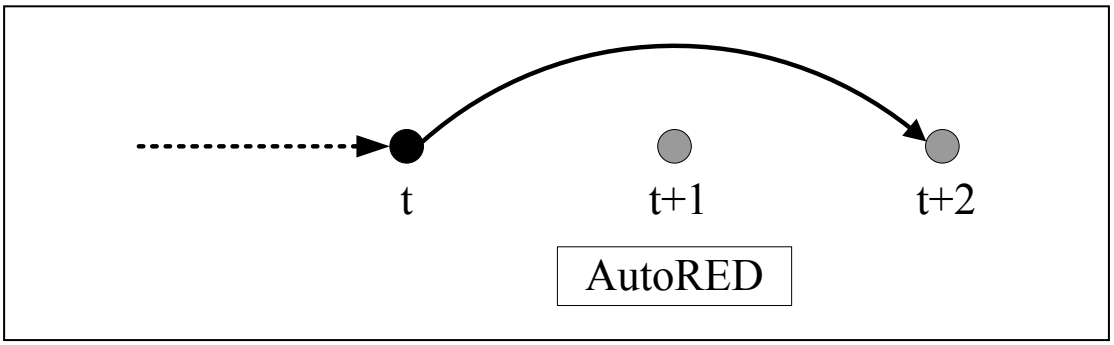

Figure 4. Congestion Characteristics predicting two steps ahead at time *t*

As this is applied to the switch-over to L_{map}-RED at time 0, the congestion characteristics value that L_{map}-RED inherits from the AutoRED algorithm is the probability value of what is likely to happen at time 0 + 1 having predicted it at time 0 – 1. Thus L_{map}-RED employs a value that has the attribute of predicting what is likely to happen at time 0 + 1 at time 0, which is essentially the next step. Fig. 5 illustrates this as below.

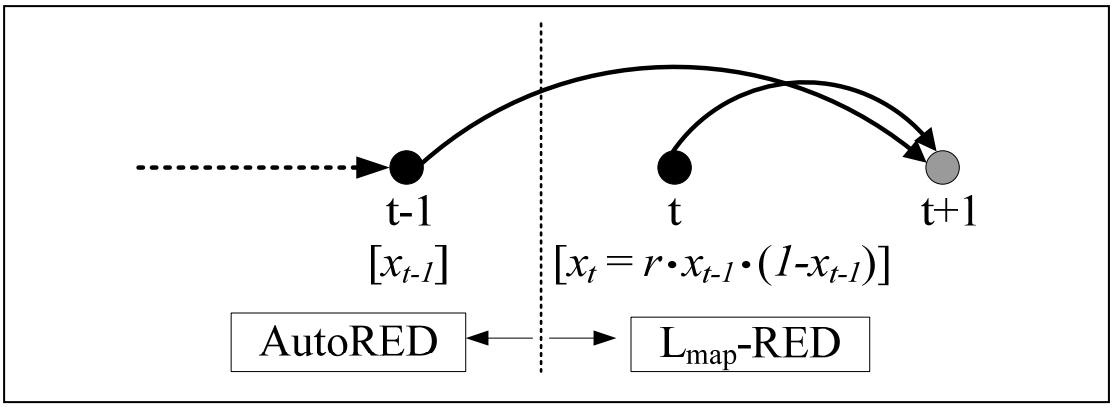

Figure 5. Congestion characteristics value at the switch-over

Therefore it can be said that the logistic map is employed in L_{map}-RED in this regard that the function of the logistic map follows the probability of the congestion characteristics in the next step as in population growth the equation is modeling.

CHAPTER IV

A NEW PROPOSED METRIC

Just as cars in a traffic jam alternate between moving forward and stopping, network traffics experience similar behavior at a router when queue oscillation occurs. In one moment a network flow stops moving as packets are admitted in the queue where sometimes it overflows. In the next moment, however, packets move forward with hardly any delay in the queue, but then only few sources are sending packets.

A severe case of this type of queue behavior known as chaotic queue oscillation in a router is to be avoided especially because of its effects on the QoS. It is an established fact that the throughput and the packet loss rate suffer when the chaotic queue oscillation occurs. These performance metrics are usually measured over a period of time and assist the understanding of the network status over such time. It is not as meaningful to use these metrics over a very short period due to the varying nature of network traffics. There are other effects of the chaotic queue oscillation on the QoS that are problematic even during a short period of time. Even if it may be remotely possible that a router with chaotic queue oscillation produces the same throughput and packet loss rate for a period of time as a router with stabilized queue oscillation, the effect of the chaotic queue oscillation on the QoS over a short period still remains. It is granted that problems for a short period of time only augment when they persist longer. Looking into the chaotic

queue oscillation, these effects have to be considered and possibly measured. This section describes two such problematic effects and it is followed by an attempt to create a new metric for such effects.

Background

When the effect of the chaotic queue oscillation on the QoS over a short period time is dissected, two problems stand out. First, large queuing delays for a cluster of packets at one moment and almost no delay at next increases jitter at a receiving end. There are greater chances of packets arriving out of order. Not only that, a group of packets may arrive after the wait time has passed, in which case the receiver would have sent a retransmission request to the sender. Therefore otherwise unnecessary network traffic is added in the pipeline and this may invoke unwarranted congestion avoid mechanism at the source. This is not desirable especially for time-sensitive traffics.

Second, in a valley of oscillation where the current queue size is very low, router resources are underutilized while in a peak period more packets are marked or even dropped due to an overflow. Queue overflow is what AQM mechanisms aim to eliminate. If the queue had not been widely fluctuating and the same number of packets had come in steadily, no packet drop would have been necessary and the resources utilized more effectively. In this respect, even with the same throughput and packet loss rate, stabilized queue oscillation produces better QoS than the chaotic queue oscillation.

From a customer's point of view, the network delay caused by a high queue size means low QoS. The longer a queue stays highly occupied (i.e. high queue size persists), the less the customer would be happy about the network performance. On the other hand,

when the queue is underutilized (i.e. low queue size persists) network service providers would be unhappy. The longer the queue stays that way, the less the service providers would be satisfied with router resource utilization. It is worth noting that not only the service providers but also those customers whose network transmissions had been backed off due to the decrease of TCP window size would experience low QoS since it is one of the main reasons why the queue becomes underutilized in the first place. So in terms of the QoS, note that not only the moment of high or low queue size but also how long or how persistently the queue stays the way it has been matters.

Therefore, it is important to be able to observe this aspect of queue behavior and to do so in an objective way. The method of calculating queuing delays presented in [23] can be one way, but it alone is not sufficient in portraying the behavior of queue size in that the method computes queuing delay based on the overall average queue size. It is probable that both a chaotic queue oscillation and a stabilized one may have approximately the same overall average queue size. Thus, it is not adequate as a metric in representing the degree of queue oscillation although the queuing delay based on the overall average queue size is still an important metric for the QoS in a router. Looking at the degree of queue oscillation with regard to its statistics, the minimum and the maximum queue size and the standard deviation may somewhat represent the behavior of queue oscillation coupled with the overall average queue size. It can then be said that the degree of queue oscillation is more severe statistically as its minimum value lower, the maximum value higher, and the standard deviation larger.

A New Metric: Segmented Time for QoS

In addition to these pointers, this thesis proposes a new metric called "Segmented Time for the QoS (Seg-time)" that measures queue oscillation in time indicating the effect of its behavior on the QoS. More specifically, Seg-time is to answer the question, "how long does it take for the queue size to go back to its average from this moment?" While the queue size goes up and down over time, Seg-time first measures, at each segment (hence the name), the difference in time between the timestamp of the instantaneous queue size and the timestamp of a corresponding reference point which is the timestamp when the queue size becomes the same as the overall average queue size as illustrated in Fig. 6 with a simplistic view of queue oscillation.

Whenever the instantaneous queue size returns to the same value as the overall average queue size of the duration of measurement, the timestamp at that moment is the reference point for all segments that come after the previous reference point. Hence the three reference points denoted by a perforated line in the figure lie at a cross section where the current queue size moving vertically meets the overall average queue size drawn horizontally. The horizontal arrow in the figure from Segment 1 to the next reference point corresponds to the duration of time that it takes to return to its overall average queue size. In the same manner the rest of the segments get their value. Then these values are averaged out for an overall measurement in a given duration. This average is the Seg-time for the duration.

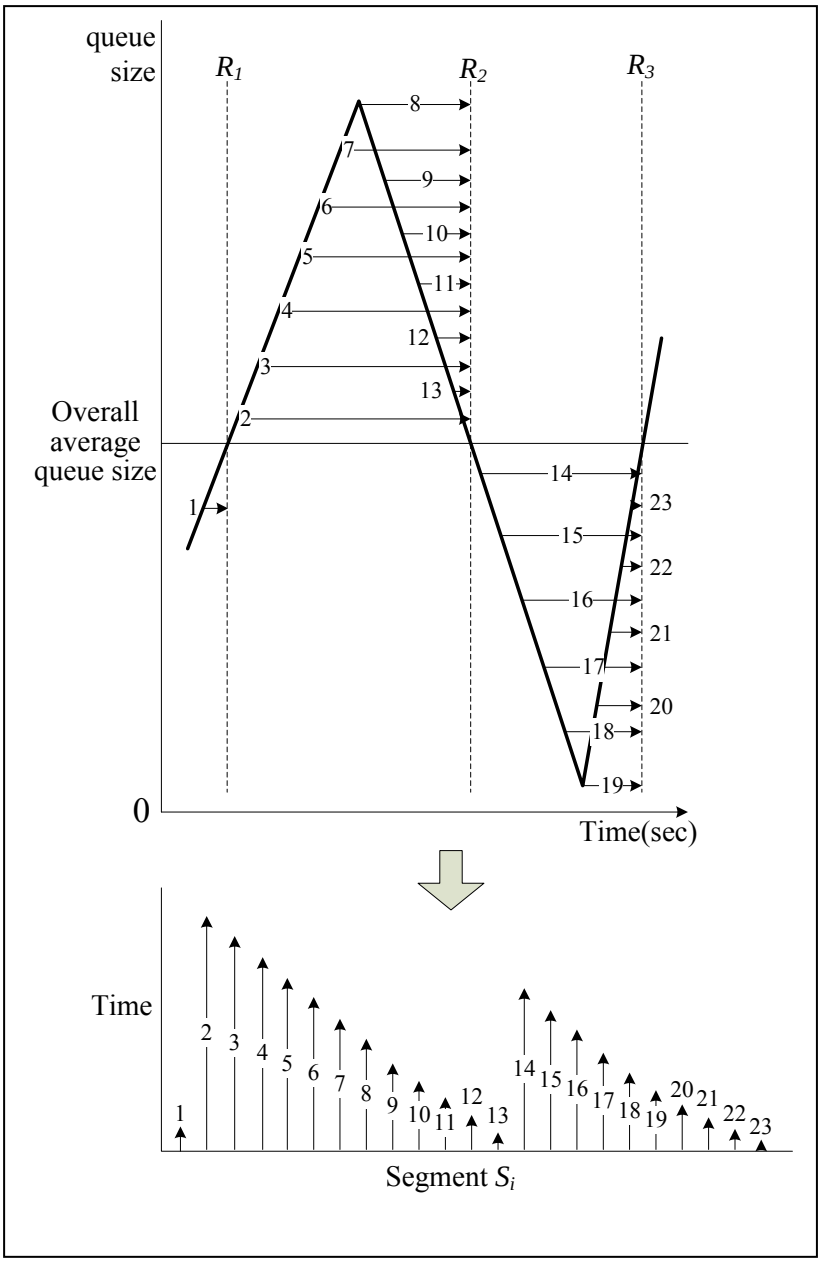

Figure 6. An simplified illustration of Seg-time measurement

The definition of Seg-time, ST_{avg}, in mathematical notation is presented as below.

Assuming the overall average queue size has been calculated for the duration of measurement, *D*. Then, *D* can be divided into a total number of *n* parts where each part is

composed of a segment portion, M_j, and a reference point R_j, such that $D = M_1R_1M_2R_2 \ldots M_jR_j \ldots M_{n-1}R_{n-1}\ M_nR_n$ where a reference point, R_j, is a timestamp when the current queue size is the same as the overall average queue size.

A segment portion M_j comprises segments, S_i, such that

$$M_1 = S_1 \ldots S_{m_1},$$

$$M_2 = S_{m_1+1} \ldots S_{m_2},$$

$$M_3 = S_{m_2+1} \ldots S_{m_3},$$

$$\vdots$$

$$M_j = S_{m_{j-1}+1} \ldots S_{m_j}$$

$$\vdots$$

$$M_n = S_{m_{n-1}+1} \ldots S_{m_n}$$

where S_i is a timestamp for the segment i and m_j is the total number of segments counted from M_1 to M_j. The total number of all segments in the measurement time D, thus, is m_n. And $m_j - m_{j-1}$ is the number of segments in the segment portion M_j which contains segments from the $m_{j-1}+1^{\text{th}}$, $S_{m_{j-1}+1}$, to the m_j^{th}, S_{m_j}.

Then, the sum of the time difference between segments and a reference point in each part, P_j, is calculated by

$$P_j = \sum_{i=m_{j-1}+1}^{m_j} (R_j - S_i)$$

Therefore, P_1, P_2, P_n, can be written as following:

$$P_1 = \sum_{i=1}^{m_1} (R_1 - S_i)$$

$$P_2 = \sum_{i=m_1+1}^{m_2} (R_2 - S_i)$$

⋮

$$P_n = \sum_{i=m_{n-1}+1}^{m_n} (R_n - S_i)$$

Then the total sum, ST_{sum}, can be written as:

$$ST_{sum} = \sum_{j=1}^{n} P_j = \sum_{j=1}^{n} \sum_{i=m_{j-1}+1}^{m_j} (R_j - S_i) \qquad (16)$$

Finally, Seg-time, ST_{avg} is computed by dividing ST_{sum} in (16) by the total number of segments in D, m_n

$$ST_{avg} = \frac{ST_{sum}}{m_n} \qquad (17)$$

This ST_{avg} is considered as the new metric that measures the effect of the degree of queue oscillation on the QoS.

Fig. 7 exhibits the plot of $(R_j - S_i)$ for each incoming packet for computing Seg-time in the network simulation of 30 TCP flows in the TCP-only environment. The behavior of the current queue size over the given time duration is shown in Fig. 8. The overall average queue size in this particular example is 28.8281 as indicated in Fig. 8 with a thick line. It can be observed that each cross-section of the overall average queue size and the actual current queue size in Fig. 8 corresponds to the value 0 in Fig. 7. Moreover, as the duration between one reference point and the next is longer in Fig. 8, the corresponding spike in Fig. 7 goes higher. The spike from around 16 seconds to a little over 17 seconds is circled in red as an example.

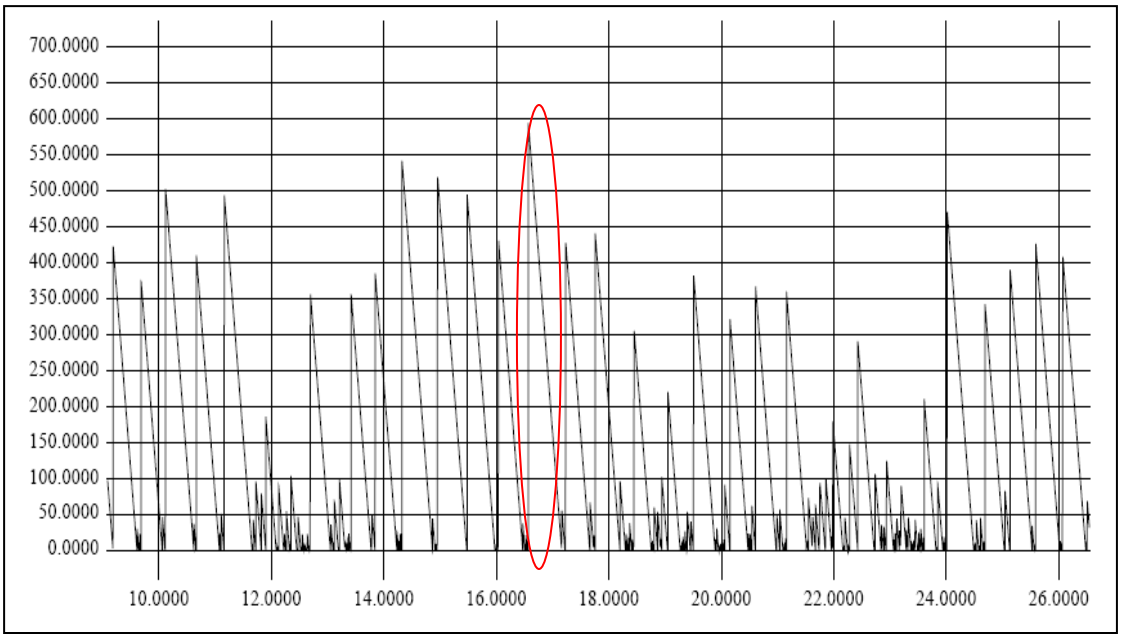

Figure 7. The plot of ($R_j - S_i$) for each incoming packet (X axis in ms)

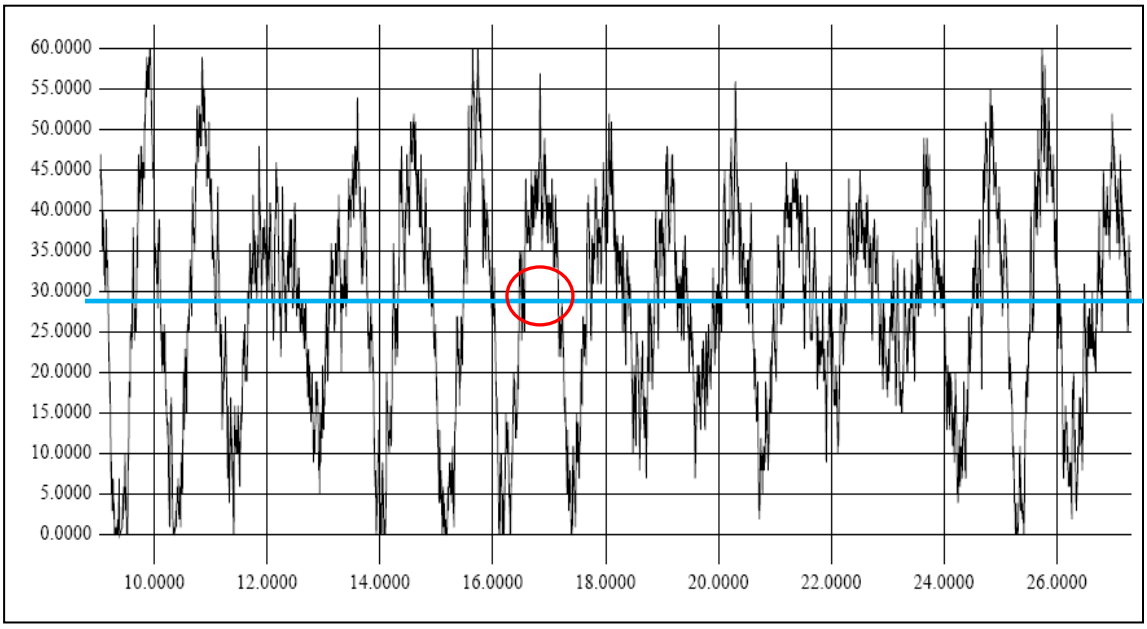

Figure 8. The current queue size behavior of 30 TCP flows in the TCP-only environment

One may ask why not use the time duration from one reference point to the next reference point so as to represent how long a period of high or low queue size persists. One reason why a time difference value between two reference points is not used is that whether processing 1,000 packets or 10,000 packets in the same time difference implies different levels of the QoS. Handling 10,000 packets would have the better QoS in terms of the delay per packet. Additionally, although queue oscillation appears to be periodic and sinusoidal, in a closer look it is not. Therefore, given the same value of the time difference, various forms of peaks or lows may exist. This is another reason why segmented intervals or each packet should be used for a meaningful measurement.

One may also ask why the overall average queue size is used as a basis for reference points. The overall average queue size can be regarded as a queue size that a particular AQM retains in a period of time considered for measurement. It can also be viewed as a negotiated point of the link utilization and the resource utilization that can reasonably satisfy both end users and service providers.

A larger Seg-time value means longer durations for the queue size to get back to the overall average value. If the queue is underutilized currently, a larger Seg-time means longer durations for the queue to exit the underutilization and get back up to the average level of occupation. Along this line of thinking, it can be said that the lesser value of the Seg-time means a better quality of service in terms of less jitter, less delay, and less underutilization of queue resources. Experiments in the next chapter display analogous results.

In the simulation experiments, the implementation of measuring Seg-time includes the followings: each incoming packet is regarded as a segment on its own. Since the overall average queue size is unlikely to be an integer, two integers are taken such that, for the overall average queue size, q_avg_{all} , $\lfloor q\_avg_{all} \rfloor$ and $\lceil q\_avg_{all} \rceil$ are regarded as the overall average queue sizes for the purpose of calculating Seg-time. For example, the simulation for the 30 TCP flows on RED in TCP-only environment yields 28.8281 as the overall average queue size as shown in Table II in the fifth chapter. Since $\lfloor 28.8281 \rfloor = 28$, $\lceil 28.8281 \rceil = 29$, timestamps of whenever the current queue size is either 28 or 29 are taken as reference points. Here is an excerpt of the movement of the current queue size over time taken from the same simulation:

Timestamp	queue size	
16.467793	27	
16.469025	27	
16.469047	**28**	**➔ reference point**
16.469127	**29**	**➔ reference point**
16.469213	30	
16.470257	30	
16.470445	31	

Furthermore, if the current queue size suddenly drops from a queue size greater than 29 to less than 28, or vice versa, the middle of the timestamps of the two points, one greater than 29 and the other less than 28, is taken as a reference point as below.

Timestamp	queue size	
16.028739	32	
16.031718	**30**	**16.03508 is taken as a reference point**
16.038431	**25**	
16.038856	26	

It should also be noted that a Seg-time value for the purpose of comparison of the degree of queue oscillation is only meaningful when all the topological parameters are kept the same.

CHAPTER V

RESULTS

Using NS-2 as the network simulation tool [24], experiments are performed in both a TCP-only environment and a TCP and UDP combined environment for RED, AutoRED and L_{map}-RED. As illustrated in Fig. 9, network topology for the simulation consists of *n* number of source nodes, the matching number of destination nodes, and one bottleneck link. This is known as a "dumbbell" configuration.

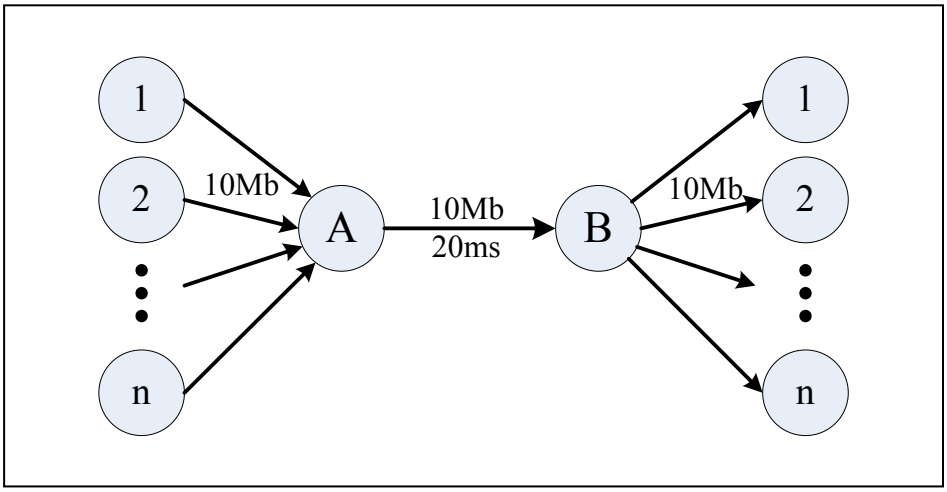

Figure 9. The network topology for experiments

The network and gateway parameter setups are as following:

- TCP type : Reno with window size at each source node = 8000

- TCP packet size = 1500 bytes
- UDP packet size = 1000 bytes for the TCP and UDP combined environment
- Bandwidth between node A and B = 10 Mbps
- Delay of the link between node A and B = 20 ms
- Queue size on the bottleneck link = 60 packets. For the ease of representation, queue size in packet is chosen rather than the queue size in byte option.
- Bandwidth of the links from source nodes to the bottleneck node A = 10 Mbps
- Delay of each link from source node to node A : a random number in [0 .. 35] ms
- Bandwidth of the links between node B and destination nodes = 10Mbps

The bottleneck queue parameters shared by RED, AutoRED, and L_{map}-RED are as follows:

- Minimum threshold = 10 packets
- Maximum threshold = 30 packets
- Maximum probability = 0.02
- Explicit Congestion Notification (ECN) is set to True. ECN allows the queue to set the ECN bit in the TCP header of incoming packets to notify source nodes of congestion status in order that source nodes can invoke the congestion avoidance mechanism.
- "gentle" parameter is set to True. This modification to RED is recommended by the authors of RED for robustness [25]. When this parameter is set to True, the packet-dropping probability increases from the maximum probability to 1 as the average queue size increases from the maximum threshold to twice the maximum

threshold. As a note, the authors claim that this option makes RED more robust to the setting of the parameters, the maximum threshold and the maximum probability. However, reference [26] shows that the option is not effective in stabilizing chaotic queue oscillation. Still this option is set to True in this experiments per the recommendation.

Aside from these common parameters shared with other mechanisms, RED uses a constant value for the weight parameter, w_q, set to 0.002. The same parameter is automatically calculated in AutoRED and L_{map}-RED.

In case of L_{map}-RED, in addition to the common parameters, r value from its algorithm (14) is controlled by user. Based on the characteristics of the logistic map behaviors varied by r value, eight r values are chosen for simulation. These values are composed of boundary and middle values and the justification of their selection is briefly described as below:

- 1.333: AutoRED simulations observe the congestion characteristic value approaching toward the value 0.25 in a steady state as in Fig. 10. And r value of 1.333 provides X_{n+1} stabilizing around 0.25. Supposedly this r value models the original AutoRED in a steady state. Therefore r value = 1.333 may generalize AutoRED in its steady state.

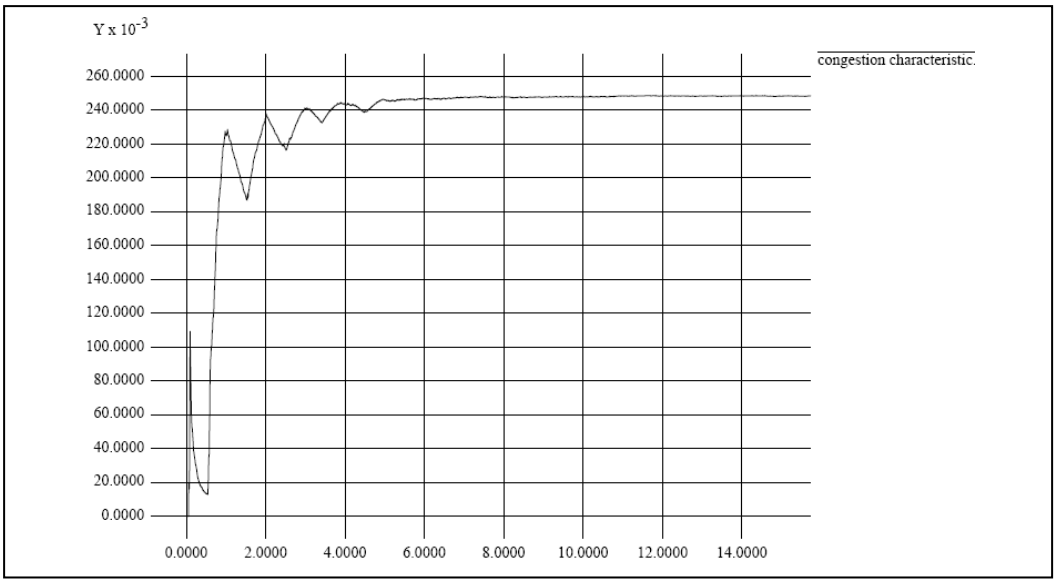

Figure 10. Congestion characteristics value in AutoRED for 40 TCP flows

- 1.5 : a middle value of the region of $1 \leq r < 2$ and X_{n+1} stabilizes on 0.333
- 2 : a boundary value of the region of $2 \leq r < 3$ and X_{n+1} stabilizes on 0.5
- 2.5 : a middle value of the region of $2 \leq r < 3$ and X_{n+1} stabilizes on 0.6
- 3 : a boundary value of the region of $2 \leq r < 3$ and X_{n+1} eventually stabilizes on 0.666 though very slowly
- 3.2 : a middle value of the region of $3 \leq r < 1 + \sqrt{6}$ (≈ 3.449) where X_{n+1} oscillates between two values
- 3.5 : a middle value of the region of $1 + \sqrt{6}$ (≈ 3.449) $\leq r < 3.544$ where X_{n+1} oscillates between four values
- 3.58 : a boundary value of the region where the logistic map exhibits chaos

The main variable in the experiment is the number of network flows. In the TCP-only environment, the number of TCP network flows is increased from 5 to 240 so as to

control the level of congestion in the network. In the TCP and UDP combined environment, the number of TCP flows is set to 10 while the number of UDP network flows is varied from 5 to 50. This way the effect of the UDP flows can be observed. In case of L_{map}-RED, the eight *r* values are run per each number of flows to examine the effect of varying *r* values in the same congestion level in both environments.

Time control scenario of network traffic in each simulation is as follows. In both TCP-only and TCP and UDP combined environments, each simulation last 60 seconds. It is assumed that the network should be in steady state after 5 seconds. For the first 5 seconds, the original RED algorithm runs and then at the 5^{th} second AutoRED or L_{map}-RED algorithm takes over. In L_{map}-RED, it should be noted that the initial value X_{0-1} in (15) comes from calculating the congestion characteristics value of AutoRED in (7) for the first 5 seconds. Although it is RED that runs as the queue management mechanism during that time period, the computation runs in parallel in order to seed L_{map}-RED algorithm properly. At the 5^{th} second when the simulation transitions to L_{map}-RED from RED, the congestion characteristics value of AutoRED at the previous step is plugged in to X_{0-1} in the L_{map}-RED algorithm in (15).

Simulation results are presented with link utilization and packet loss rate as performance metrics. Additionally, overall average queue size, minimum queue size, maximum queue size, and standard deviation are included as statistical measures for the comparison of the degree of queue oscillation. Also Seg-time is presented for measuring the effect on the QoS by queue oscillation.

As a note, link utilization is calculated from throughput in Mbps. Total throughput is divided by the maximum bandwidth of the bottleneck link which is 10 Mbps in the experiments. The percentile of this value is the link utilization. Depending on the context, link utilization and throughput may be used interchangeably in the thesis.

In assessing network performance metrics, it is obvious that higher link utilization and lower packet loss rate are regarded as more favorable. Statistical metrics for the degree of queue oscillation represent more stabilized queue behavior when minimum queue size is higher, and maximum and standard deviation lower. Likewise, lower Seg-time is interpreted as leading to a better QoS.

More careful approach is called for in assessing queuing delays. It should be noted that higher queuing delays are necessary for better throughput as long as the overall average queue size is under the maximum threshold parameter in RED and its variants. Maintaining the overall average queue size to maximize the network power is one of the design goals of RED. Keeping the overall average queue size at a certain level inevitably requires the corresponding queuing delay because the only varying factor in the computation of queuing delay is the overall average queue size based on the method presented in [23]. This method is used for the experiments presented.

The formula for calculating the queuing delay, D_q, is as below:

$$D_q = \frac{\text{Overall average queue size} \times \text{packet size (data + header)} \times 8 \text{ (bit)}}{\text{Maximum bandwidth of the link in bps}}$$

TCP-only Environment

The number of TCP Reno flows in each run of simulation is accrued from 5 to 240 in varied increments to congest the network at the bottleneck. All of the TCP flows start at the beginning with randomized delays ranging from 0 to 35 ms and the measurement of statistical metrics and Seg-time starts from 5.5 seconds. This is to filter out any transient behaviors that may occur when the queue management mechanism changes at 5 seconds.

Comparison of RED and AutoRED and a Validation of Seg-time

First of all, the performance metrics for RED and AutoRED are compared. As Fig. 11 shows, AutoRED yields higher link utilization in more congested networks where the number of TCP flows is 23 and above.

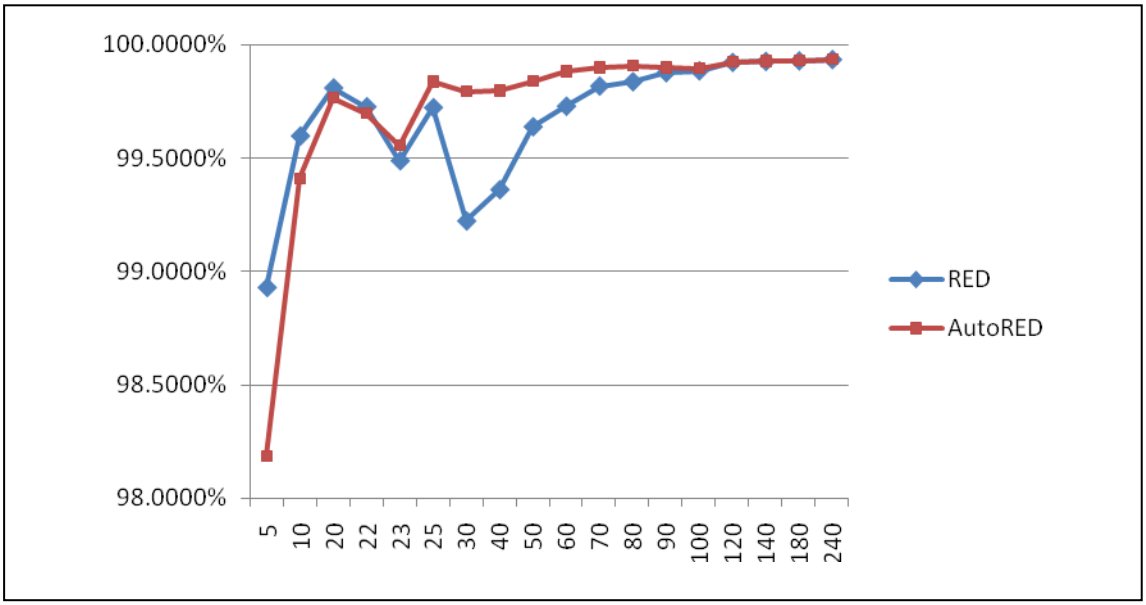

Figure 11. Comparison of link utilization of RED with AutoRED

On the other hand, packet loss rates of AutoRED are consistently higher than those of RED and queuing delays display no visible improvements in highly congested networks as shown in Fig. 12 and Fig. 13 respectively. The actual data of the results of these simulations are presented in Table II.

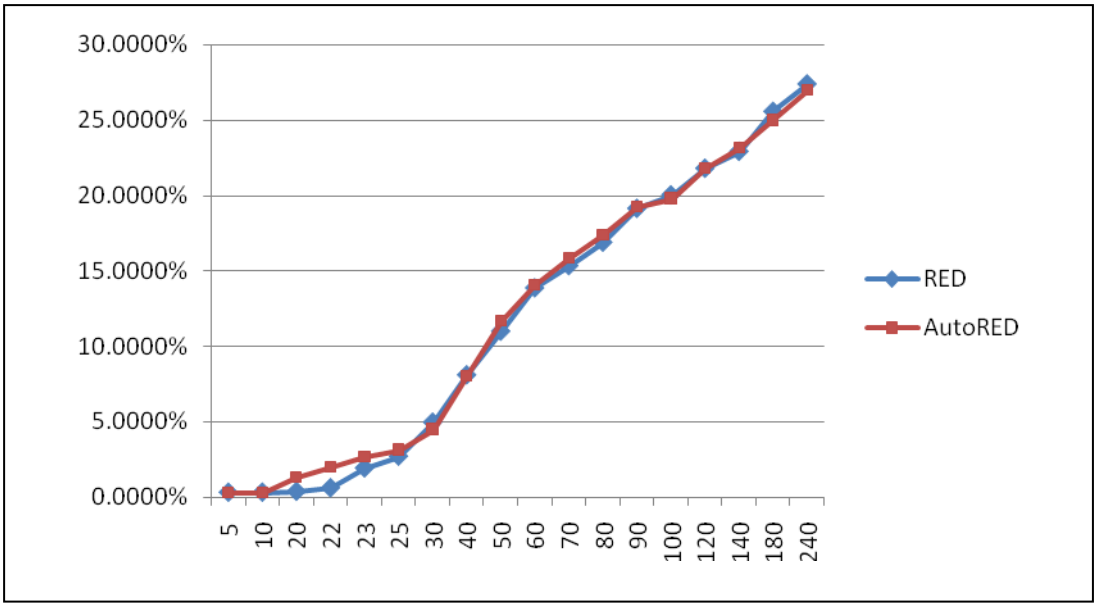

Figure 12. Comparison of packet loss rate

In these simulations, both RED and AutoRED exhibit the overall average queue size under 30, which is the maximum threshold, up to 30 TCP flows as shown in Table II. Therefore, higher queuing delays in RED queue for the number of flows up to 30 is considered more favorable in terms of throughput. This is analogous of the simulation results showing higher throughput in RED queues up to 22 TCP flows. However, the RED throughput drops from 23 TCP flows. The simulations with 23, 25, and 30 TCP flows have lower throughput even though they exhibit higher overall average queue size

under the maximum threshold. The statistical metrics and Seg-time measurements coincide with the network performance results in that the values of minimum and maximum queue size, standard deviation, and Seg-time on RED scheme are assessed unfavorable exactly in the same region that the network performance seems poor. This shows, first, that there exists a strong relation between the degree of queue oscillation and the network performance. Secondly, this also confirms the improvements made by AutoRED of the network performance through reducing the chaotic queue oscillation. Last but not least, this serves as a validation of Seg-time as an applicable metric representing the effect of queue oscillation on the QoS.

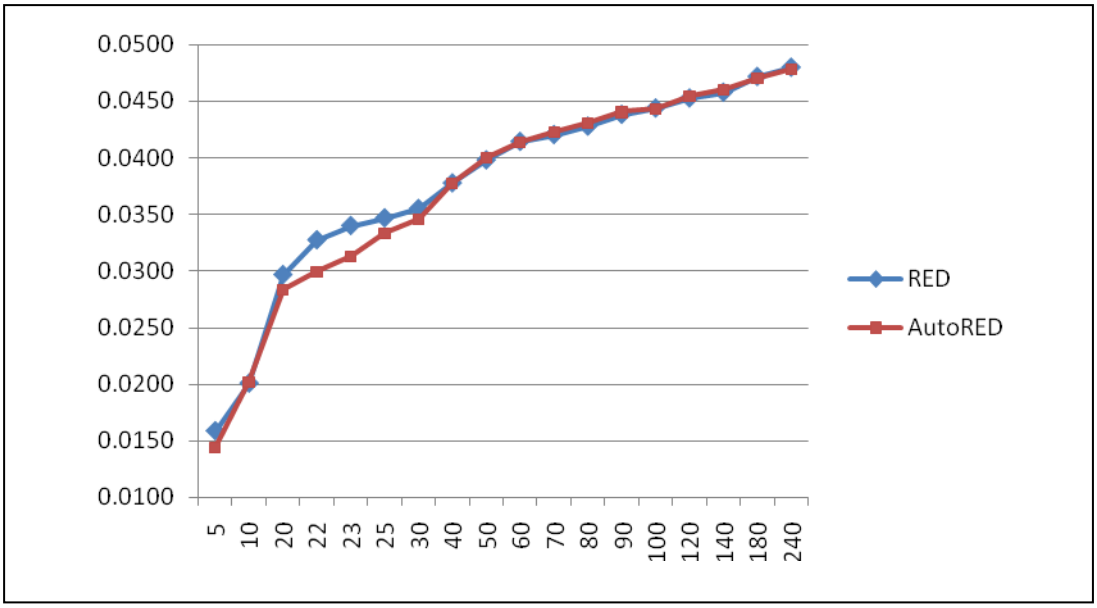

Figure 13. Comparison of queuing delay

Table II. Comparison data of network performance metrics

	RED				**AutoRED**			
Number of TCP flows	*Link utilization*	*Packet loss rate*	*Overall average queue size*	*Queuing delay*	*Link utilization*	*Packet loss rate*	*Overall average queue size*	*Queuing delay*
5	98.9284%	0.2979%	12.8534	0.0158	98.1856%	0.3002%	11.7040	0.0144
10	99.5956%	0.3020%	16.2785	0.0201	99.4083%	0.3047%	16.3520	0.0201
20	99.8070%	0.3565%	24.0809	0.0297	99.7633%	1.2876%	22.9815	0.0283
22	99.7245%	0.5972%	26.5759	0.0327	99.6938%	1.9856%	24.3068	0.0299
23	99.4863%	1.9120%	27.6037	0.0340	99.5543%	2.6408%	25.3937	0.0313
25	99.7205%	2.6873%	28.1569	0.0347	99.8331%	3.1374%	27.0373	0.0333
30	99.2215%	4.9416%	28.8281	0.0355	99.7921%	4.4967%	28.0922	0.0346
40	99.3591%	8.0835%	30.6688	0.0378	99.7945%	8.0037%	30.6343	0.0377
50	99.6361%	10.9782%	32.3210	0.0398	99.8353%	11.6832%	32.5137	0.0401
60	99.7265%	13.8491%	33.6447	0.0415	99.8785%	14.0645%	33.5703	0.0414
70	99.8152%	15.3224%	34.1401	0.0421	99.8970%	15.8571%	34.3478	0.0423
80	99.8342%	16.8809%	34.7464	0.0428	99.9052%	17.3782%	34.9815	0.0431
90	99.8741%	19.1234%	35.5908	0.0438	99.8970%	19.2678%	35.7544	0.0440
100	99.8824%	20.0174%	36.0479	0.0444	99.8950%	19.7927%	35.9658	0.0443
120	99.9196%	21.7804%	36.7682	0.0453	99.9202%	21.8081%	36.8767	0.0454
140	99.9236%	22.9056%	37.1838	0.0458	99.9236%	23.1668%	37.3555	0.0460
180	99.9263%	25.5528%	38.3172	0.0472	99.9256%	24.9881%	38.1792	0.0470
240	99.9317%	27.3717%	38.9788	0.0480	99.9321%	26.9708%	38.8190	0.0478

Now moving on to the comparison of statistical metrics and Seg-time, the comparison of minimum and maximum queue size, standard deviation, and Seg-time are illustrated in Fig. 14, Fig. 15, and Fig. 16, respectively. The detailed data are presented in Table III.

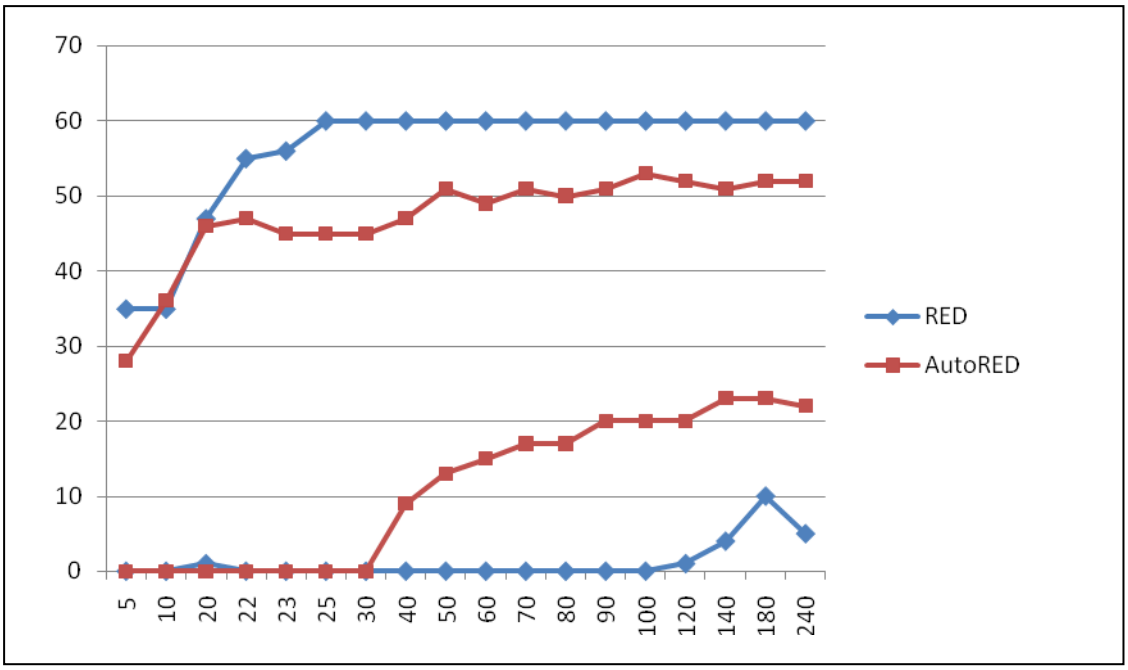

Figure 14. Comparison of minimum and maximum queue size

Fig. 14 exhibits that from 23 TCP flows maximum queue sizes of RED scheme leaps close to the maximum queue capacity which is 60 packets whereas those of AutoRED increase very gradually. Over 30 TCP flows, minimum queue sizes of RED are smaller than those of AutoRED. In fact, up until 120 flows, the queue oscillation of RED hits the bottom of the queue at 0 packet level. While the range of queue oscillation on AutoRED scheme is mostly contained between 15 and a little over 50 in highly congested networks, the queue oscillation on RED shows much higher degree by 25 packets. This clearly shows that RED contains the elements of wider oscillation vertically. This behavior is consistent in the number of flows on or above 23.

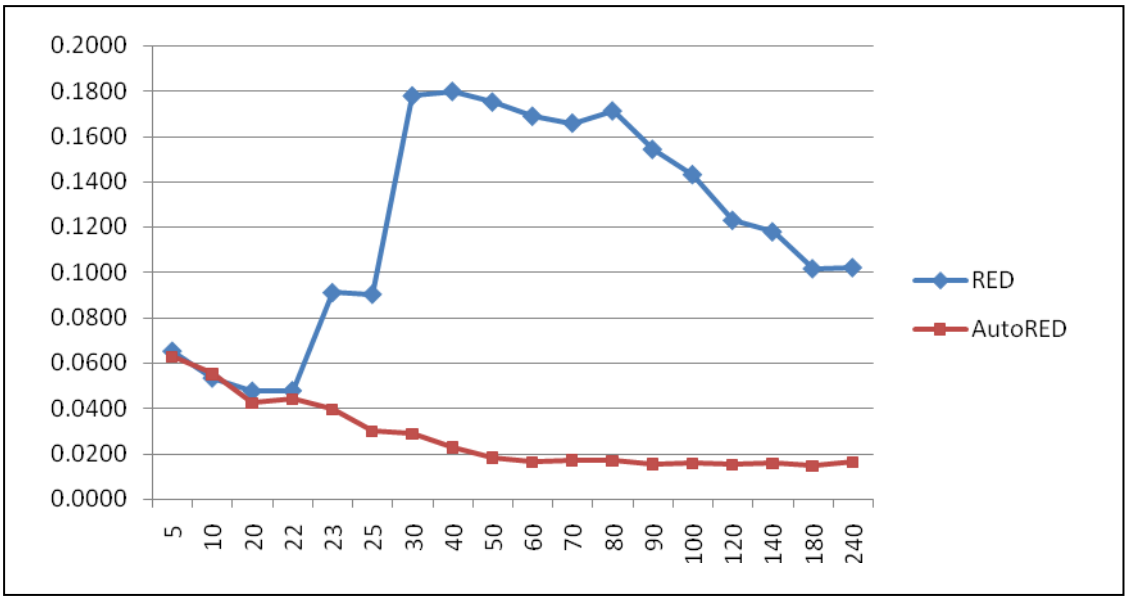

Figure 15. Comparison of standard deviation

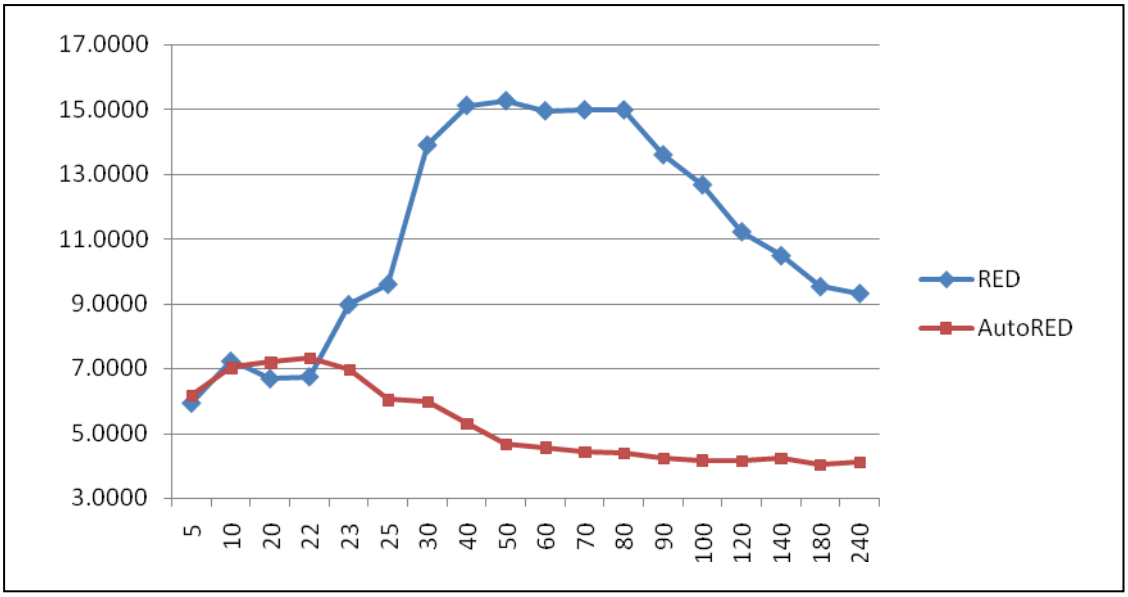

Figure 16. Comparison of Seg-time

As displayed in Fig. 15, the region below 23 flows shows contained values of standard deviation for both RED and AutoRED schemes with those of AutoRED showing

slightly higher values. Starting from 23 flows, standard deviation values of RED jump and stay high until around 80 flows and start decreasing but still over twice as high as those of AutoRED. While the standard deviation values start increasing on RED mechanism at 23 flows, those of AutoRED actually start decreasing gradually, meaning more controlled queue behavior. This observation agrees with the minimum and maximum queue size trend in Fig. 14.

Seg-time trend illustrated in Fig. 16 seems analogous to that of standard deviation in that there is a small leap at 23 flows and a big leap at 30 flows on RED mechanism retaining high values until around 80 flows and then it starts abated. Seg-time on AutoRED also shows similar values until 23 flows where RED and AutoRED part their ways. While Seg-time values on RED soar, those on AutoRED are gradually reduced in a pattern similar to the trend of standard deviation values.

It is striking that the pattern of standard deviation and that of Seg-time seem to be almost identical in that the number of TCP flows on or above 23 exhibits much higher standard deviation and longer Seg-time on RED mechanism over those of AutoRED. And after around 80 TCP flows the values of both metrics on RED appear to be abated although they are still higher than those of AutoRED. In a very simplistic view, standard deviation can be considered as representing how high and low the queue oscillation varies vertically from the overall average queue size whereas Seg-time can be regarded as describing how wide each peak and valley of the queue oscillation is horizontally. While vertical swing implies possible overflows of the queue and underutilization of the resources, horizontally wide peaks and valleys imply that the possible problems persist

degrading the level of the QoS further. Expectedly these two representations of the same queue oscillation go hand in hand since it can be viewed as looking at the same behavior from two different angles. And the relation of standard deviation and Seg-time in the measurements of these simulations affirms this.

Table III. Comparison data of statistical metrics for queue oscillation

	RED				**AutoRED**			
Number of TCP flows	*mini mum*	*maxi mum*	*standard deviation*	*Seg-time*	*mini mum*	*maxi mum*	*standard deviation*	*Seg-time*
5	0	35	5.9470	0.0651	0	28	6.1598	0.0628
10	0	35	7.2376	0.0533	0	36	7.0119	0.0552
20	1	47	6.7018	0.0474	0	46	7.1774	0.0425
22	0	55	6.7582	0.0477	0	47	7.3068	0.0440
23	0	56	8.9831	0.0909	0	45	6.9629	0.0396
25	0	60	9.6099	0.0902	0	45	6.0229	0.0301
30	0	60	13.9038	0.1777	0	45	5.9705	0.0290
40	0	60	15.1281	0.1797	9	47	5.2796	0.0229
50	0	60	15.2728	0.1750	13	51	4.6592	0.0183
60	0	60	14.9611	0.1687	15	49	4.5476	0.0163
70	0	60	14.9927	0.1657	17	51	4.3933	0.0173
80	0	60	14.9903	0.1711	17	50	4.3792	0.0170
90	0	60	13.6055	0.1541	20	51	4.2032	0.0156
100	0	60	12.6802	0.1430	20	53	4.1458	0.0158
120	1	60	11.2312	0.1228	20	52	4.1361	0.0155
140	4	60	10.4931	0.1177	23	51	4.1979	0.0158
180	10	60	9.5397	0.1015	23	52	4.0144	0.0149
240	5	60	9.3287	0.1020	22	52	4.0947	0.0163

In summary, the results of the comparison of minimum and maximum queue size, standard deviation, and Seg-time show that the simulations on RED scheme exhibit a

much higher degree of queue oscillation both vertically and horizontally than those on AutoRED, especially in the region on or above 23 TCP flows. Also it should be noted that the region exhibiting the greatest difference in their values between RED and AutoRED, which is from around 23 TCP flows up to about 80, coincides with the lower throughput region of RED in Fig. 13. Therefore, this observation seems to confirm the strong relation between high degrees of queue oscillation and poor network performance. The results also tell that AutoRED improves network performance by controlling the chaotic queue oscillation although it shows higher packet loss rate.

Not unexpectedly, the region from 5 TCP flows to 22 flows showing better throughput and lower packet loss rate on RED displays more favorable standard deviation and Seg-time values meaning that their queue oscillation is controlled. In addition, better performances in lower numbers of flows by RED can be interpreted as RED performing well in less congested networks.

This result also supports the validity of Seg-time as an applicable measure in that Seg-time displays a near linear relation to standard deviation in representing the degree of queue oscillation. Moreover, it is also affirmed by the fact that the effect of the degree of queue oscillation represented by Seg-time agrees with the proven relation between the network performance and the degree of queue oscillation.

Another validation method of Seg-time is comparing the Seg-time data and visual representations of queue behavior. Illustrations of queue behavior over time have been used for exhibiting the existence or the degree of chaotic queue oscillation.

Notice that in Table III the values of Seg-time in RED column jump from 0.0477 seconds to 0.1777 seconds from 20 flows to 30 flows with 23 and 25 showing a small amount of increase. On the other hand, Seg-time on AutoRED on 30 flows is still maintained in the order of $1/100^{th}$ seconds as before. In fact, the Seg-time values on AutoRED gradually decrease from 5 flows as the number of flows increases. Fig. 17 illustrates queue behavior over time with 20, 22, 23, 25, 30, and 40 TCP flows on RED on the left and on AutoRED on the right column in the next two pages. This visual comparison of queue behaviors on RED and AutoRED seems equivalent to the numeric values in Table III. This also validates the applicability of Seg-time as an appropriate metric in measuring the degree of queue oscillation particularly in relation to its effect on the QoS.

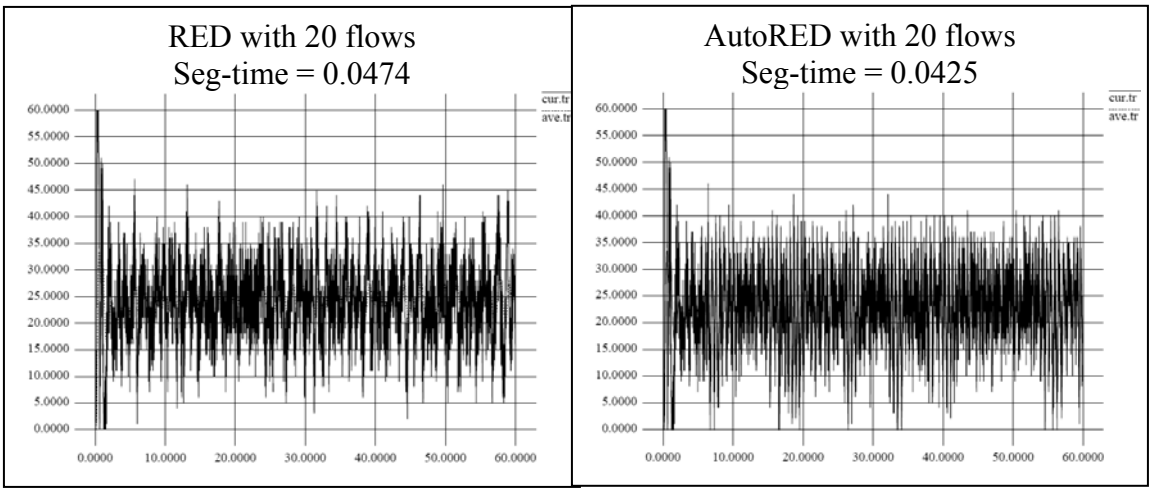

Figure 17. Visual comparison of queue behavior on RED and AutoRED

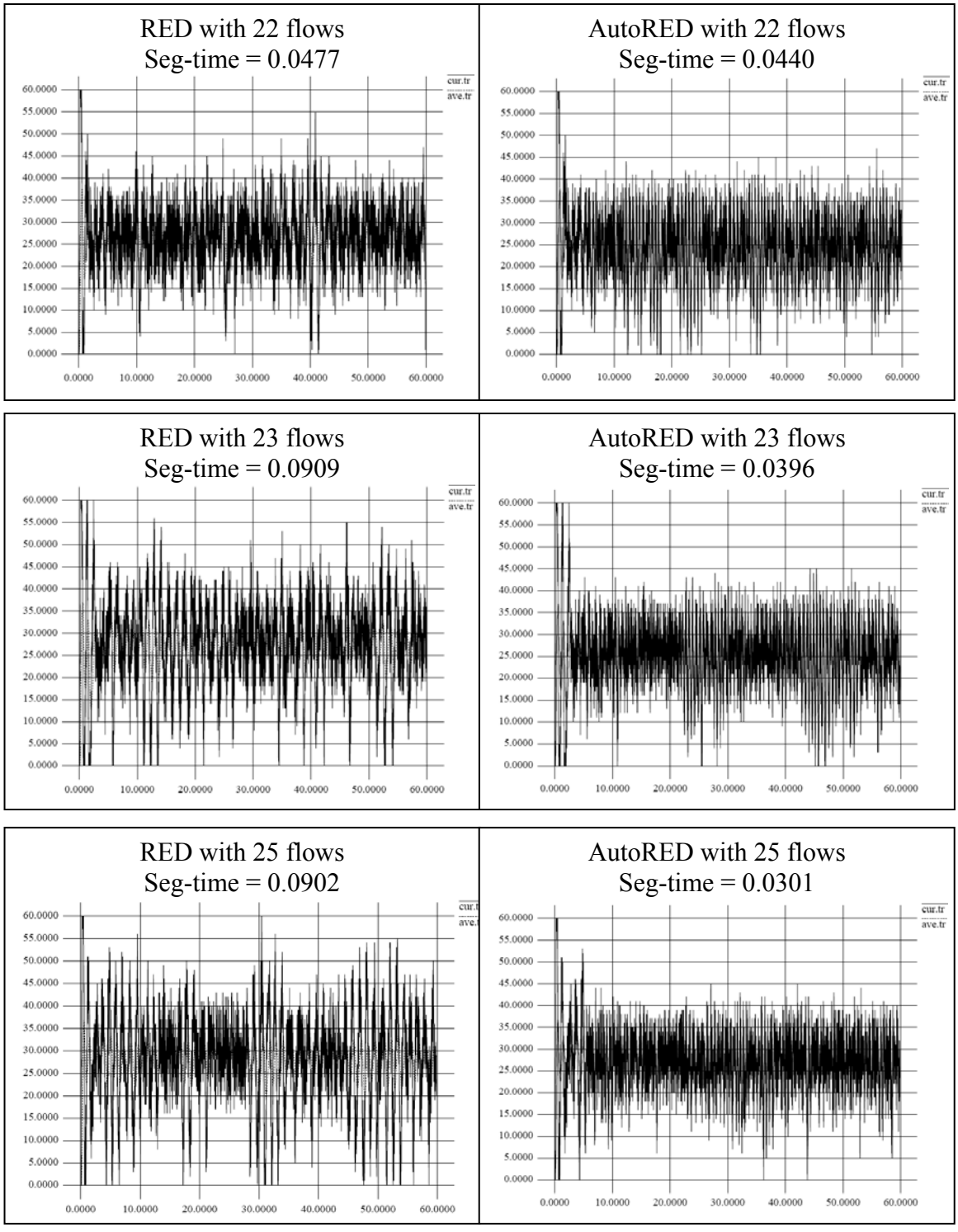

Figure 17. Visual comparison of queue behavior - continued

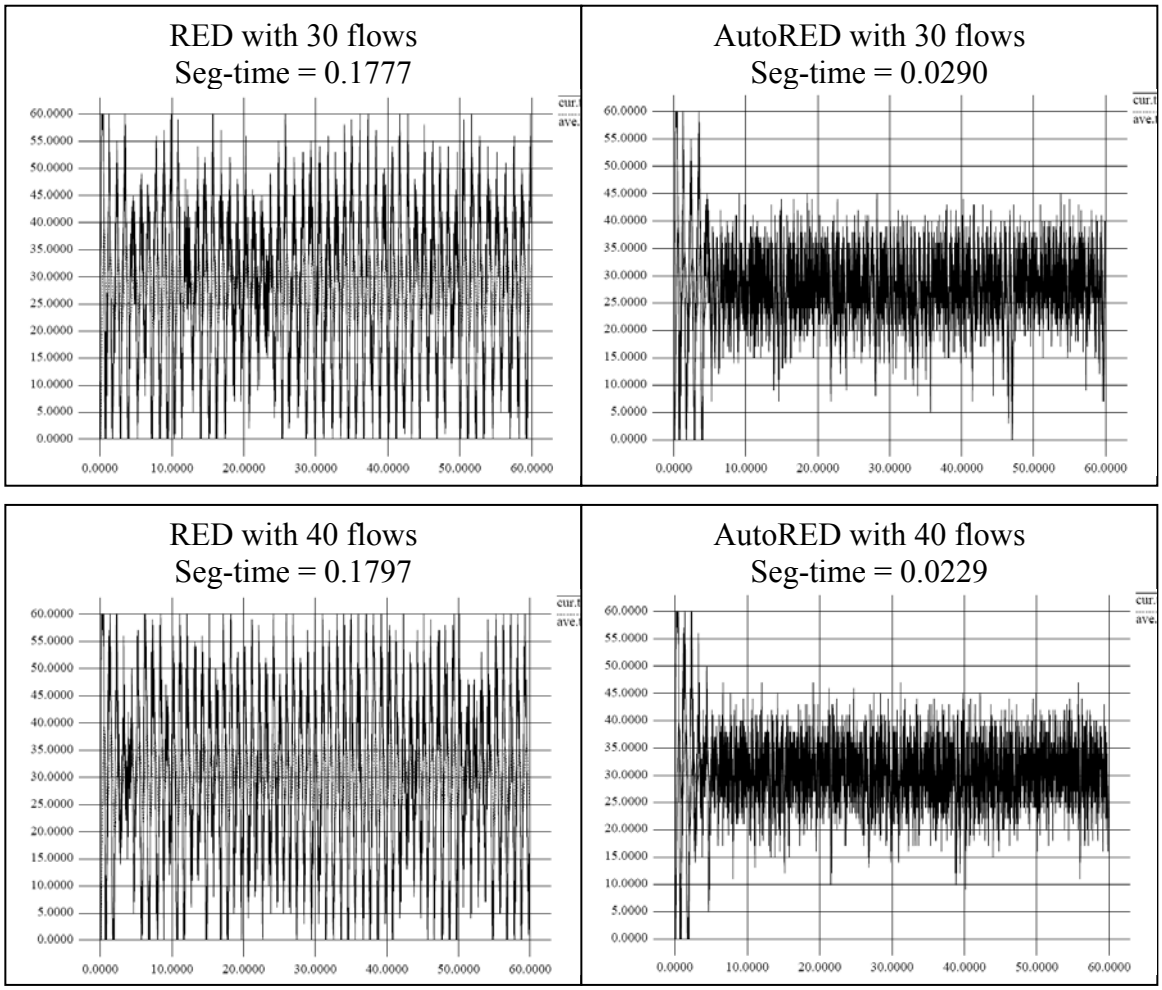

Figure 17. Visual comparison of queue behavior - continued

L_{map}-RED Simulation Results

On each number of TCP flows, eight simulations are run with eight different *r* values. Then the best result of each metric for network performance and degree of queue oscillation is selected with its corresponding *r* value. The full data can be found in Appendix A. 1.

Overall, network performance results of the simulations on L_{map}-RED approximate to those of AutoRED with slight but consistent improvements. The measurement results of link utilization and packet loss rates are shown in Fig. 18 and Fig. 19 respectively. The detailed data shown in Table IV display the steady improvements in both categories.

On link utilization, less congested networks under 25 TCP flows exhibit larger improvements. As for packet loss rates, improvements range from 0.001% to 0.6% throughout all numbers of flows except for 240.

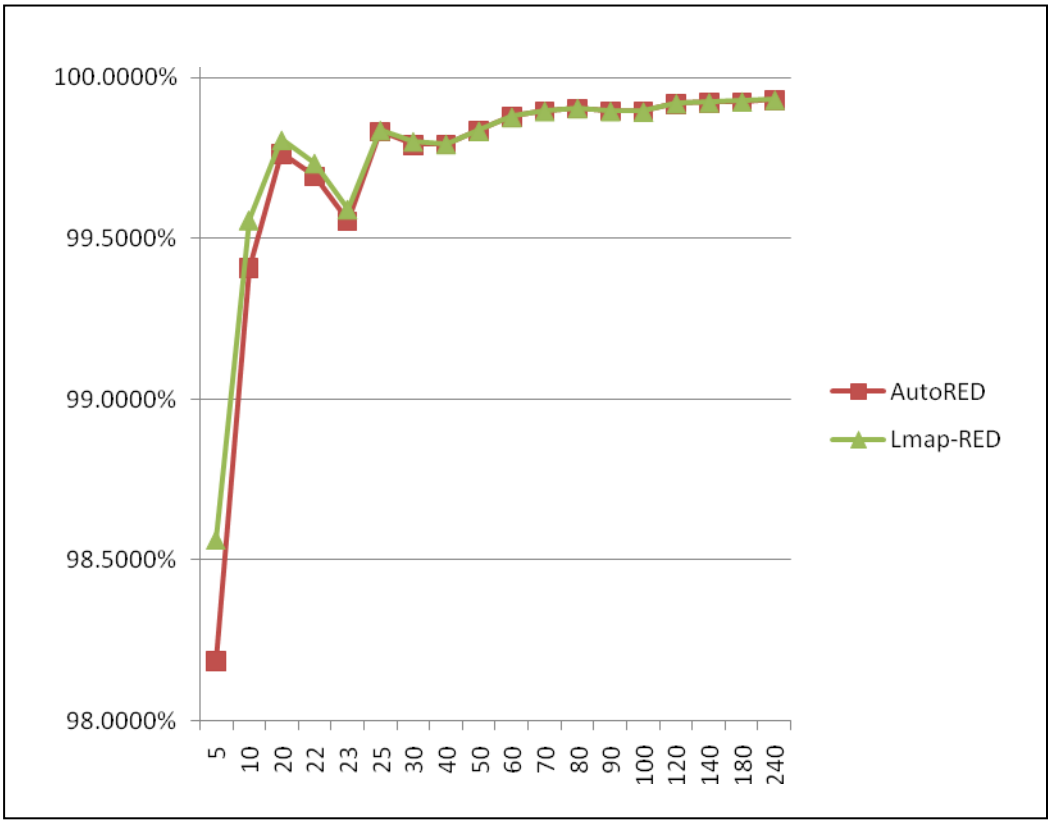

Figure 18. Comparison of link utilization on AutoRED and L_{map}-RED

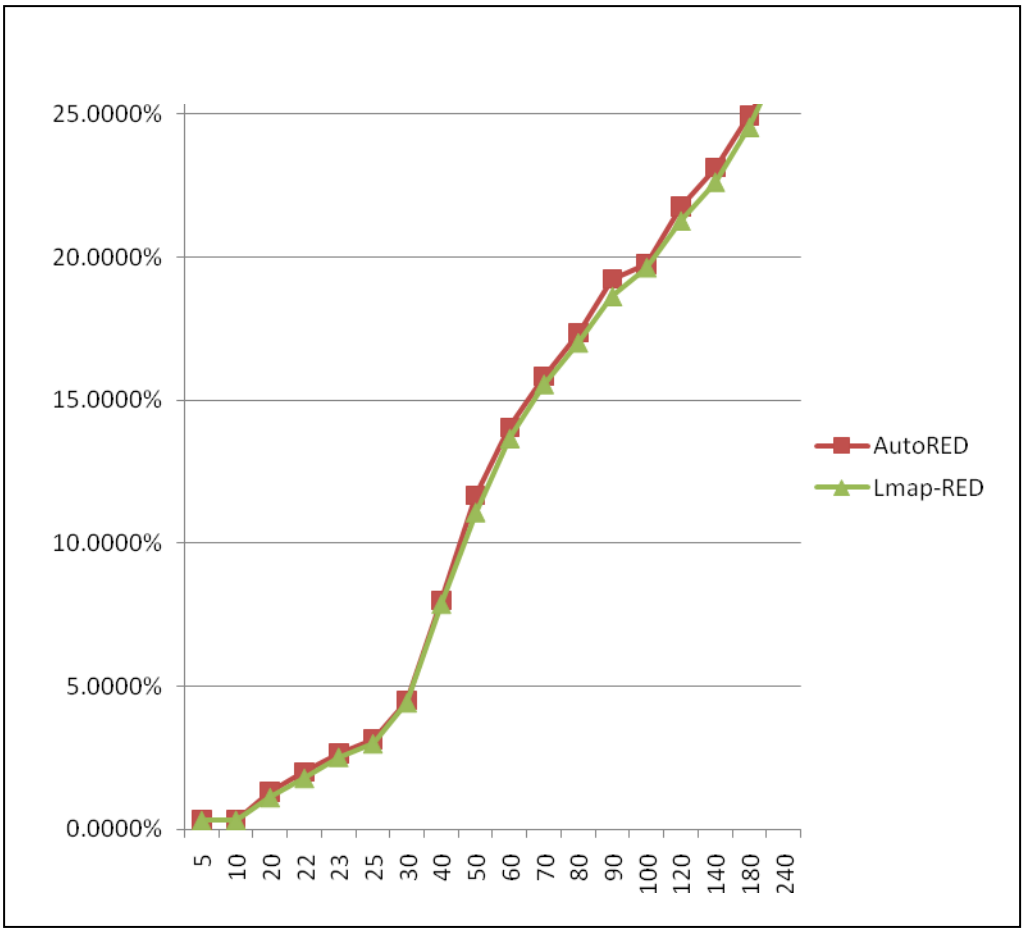

Figure 19. Comparison of packet loss rates on AutoRED and L_{map}-RED

These improvements are in addition to those that AutoRED has already made from the RED mechanism and thus their scale is very small. It should be noted that, although L_{map}-RED improves packet loss rates from AutoRED, the RED mechanism still has advantages in packet loss rates in less congested networks up to 25 TCP flows and 50, 70, and 80 in these simulations.

Table IV. Comparison data of link utilization and packet loss rate with *r* value

	AutoRED	**L_{map}-RED**		**AutoRED**	**L_{map}-RED**	
Number of TCP flows	*Link utilization*	*Link utilization*	*Best r value for Link utilization*	*Packet loss rate*	*Packet loss rate*	*Best r value for Packet loss rate*
5	98.1856%	98.5645%	2.5	0.3002%	0.2990%	2.5
10	99.4083%	99.5580%	3.2	0.3047%	0.3045%	1.333
20	99.7633%	99.8070%	3	1.2876%	1.0984%	1.333
22	99.6938%	99.7351%	1.5	1.9856%	1.7766%	1.333
23	99.5543%	99.5919%	3.5	2.6408%	2.5107%	1.333
25	99.8331%	99.8375%	3.58	3.1374%	2.9871%	3.58
30	99.7921%	99.8024%	2.5	4.4967%	4.4212%	3.2
40	99.7945%	99.7945%	3.2	8.0037%	7.8811%	3.5
50	99.8353%	99.8354%	3	11.6832%	11.0888%	3.2
60	99.8785%	99.8785%	3	14.0645%	13.6780%	3
70	99.8970%	99.8970%	3.2	15.8571%	15.5751%	3.2
80	99.9052%	99.9059%	1.5	17.3782%	17.0319%	3.2
90	99.8970%	99.8971%	3.2	19.2678%	18.6498%	2
100	99.8950%	99.8950%	3.5	19.7927%	19.6624%	3
120	99.9202%	99.9208%	2	21.8081%	21.2969%	3.2
140	99.9236%	99.9236%	3.58	23.1668%	22.6621%	3.58
180	99.9256%	99.9266%	3.58	24.9881%	24.5788%	2
240	99.9321%	99.9321%	2.5	26.9708%	27.1491%	2

Fig. 20 illustrates the improvements made by L_{map}-RED over the results of AutoRED on queuing delays. As discussed earlier, higher queuing delays are regarded better under 30 TCP flows in light of making most out of the network. Over 30, lesser queuing delays are considered advantageous. This is the pattern in which L_{map}-RED improves over AutoRED consistently except for 240 TCP flows.

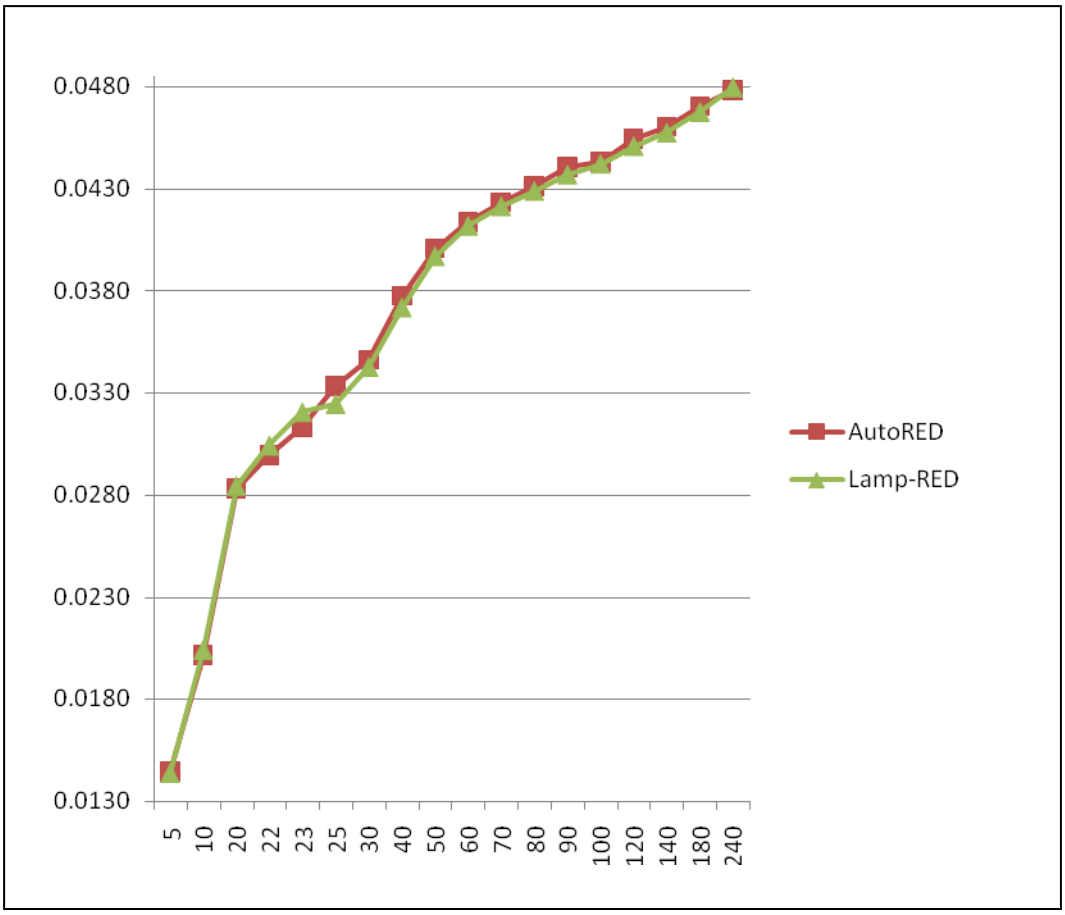

Figure 20. Comparison of queuing delays on AutoRED and L_{map}-RED

Table V presents the actual data including queue delays and overall average queue size along with the best *r* values that provide the improved results.

Looking at packet loss rates and overall average queue size, one thing to make a note of is that the simulation results on RED, AutoRED, and L_{map}-RED in the TCP-only environment all seem to exhibit a strong relation between packet loss rates and overall average queue size as can be observed in Fig. 12 and 13 for the comparison of RED and AutoRED, and in Fig. 19 and 20 for the comparison of AutoRED and L_{map}-RED. More specifically, pivoted at around 30 and 40 TCP flows where the overall average queue size

hits the maximum threshold, 30, higher overall average queue size goes together with lower packet loss rate up to 30 flows displaying an inverse relationship. But over 30 flows, overall average queue size appears to have a direct relationship to packet loss rate. The detailed data for this comparison are presented in Table VI.

Table V. Comparison data of queuing delay and overall average queue size with *r* value

	AutoRED		**L_{map}-RED**		
Number of TCP flows	*Overall average queue size*	*Queuing delay*	*Overall average queue size*	*Queuing delay*	*Best r value for queuing delay*
5	11.7040	0.0144	11.68137	0.0144	1.5
10	16.3520	0.0201	16.57656	0.0204	1.333
20	22.9815	0.0283	23.10391	0.0285	3.58
22	24.3068	0.0299	24.69771	0.0304	1.333
23	25.3937	0.0313	26.02212	0.0321	1.333
25	27.0373	0.0333	26.34913	0.0325	2
30	28.0922	0.0346	27.8203	0.0343	3.58
40	30.6343	0.0377	30.19001	0.0372	3.2
50	32.5137	0.0401	32.22483	0.0397	3.2
60	33.5703	0.0414	33.43114	0.0412	3
70	34.3478	0.0423	34.21268	0.0422	3.2
80	34.9815	0.0431	34.82078	0.0429	3.2
90	35.7544	0.0440	35.47095	0.0437	2
100	35.9658	0.0443	35.89763	0.0442	3
120	36.8767	0.0454	36.5995	0.0451	3.2
140	37.3555	0.0460	37.13542	0.0458	3.58
180	38.1792	0.0470	37.95568	0.0468	2
240	38.8190	0.0478	38.93321	0.0480	2.5

Table VI. Comparison of packet loss rate and overall average queue size

	RED		**AutoRED**		**L_{map}-RED**	
Number of TCP flows	*Packet loss rate*	*Overall average queue size*	*Packet loss rate*	*Overall average queue size*	*Packet loss rate*	*Overall average queue size*
5	0.2979%	12.8534	0.3002%	11.7040	0.2990%	11.681371
10	0.3020%	16.2785	0.3047%	16.3520	0.3045%	16.576555
20	0.3565%	24.0809	1.2876%	22.9815	1.0984%	23.103912
22	0.5972%	26.5759	1.9856%	24.3068	1.7766%	24.697705
23	1.9120%	27.6037	2.6408%	25.3937	2.5107%	26.022119
25	2.6873%	28.1569	3.1374%	27.0373	2.9871%	26.349133
30	4.9416%	28.8281	4.4967%	28.0922	4.4212%	27.820304
40	8.0835%	30.6688	8.0037%	30.6343	7.8811%	30.190014
50	10.9782%	32.3210	11.6832%	32.5137	11.0888%	32.22483
60	13.8491%	33.6447	14.0645%	33.5703	13.6780%	33.431137
70	15.3224%	34.1401	15.8571%	34.3478	15.5751%	34.212681
80	16.8809%	34.7464	17.3782%	34.9815	17.0319%	34.820782
90	19.1234%	35.5908	19.2678%	35.7544	18.6498%	35.470952
100	20.0174%	36.0479	19.7927%	35.9658	19.6624%	35.897634
120	21.7804%	36.7682	21.8081%	36.8767	21.2969%	36.599499
140	22.9056%	37.1838	23.1668%	37.3555	22.6621%	37.135416
180	25.5528%	38.3172	24.9881%	38.1792	24.5788%	37.955678
240	27.3717%	38.9788	26.9708%	38.8190	27.1491%	38.933207

As far as the degree of queue oscillation is concerned, the improvements made by L_{map}-RED with regards to statistical metrics and Seg time, are relatively larger as shown in the following three figures: Fig. 21 for the minimum and maximum queue size, Fig. 22 for standard deviation, and Fig. 23 for Seg-time. The detailed data with the best *r* values producing these results are shown in Table VII.

Figure 21. Comparison of minimum and maximum queue size

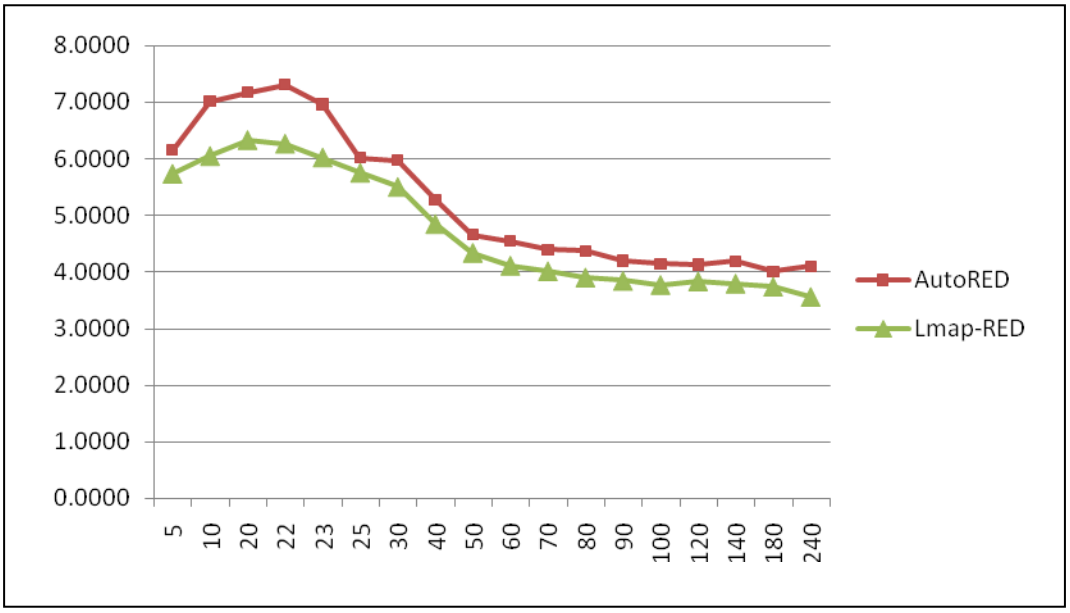

Figure 22. Comparison of standard deviation

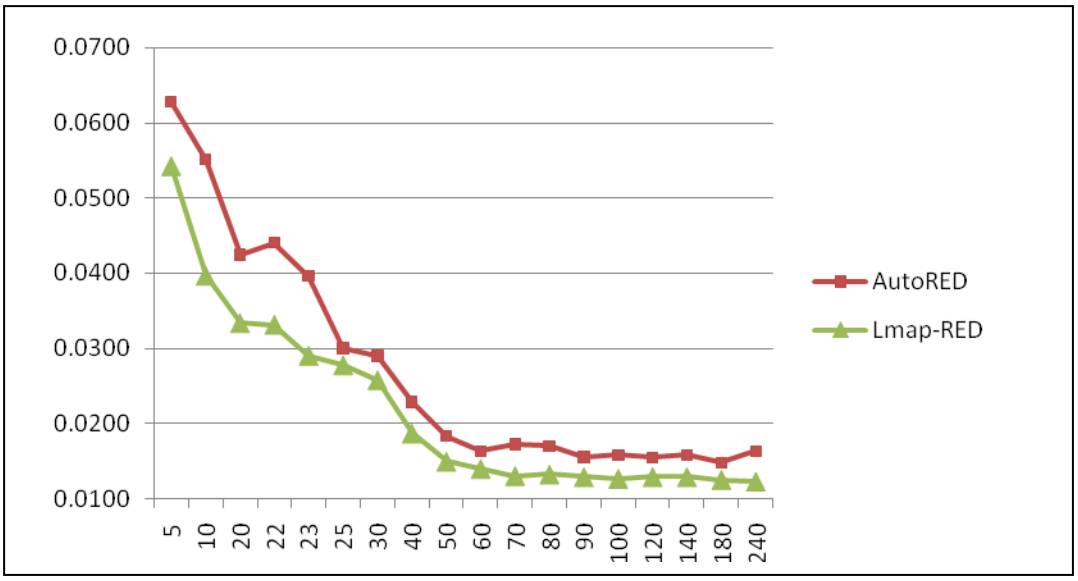

Figure 23. Comparison of Seg-time

As an example, 23 TCP flows is taken to show the visual representation of these improvements in standard deviation and Seg-time. It is apparent that there are no high peaks in the queue behavior of both AutoRED and L_{map}-RED. L_{map}-RED hits lows less frequently controlling the queue oscillation more evenly.

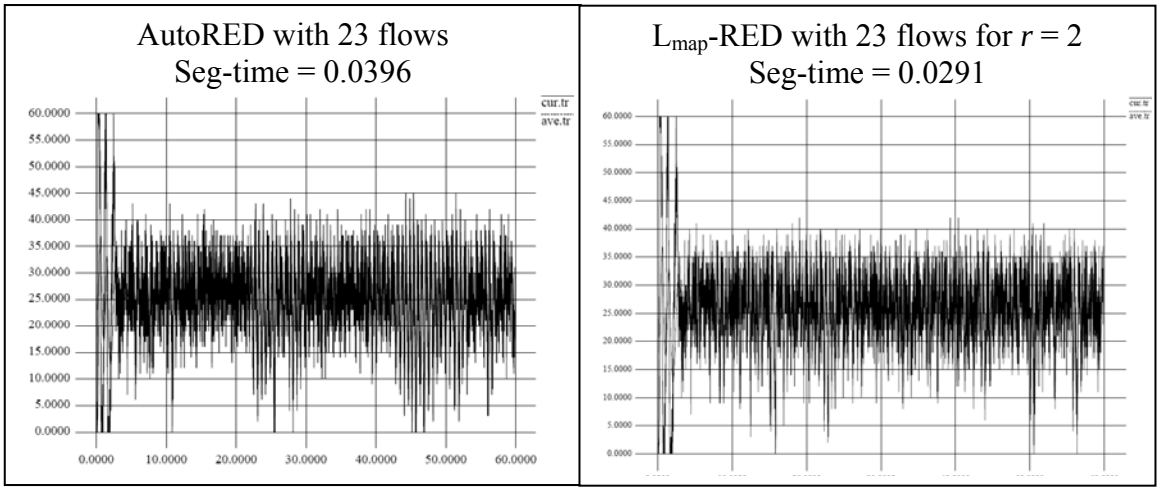

Figure 24. Visual comparison of queue behaviors on AutoRED and L_{map}-RED

Table VII. Comparison data of the minimum and maximum queue size, standard deviation, and Seg-time with best *r* value

	AutoRED			**L_{map}-RED**				**AutoRED**	**L_{map}-RED**	
Number of TCP flows	*mini mum*	*maxi mum*	*standard deviation*	*mini mum*	*maxi mum*	*standard deviation*	*Best r value for stdev*	*Seg-time*	*Seg-time*	*Best r value for Seg-time*
5	0	28	6.1598	0	27	5.7322	3.58	0.0628	0.0542	2
10	0	36	7.0119	0	36	6.0475	3.2	0.0552	0.0397	3.2
20	0	46	7.1774	0	42	6.3293	3.58	0.0425	0.0334	3.58
22	0	47	7.3068	0	43	6.2666	3.5	0.0440	0.0332	3.5
23	0	45	6.9629	0	42	6.0198	2	0.0396	0.0291	2
25	0	45	6.0229	5	44	5.7524	3.58	0.0301	0.0278	3.58
30	0	45	5.9705	3	43	5.5004	2.5	0.0290	0.0258	3.58
40	9	47	5.2796	10	46	4.8461	2.5	0.0229	0.0188	2.5
50	13	51	4.6592	14	46	4.3357	3	0.0183	0.0150	3
60	15	49	4.5476	16	49	4.1095	3.5	0.0163	0.0140	3
70	17	51	4.3933	18	49	4.0114	2.5	0.0173	0.0131	2.5
80	17	50	4.3792	19	49	3.8986	2.5	0.0170	0.0133	2.5
90	20	51	4.2032	20	50	3.8392	3.2	0.0156	0.0130	3.2
100	20	53	4.1458	21	50	3.7691	3.58	0.0158	0.0127	3.58
120	20	52	4.1361	18	49	3.8322	3.2	0.0155	0.0130	3.2
140	23	51	4.1979	20	51	3.7911	2.5	0.0158	0.0130	3.58
180	23	52	4.0144	23	51	3.7367	3	0.0149	0.0126	3.2
240	22	52	4.0947	23	53	3.5547	3.58	0.0163	0.0124	3.58

Tables IV, V, and VII contain *r* value coulmns that produce the best possible results for each metric per each simulation. An *r* value producing the best value in one metric category also produces in another or more. From this, relationships between categories are found in that there are two categories that share the same *r* values more often than others. First of all, the obvious relationship as mentioned in the comparison of

the results of RED and AutoRED experiments is the one between standard deviation and Seg-time. As illustrated in Fig. 25, many dots are overlapped meaning the same *r* value produces the best results for both standard deviation and Seg-time. Since both are representations of queue oscillation, this result is not surprising.

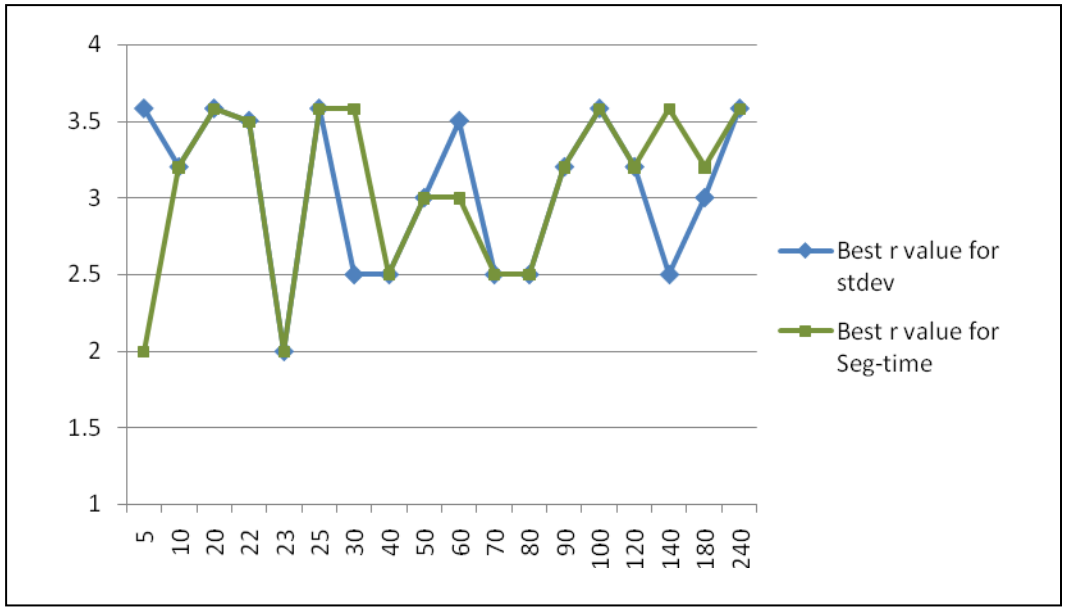

Figure 25. Best *r* values for standard deviation and Seg-time

The second relationship is between packet loss rate and overall average queue size (or queuing delay) displayed in Fig. 26. In highly congested networks from 50 flows to 180, an *r* value that provides the best packet loss rate in one simulation generates the best queuing delay as well.

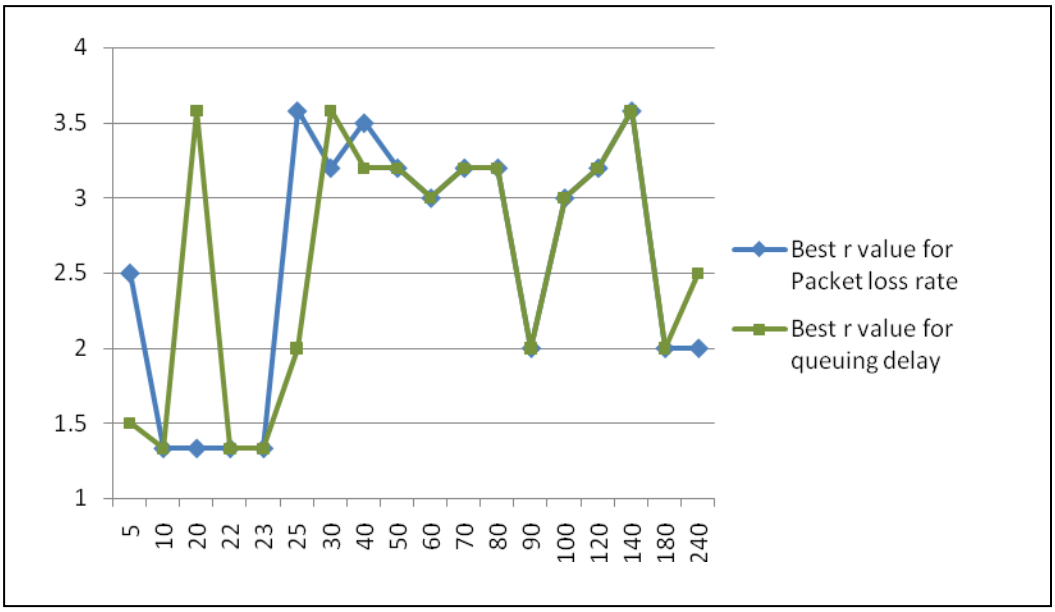

Figure 26. Best *r* values for packet loss rate and queuing delay

This relationship has been speculated earlier but looking at the effect of *r* values on otherwise the same simulation helps identifying the relationship more closely. What is seen from the illustrations of packet loss rates and overall average queue size over eight *r* values is that up to 30 flows an inverse relationship is observed while over 30 a direct relationship is clearly shown. Fig. 27 demonstrates this by presenting plots of packet loss rates on the left and overall average queue size on the right varied by eight *r* values for the number of flows – 5, 25, 40, 90, and 240.

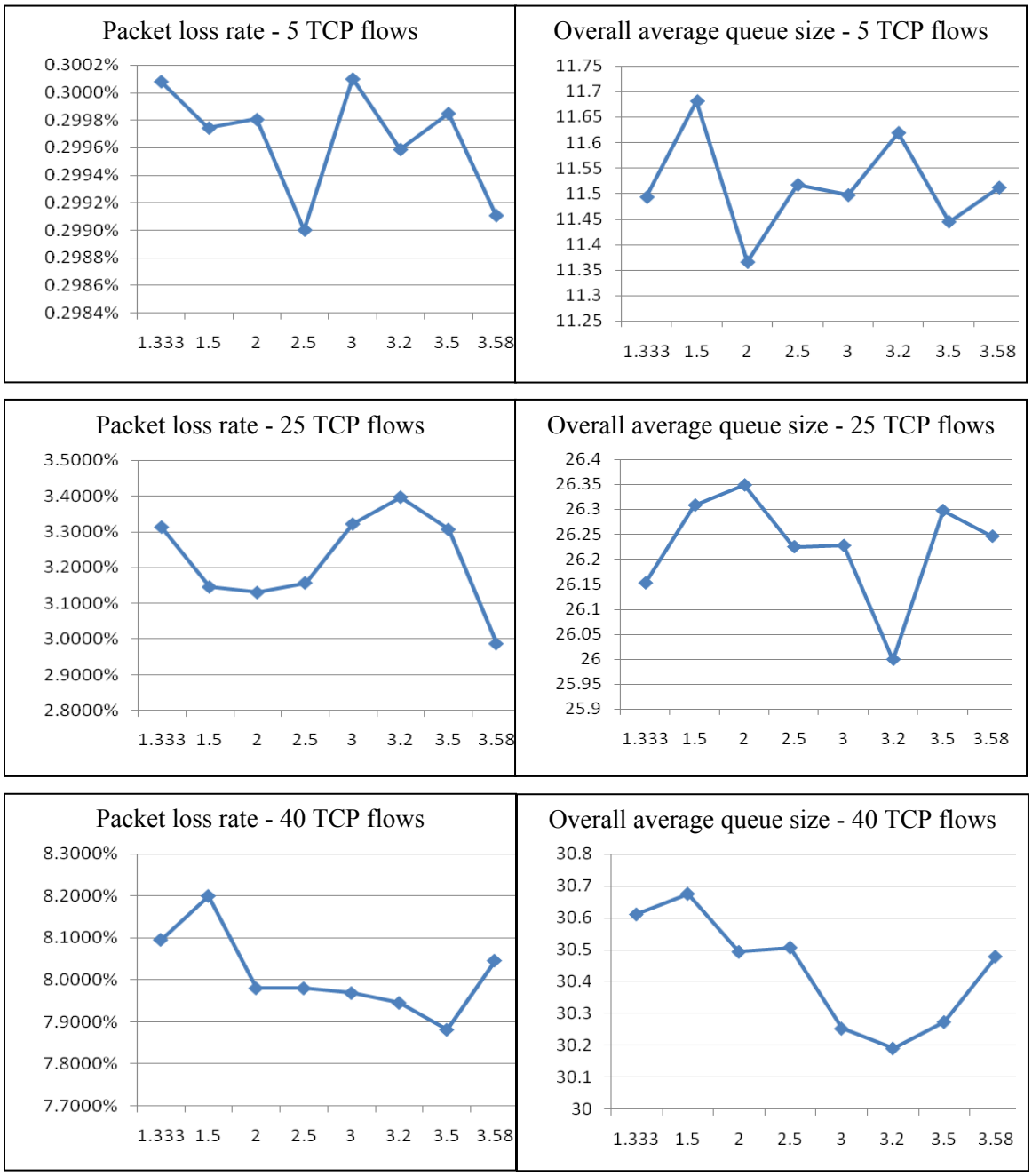

Figure 27. Relationship of packet loss rate and overall average queue size

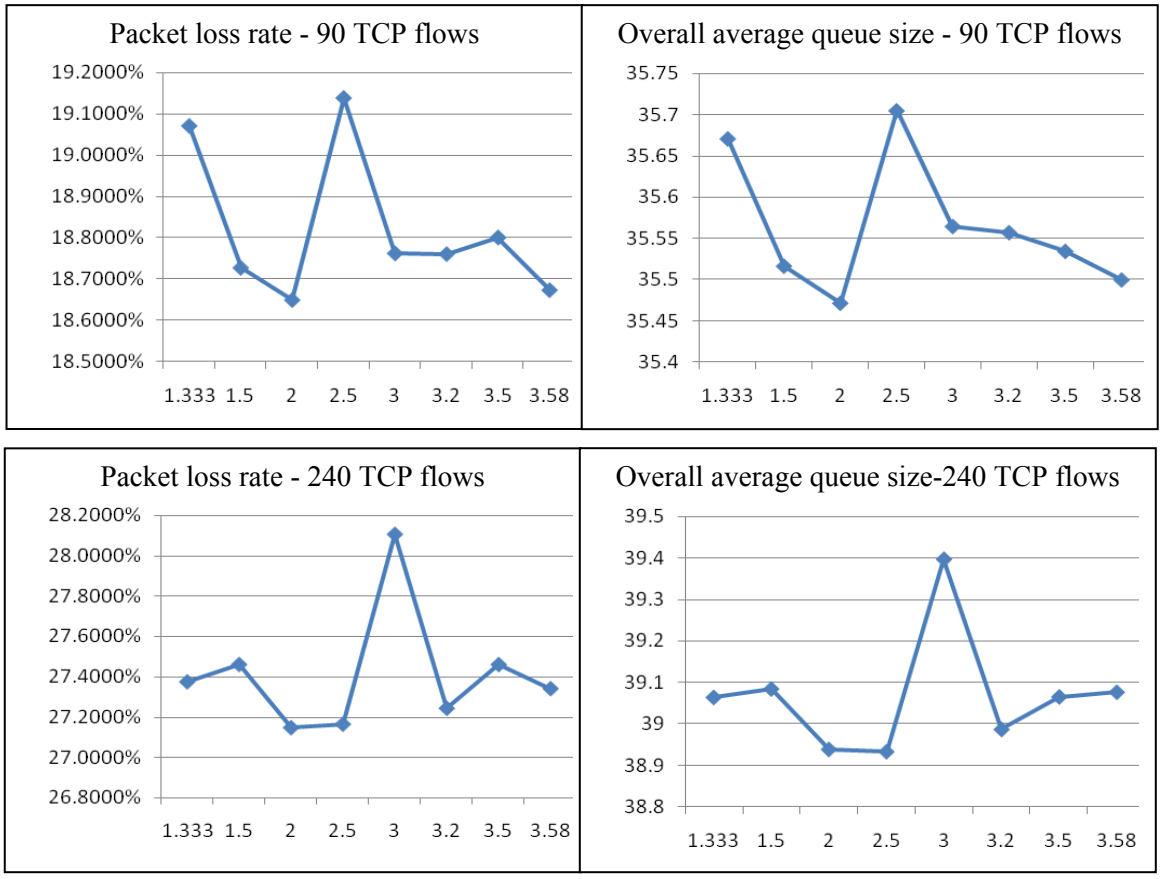

Figure 27. Relationship of packet loss rate and overall average queue size - continued

This concludes the description of experiments done in TCP-only environments and their findings.

TCP and UDP Combined Environment

In experimenting behaviors of RED, AutoRED, and L_{map}-RED in a mixed network environment of TCP and UDP flows, the number of UDP flows is increased from 5 to 50 while the number of TCP flows is kept at 10 as the 10 TCP flows have exhibited higher performance metric values for RED in the TCP-only environment. UDP flows are injected into the network at 9.5 seconds and all measurements for both network

performance and degree of queue oscillation, start from 10 seconds giving 0.5 seconds for filtering any transient behaviors. TCP packet size is 1,500 bytes whereas UDP packet size is 1,000 bytes because NS-2 breaks UDP packets larger than 1,000 bytes. Broken packets are undesirable because they will occupy two packet spaces in the queue. 10 TCP flows with 15 UDP flows, thus, can be translated to about an equal share of the bandwidth at the bottleneck. Additionally, the interval of CBR over UDP used in these simulations is to set 40 packets per second or at 3.2Mbps. The full data of L_{map}-RED with eight *r* values per each number of UDP flows can be found in Appendix A. 2.

Overall, AutoRED and L_{map}-RED exhibit better throughput and less packet loss rate than RED as shown in Fig. 28 and Fig. 29 respectively. All accounts of throughputs on AutoRED are equal to or greater than those of RED and L_{map}-RED shows further improvements from AutoRED even in a larger scale than that of the TCP-only environment. Packet loss rates on AutoRED show small improvements over those on RED in 20, 25, and 50 flows, but in fact AutoRED exhibits poor packet loss rates worse than RED in the rest of flows. In contrast, it is the L_{map}-RED that displays improvements in packet loss rates in all simulations except for 5 UDP flows. The noticeable dip in link utilization on 15 and 20 UDP flows on RED appears to be the same pattern exhibited in the TCP-only environment. The detailed data for the comparison of link utilization, packet loss rate, and overall average queue size can be found in Table VIII which helps recognizing the improvements made in a small scale not so apparent in the figures.

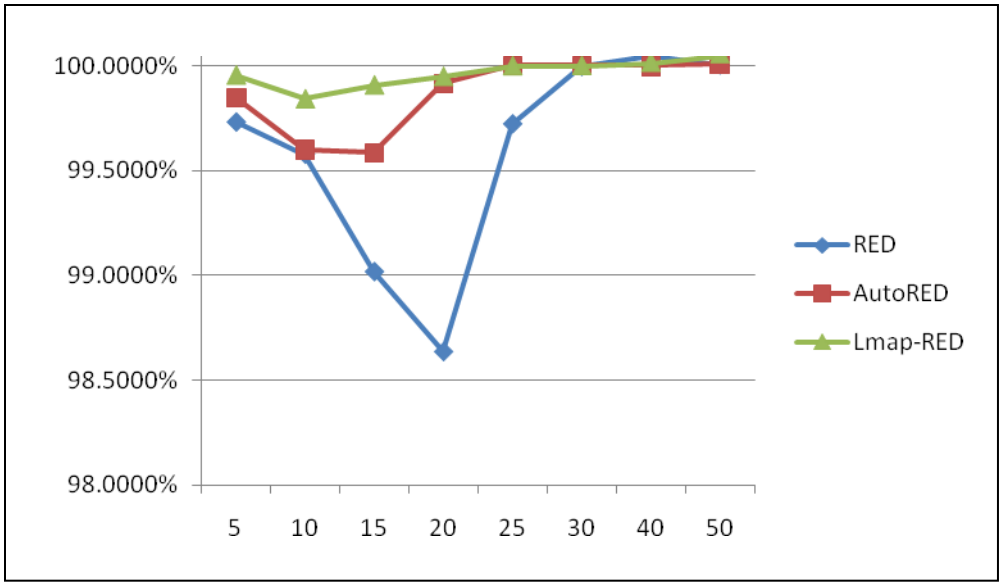

Figure 28. Comparison of link utilization

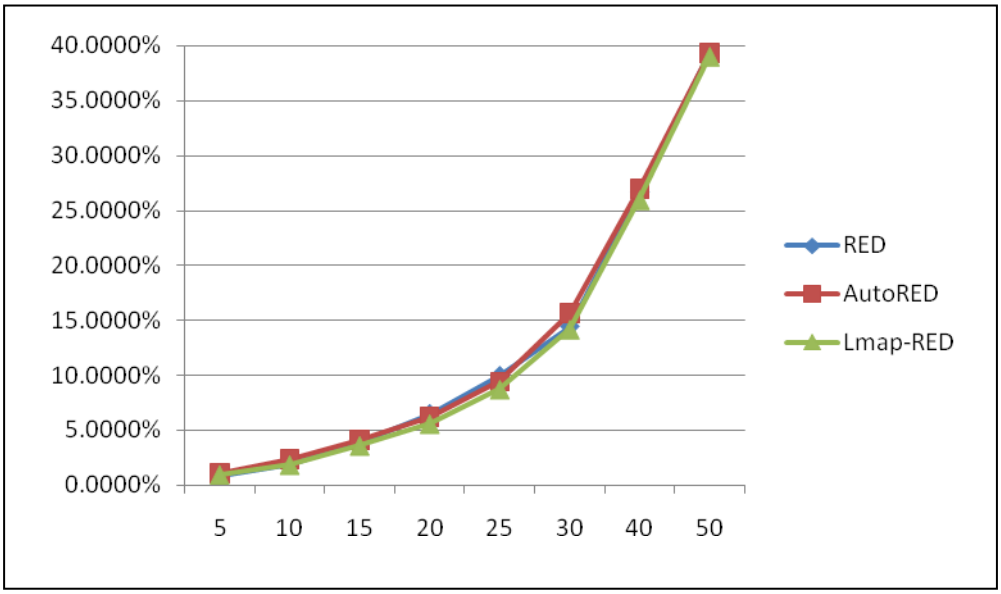

Figure 29. Comparison of packet loss rate

Table VIII. Comparison data of link utilization, packet loss rate, and overall average queue size with *r* value

	RED			**AutoRED**			**Lmap-RED**					
Number of UDP flows	*Link utilization*	*Packet loss rate*	*Overall average queue size*	*Link utilization*	*Packet loss rate*	*Overall average queue size*	*Best r value for Link utilization*	*Link utilization*	*Best r value for Packet loss rate*	*Packet loss rate*	*Best r value for average queue size*	*Overall average queue size*
5	99.7341%	0.8748%	19.2620	99.8500%	1.0165%	18.9676	2.5	99.9574%	3.58	0.9146%	2	18.8889
10	99.5798%	1.9561%	22.5489	99.5994%	2.2931%	20.7895	2.5	99.8435%	1.333	1.8156%	3	21.3694
15	99.0188%	3.9025%	25.6297	99.5853%	4.0941%	23.1975	3	99.9096%	2.5	3.5691%	3.5	23.5992
20	98.6361%	6.4394%	27.0872	99.9174%	6.1532%	26.4205	2	99.9505%	2.5	5.5439%	2	26.5165
25	99.7256%	9.9146%	29.0824	100.0048%	9.3852%	30.0423	2.5	100.0019%	3.2	8.6823%	2.5	30.1761
30	99.9998%	14.4148%	33.3497	100.0024%	15.5870%	34.0826	3	100.0030%	3.58	14.1431%	3.58	33.3799
40	100.0502%	26.6715%	39.1618	99.9999%	26.9060%	39.2197	1.333	100.0181%	2.5	25.9403%	1.333	38.8307
50	100.0072%	39.3144%	43.7521	100.0079%	39.2869%	43.9609	1.5	100.0596%	1.5	39.0250%	1.333	43.9722

One thing to note is that some link utilization numbers are over 100% as in Table VIII. Throughput is measured by the number of dequeued packets from the node A with the AQM queue and it can be speculated that some packets have been already in the pipe when the measurement starts at 10 seconds. Furthermore, throughput of TCP flows and that of UDP flows are calculated separately and added together for the total throughput and their start and end time is slightly different. This may have attributed to that as well. It should be safe to assume the link utilization to be nearly 100%.

Again, RED exhibits favorable overall average queue size in that, as is in the TCP-only environment, the number close to the maximum threshold, 30, is considered better. For example, on 25 UDP flows, the overall average queue size 29.0824 on RED is regarded as better than 30.0423 on AutoRED and 30.176 on L_{map}-RED. The region that RED displays better overall average queue size is again in less congested networks. This result is parallel to that of the TCP-only environment as shown in Fig. 30.

It should be noted that in the TCP and UDP combined environment, queuing delays cannot be calculated in the same method presented earlier because TCP and UDP packet sizes are different and the queue capacity in these simulations is in the unit of packets instead of bytes. However, the overall average queue size can replace queuing delay in observing its effect and relationship with other metrics because they have a direct linear relationship.

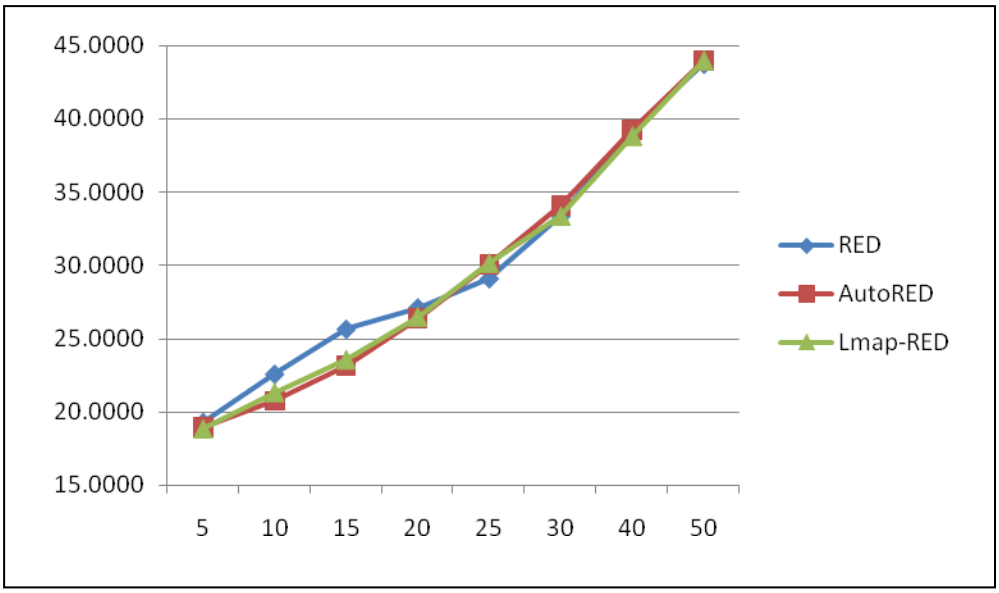

Figure 30. Comparison of overall average queue size

In measuring the degree of queue oscillation, AutoRED and L_{map}-RED show favorable values for the minimum and maximum queue size, standard deviation, and Seg-time. L_{map}-RED shows improvements in all accounts over the results of AutoRED that exhibit better outcomes than those of RED. Fig. 31, Fig. 32, and Fig. 33 illustrate the comparison of the minimum and maximum queue size, standard deviation, and Seg-time respectively. It should be noted that there is a single set of *r* values chosen for both link utilization and packet loss rates on UDP flows because the number of enqueued packets are the same per given number of flows in CBR applications. The actual data for these metrics and the best *r* values are found in Table IX.

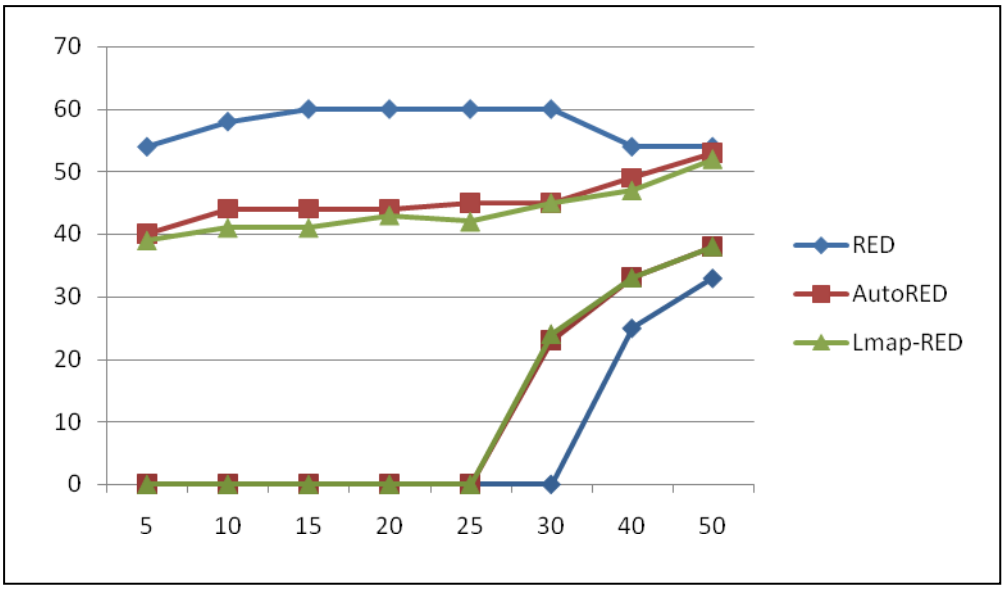

Figure 31. Comparison of minimum and maximum queue size

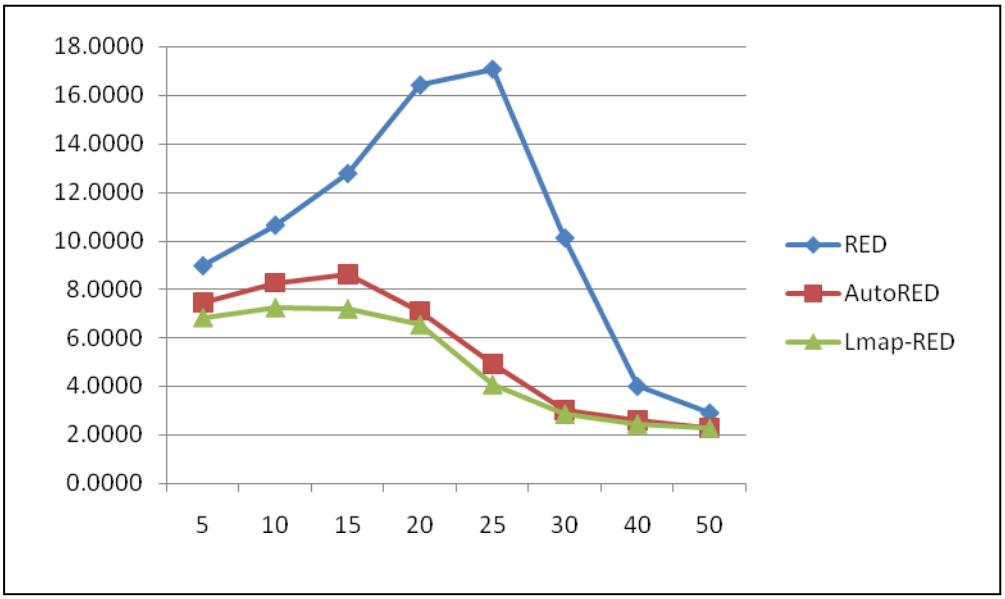

Figure 32. Comparison of standard deviation

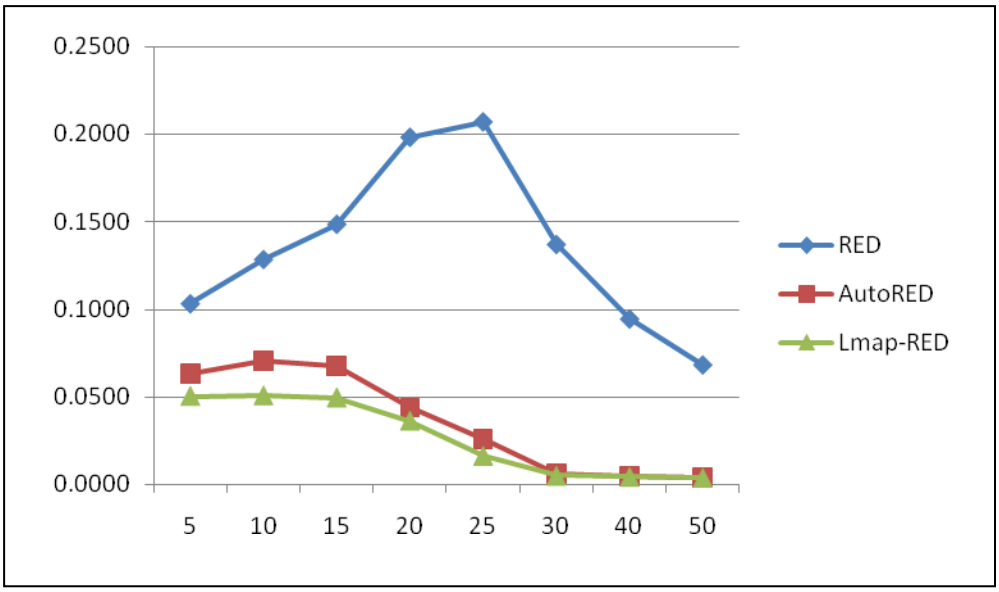

Figure 33. Comparison of Seg-time

Table IX. Comparison data of the minimum and maximum queue size, standard deviation, and Seg-time with *r* value

	RED				**AutoRED**				**Lmap-RED**					
Number of UDP flows	*mini mum*	*maxi mum*	*standard deviation*	*Seg-time*	*mini mum*	*maxi mum*	*standard deviation*	*Seg-time*	*Best r value for stdev*	*mini mum*	*maxi mum*	*standard deviation*	*Best r value for Seg-time*	*Seg-time*
5	0	54	8.9785	0.1031	0	40	7.4206	0.0633	2.5	0	39	6.82071	2.5	0.0503
10	0	58	10.6440	0.1283	0	44	8.2435	0.0707	3	0	41	7.22894	3	0.0506
15	0	60	12.7720	0.1484	0	44	8.5980	0.0677	3.5	0	41	7.17993	3.5	0.0494
20	0	60	16.4059	0.1980	0	44	7.0956	0.0439	3.2	0	43	6.52776	3.2	0.0362
25	0	60	17.0618	0.2067	0	45	4.9042	0.0256	3	0	42	4.0769	3	0.0162
30	0	60	10.1227	0.1368	23	45	3.0265	0.0060	2.5	24	45	2.87104	2	0.0048
40	25	54	4.0161	0.0944	33	49	2.5994	0.0046	3	33	47	2.46506	3.2	0.0043
50	33	54	2.9205	0.0682	38	53	2.2848	0.0036	1.333	38	52	2.28632	1.333	0.0036

Noticeably in Fig. 32 and Fig. 33, the simulations on RED exhibit higher degrees of queue oscillation especially starting at 15 and 20 flows peaking at 25. This is also the same region the link utilization is poor on RED. However, it drops from 30 UDP. The visual comparison of queue behaviors on RED with those of L_{map}-RED is presented in Fig. 34.

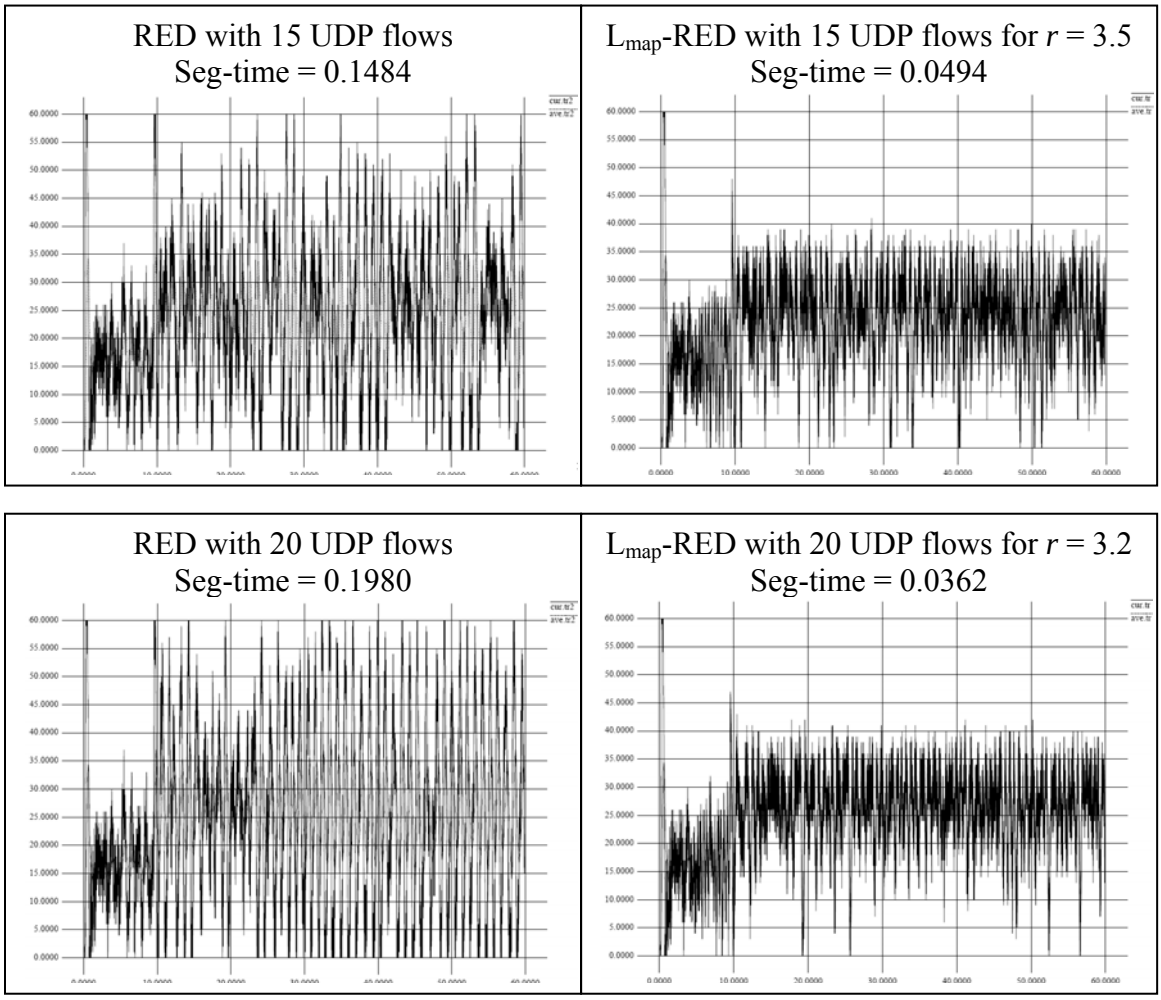

Figure 34. Visual comparison of queue behaviors on RED and L_{map}-RED

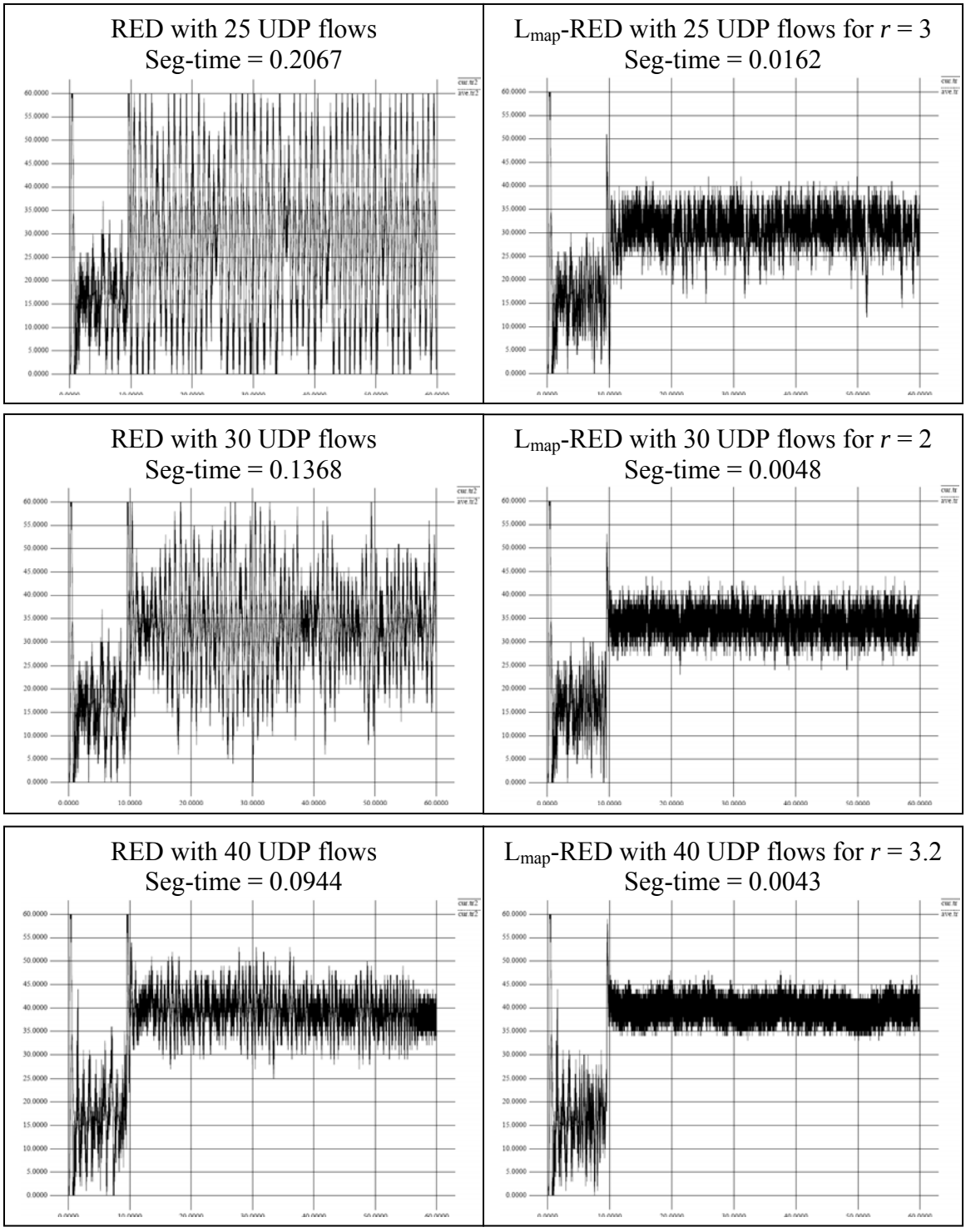

Figure 34. Visual comparison of queue behaviors on RED and L_{map}-RED - continued

Note that the injection of UDP traffic starts at 9.5 seconds, therefore, the time between 0 and 9.5 seconds only consists of 10 TCP flows. In the case of L_{map}-RED, as in the TCP-only environment, 0 to 5 seconds is on RED and then the AQM mechanism is switched to L_{map}-RED. The increase and decrease of the chaotic queue oscillation on RED is shown clearly.

As in the TCP-only environment, the relationship between Seg-time and standard deviation is observed. The plot of the best *r* values for standard deviation and Seg-time affirms this as shown in Fig. 35.

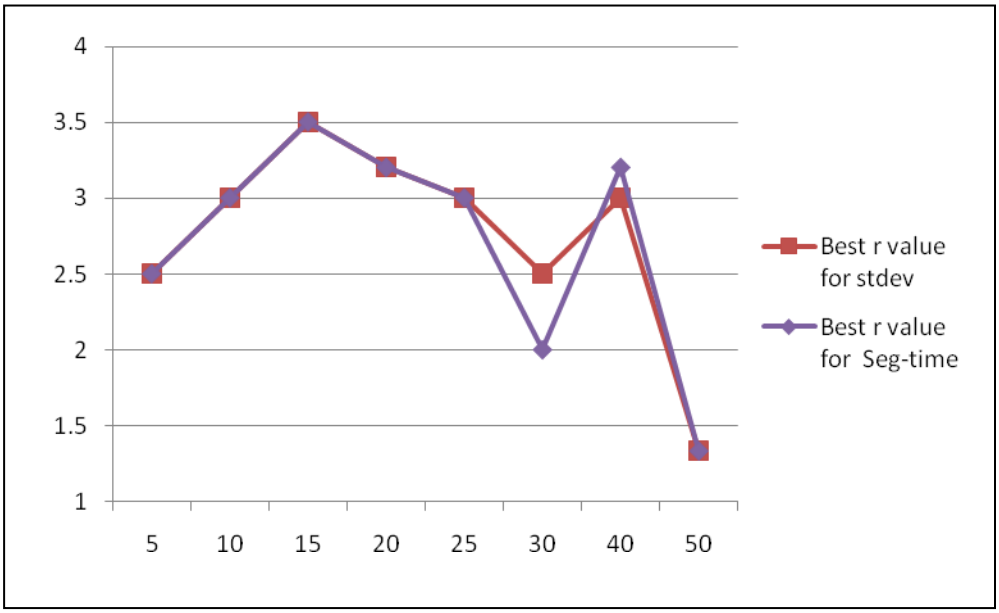

Figure 35. Comparison of best *r* values for standard deviation and Seg-time

Compared with the TCP-only environment, the relationship between packet loss rate and overall average queue size is not found to be as strong. They do not share the same *r* value as they do in the TCP-only environment. And the inverse relationship

observed in the simulations of which overall average queue size is less than the maximum threshold is not apparent.

Fairness Analysis

How fair are RED, AutoRED, and L_{map}-RED in treating TCP and UDP flows when they are combined? Two approaches have been made in an attempt to answer this. One way is to look at TCP and UDP traffics independently as if they are not combined. Each type of traffic is measured in terms of link utilization and packet loss rate separately from the other. This helps to see how the performance of TCP and UDP traffics changes as the number of UDP flows increases. Moreover, it allows the comparison of TCP and UDP performances by their type. The other method is to explore the ratio in link utilization and packet loss rate of TCP traffics versus UDP traffics with regards to those of the combined traffics. This is to investigate if any mechanism favors one type of traffics over the other.

Since the values of aggregate link utilization are very close to 100%, the independent value of TCP and UDP traffics is approximately equivalent to their ratio. Minor differences come from the fact that some of the aggregate link utilization is less than 100% and that the TCP payload of 1,500 bytes without the 40 bytes of header has been used in calculating throughput. Therefore, only the independent values of link utilization of TCP traffics and UDP traffics are shown in Fig. 36 and Fig. 37 respectively. Table X presents the detailed data. The ratio of link utilization between TCP and UDP traffics is contained in Table XI.

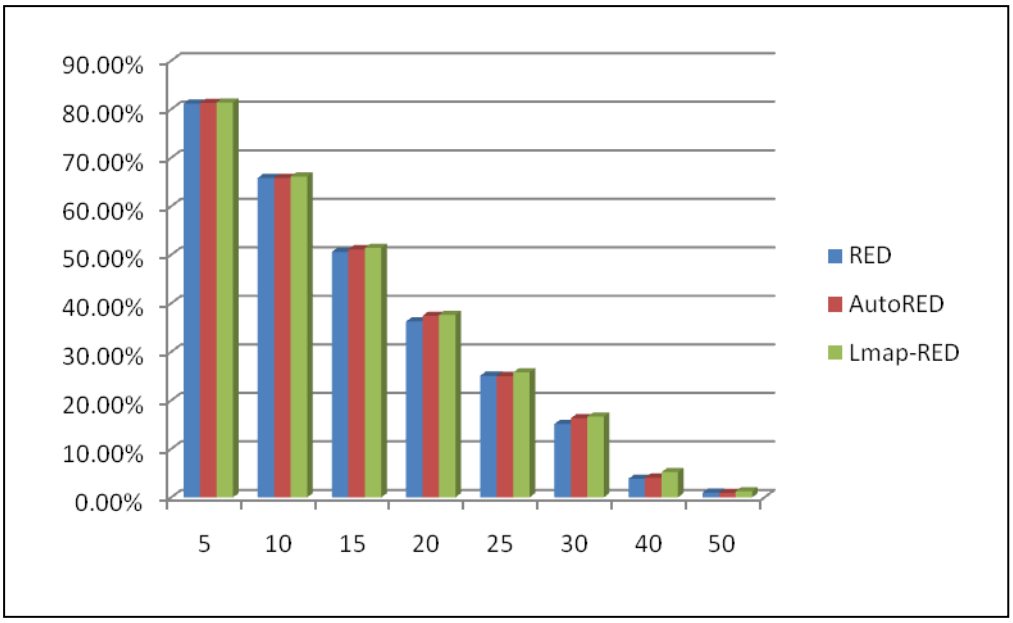

Figure 36. Comparison of TCP composition in link utilization

In these figures, the performance of L_{map}-RED shows better link utilization in all simulations for both TCP and UDP traffics. It increases overall throughput without favoring either TCP traffics or UDP traffics. AutoRED exhibits very slight improvements over RED in UDP link utilization from 10 to 25 flows.

No apparent differences are found in treating TCP and UDP traffics on all three schemes. One minor point is that RED has less TCP traffics than UDP traffics on 30 and 40 flows compared with AutoRED and L_{map}-RED.

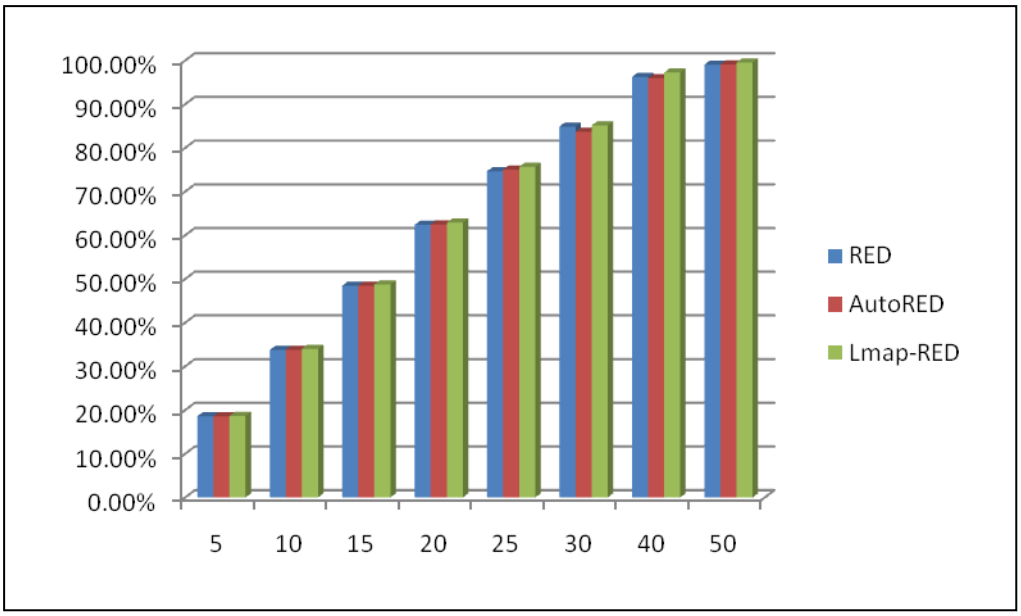

Figure 37. Comparison of UDP composition in link utilization

Table X. Comparison data of independent link utilization of TCP and UDP with r value

		RED		**Auto-RED**		**Lmap-RED**			
		Link utilization		*Link utilization*		*Link utilization*			
No. of TCP flows	*No. of UDP flows*	*TCP*	*UDP*	*TCP*	*UDP*	*Best r value for TCP*	*TCP*	*Best r value for UDP*	*UDP*
10	5	81.1597%	18.5744%	81.2820%	18.5680%	2.5	81.3718%	3.58	18.6512%
10	10	65.8390%	33.7408%	65.8106%	33.7888%	3	66.1200%	1.333	34.0016%
10	15	50.5772%	48.4416%	51.1133%	48.4720%	3.2	51.4227%	2.5	48.7360%
10	20	36.2553%	62.3808%	37.4118%	62.5056%	2	37.6065%	2.5	62.8928%
10	25	25.0728%	74.6528%	24.9984%	75.0064%	2	25.7445%	3.2	75.6704%
10	30	15.1086%	84.8912%	16.3032%	83.6992%	2.5	16.6321%	3.58	85.1456%
10	40	3.8374%	96.2128%	4.0591%	95.9408%	3.5	5.1868%	2.5	97.2304%
10	50	0.9704%	99.0368%	0.8959%	99.1120%	3.2	1.2230%	1.5	99.5488%

Table XI. Comparison data of ratio between TCP and UDP link utilization with *r* value

		RED		**Auto-RED**		**Lmap-RED**			
		Link utilization		*Link utilization*		*Link utilization*			
No. of TCP flows	*No. of UDP flows*	*TCP*	*UDP*	*TCP*	*UDP*	*Best r value for TCP*	*TCP*	*Best r value for UDP*	*UDP*
10	5	80.9738%	19.0262%	81.4037%	18.5963%	2.5	81.0047%	3.58	19.0734%
10	10	65.5245%	34.4755%	65.4822%	34.5178%	3	65.6345%	1.333	34.7611%
10	15	50.4188%	49.5812%	50.6672%	49.3328%	3.2	50.8748%	2.5	49.4613%
10	20	36.1423%	63.8577%	36.8278%	63.1722%	2	37.0090%	2.5	63.5461%
10	25	24.6485%	75.3515%	24.5026%	75.4974%	2	25.2440%	3.2	76.1517%
10	30	14.7727%	85.2273%	15.9458%	84.0542%	2.5	16.2700%	3.58	85.4745%
10	40	3.6924%	96.3076%	3.9570%	96.0430%	3.5	5.0588%	2.5	97.3005%
10	50	0.9386%	99.0614%	0.8642%	99.1358%	3.2	1.1740%	1.5	99.5607%

In comparing packet loss rates, the TCP payload size of 1,500 bytes has been used for computing the ratio as in link utilization. However, unlike the computation of link utilization in which throughputs of TCP and UDP in Mbps are put together and divided by the bandwidth of the link, the packet loss rate of the combined traffics is not the sum of each packet loss rate of TCP and UDP flows. Each type calculates packet loss rate as if the other type of traffic does not exist. Hence, the packet loss rate for TCP flows is the percentage of the number of dropped TCP packets out of the number of enqueued TCP packets and so the UDP packet loss rate is the dropped UDP packets out of enqueued UDP packets. Thus these rates are not the composition of the combined network as is the case with link utilization. First the packet loss rates of each type of flows are shown in Fig. 38 for TCP and Fig. 39 for UDP. The detailed data can be found in Table XII.

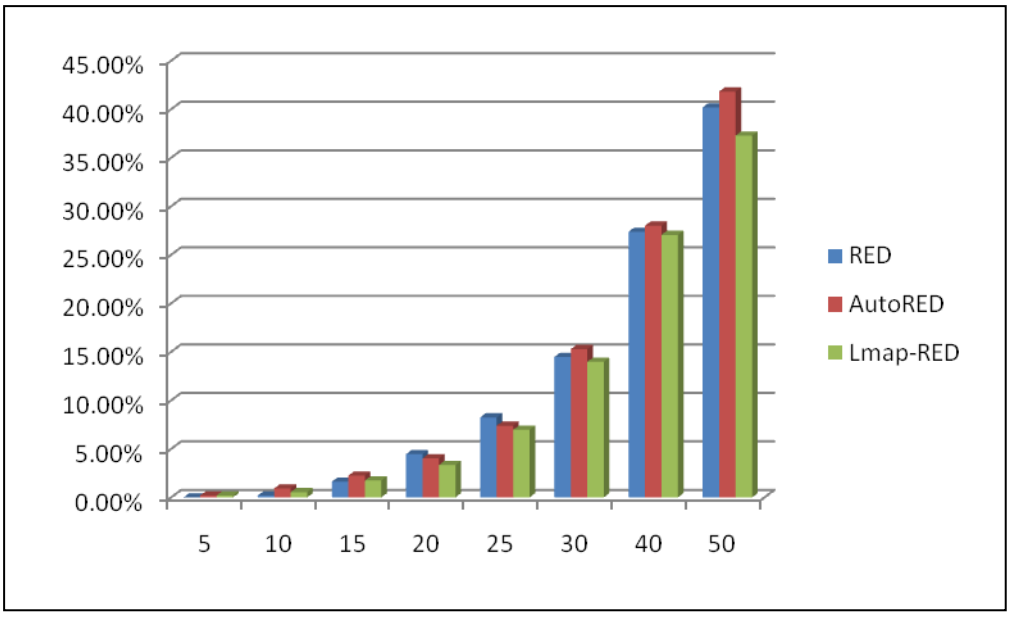

Figure 38. Comparison of independent TCP packet loss rates

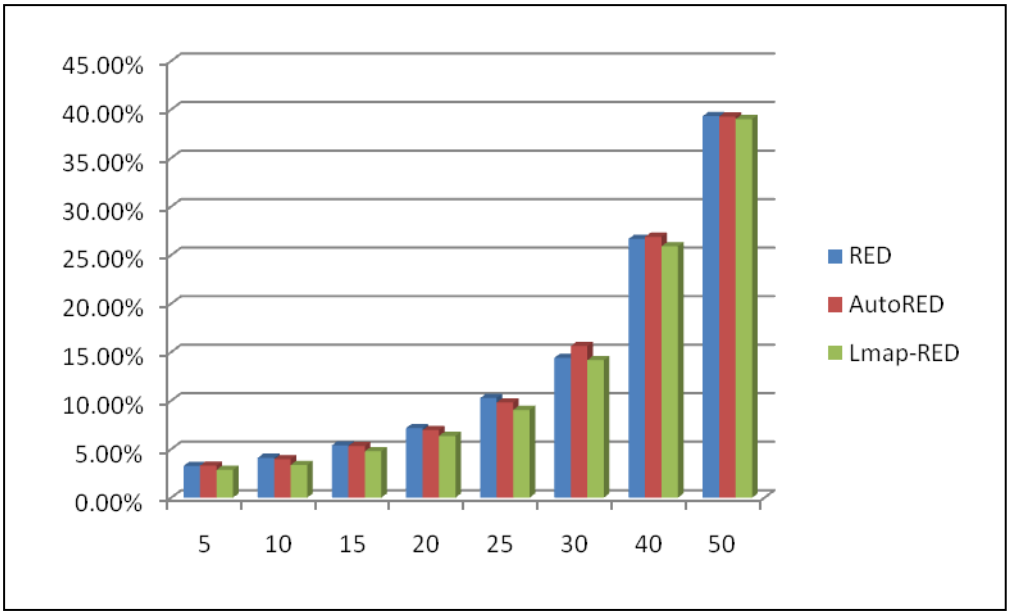

Figure 39. Comparison of independent UDP packet loss rates

L_{map}-RED exhibits favorable packet loss rates in all accounts except for TCP packet loss rates from 5 to 15 flows where RED displays better packet loss rates. The packet loss rate on L_{map}-RED on 5 flows is the worst of the three mechanisms. RED shows poor packet loss rates on 20 and 25 flows for both TCP and UDP while AutoRED displays poor packet loss rates on 30 and 40 flows for both TCP and UDP traffics. In addition, the UDP packet loss rates are almost identical in 50 UDP flows on all three schemes while the TCP packet loss rates rather differ.

Table XII. Comparison data of independent packet loss rate of TCP and UDP with *r* value

		RED		**Auto-RED**		**Lmap-RED**			
		Packet loss rate		*Packet loss rate*		*Packet loss rate*			
No. of TCP flows	*No. of UDP flows*	*TCP*	*UDP*	*TCP*	*UDP*	*Best r value for TCP*	*TCP*	*Best r value for UDP*	*UDP*
10	5	0.0030%	3.2667%	0.1876%	3.3000%	2.5	0.1904%	3.58	2.8583%
10	10	0.1905%	4.1045%	0.9305%	3.9636%	1.333	0.5385%	1.333	3.3682%
10	15	1.6248%	5.3875%	2.2476%	5.3188%	2.5	1.7375%	2.5	4.7781%
10	20	4.4489%	7.1690%	4.0192%	6.9571%	1.333	3.3434%	2.5	6.3571%
10	25	8.2424%	10.2712%	7.3807%	9.8077%	3.2	6.9557%	3.2	9.0346%
10	30	14.4730%	14.4081%	15.2792%	15.6258%	3.58	13.9678%	3.58	14.1629%
10	40	27.3629%	26.6537%	27.9965%	26.8756%	1.333	27.0433%	2.5	25.8927%
10	50	40.1840%	39.3088%	41.8152%	39.2716%	3.2	37.2751%	1.5	39.0029%

The ratio of the packet loss rates is based on the number of bytes accounted by dropped TCP or UDP packets out of the total number of bytes dropped in a simulation. Therefore, for example, two lost TCP packets in bytes are equivalent to three lost UDP

packets. The ratio of packet loss rates between TCP traffics and UDP traffics are shown in Fig. 40 and 41 for TCP and UDP respectively.

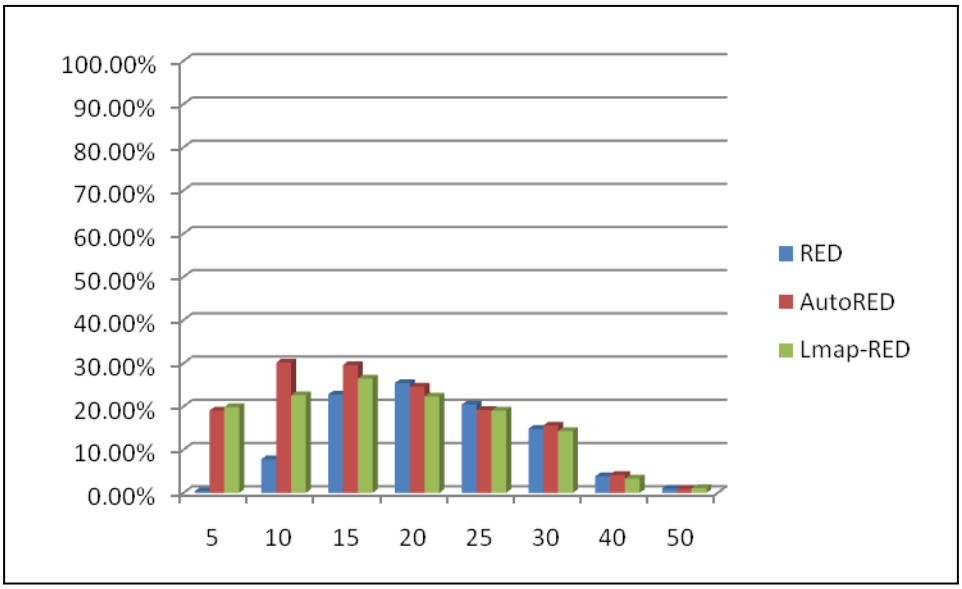

Figure 40. Comparison of the ratio of TCP packet loss rates

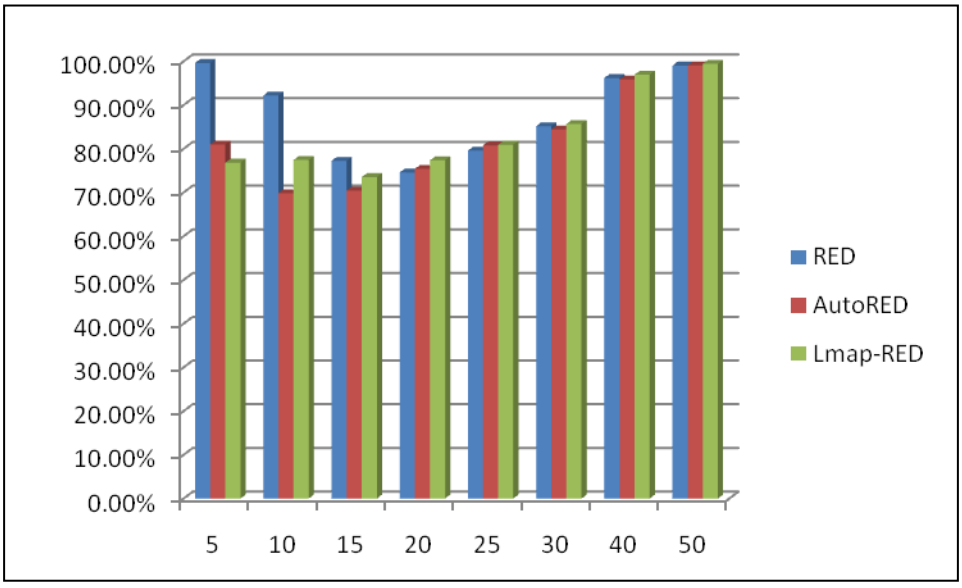

Figure 41. Comparison of the ratio of UDP packet loss rates

First of all, it is apparent that UDP packet loss is much greater than that of TCP in all number of UDP flows even when TCP traffics are prevailing. The ratio of UDP packet loss ranges from 70 % to over 99%. From 5 to 15 flows, the ratio of UDP packet loss decreases on all three schemes while the ratio of TCP packet loss increases even though the number of UDP flows increases. From 20 UDP flows, the ratio of UDP packet loss increases as the UDP flows increase.

No variation is found in each scheme's treatment on or above 20 UDP flows in highly congested networks. But the results from 5 to 15 UDP flows on RED stand out. They show much smaller ratio of TCP packet loss comparing with that of AutoRED and L_{map}-RED. Contrary to this, the ratio of TCP packet loss on AutoRED is higher than RED and L_{map}-RED in the same region of 5 to 15 flows. The complete data for the ratio of TCP and UDP traffics in packet loss rates are presented in Table XIII.

Table XIII. Comparison data of the ratio of TCP and UDP packet loss rates with *r* value

		RED		**Auto-RED**		**Lmap-RED**			
		Packet loss rate		*Packet loss rate*		*Packet loss rate*			
No. of TCP flows	*No. of UDP flows*	*TCP*	*UDP*	*TCP*	*UDP*	*Best r value for TCP*	*TCP*	*Best r value for UDP*	*UDP*
10	5	0.3812%	99.6188%	19.0184%	80.9816%	2.5	19.8738%	3.58	76.8197%
10	10	7.8101%	92.1899%	30.1562%	69.8438%	1.333	22.5705%	1.333	77.4295%
10	15	22.7772%	77.2228%	29.5967%	70.4033%	2.5	26.4727%	2.5	73.5273%
10	20	25.4426%	74.5574%	24.6130%	75.3870%	1.333	22.3381%	2.5	77.3577%
10	25	20.4261%	79.5739%	19.2207%	80.7793%	3.2	19.0698%	3.2	80.9302%
10	30	14.8427%	85.1573%	15.5914%	84.4086%	3.58	14.3275%	3.58	85.6725%
10	40	3.8219%	96.1781%	4.1743%	95.8257%	1.333	3.3179%	2.5	96.9520%
10	50	0.9707%	99.0293%	0.9569%	99.0431%	3.2	1.0691%	1.5	99.4227%

CHAPTER VI

DISCUSSIONS

Main findings of the experiments include the following points:

The comparison of RED, AutoRED, and L_{map}-RED in TCP-only environments presents the relationship between network performance and the chaotic queue oscillation as previous works have stated [1], [17], [19]. RED exhibits low link utilization especially in the region where the overall average queue size is near the maximum threshold of the RED mechanism. The region of lower throughput on RED which is from 23 flows to 80 flows is the same region where the chaotic queue oscillation is observed visibly and statistically with its queue size swinging from bottom to top and larger values of standard deviation and Seg-time. AutoRED and L_{map}-RED display improvements in link utilization in this region. However RED shows better link utilization and packet loss rates in less congested networks along with better queuing delays in that region. The best values of L_{map}-RED present slightly more favorable results where AutoRED has already made improvements.

In TCP-UDP combined environments, the best values of L_{map}-RED display improvements in all accounts except for the packet loss rate on 5 UDP flows, the least amount of traffics in these simulations. RED exhibits its strength in controlling overall average queue size better than L_{map}-RED and AutoRED. RED still displays the chaotic

queue oscillation until 30 and 40 UDP flow simulations where UDP flows dominate. All three schemes seem to treat TCP and UDP traffics without favoring one over the other except for RED showing lower packet loss rates on TCP traffics in 5, 10, and 15flows. RED's chaotic queue oscillation is stabilized in both AutoRED and L_{map}-RED.

The newly proposed metric, Seg-time, is discussed for representing a type of delay caused by the queue oscillation. The comparison of RED and AutoRED in the TCP-only environment serves as a test bed in verifying Seg-time as a valid metric. The Seg-time measurement provides analogous results in that the region of higher Seg-time values on RED is the same region that exhibits the chaotic queue oscillation visibly and statistically. Particularly, Seg-time and standard deviation appear to exhibit a direct linear relationship as shown in Fig. 14 and Fig. 16.

One way of interpreting Seg-time values in terms of its effect on the QoS is this. In the bottleneck link with 10Mbps bandwidth, the transmission time of one TCP packet of 1,540 bytes is 0.001232 seconds, which is calculated by 1540×8 bits divided by 10^7. Then the largest Seg-time value in the simulations in the TCP-only environment, 0.1797 seconds, on 40 TCP flows on RED can be translated into at least 145 packets as opposed to 10 packets on the 240 flows on L_{map}-RED. This can mean that on average the queue may stay overloaded or underutilized in 145 packet-time. The comparison data of Seg-time in TCP packet in the TCP-only and the TCP and UDP combined environments are found in Table XIV and Table XV respectively. Since a UDP packet size is 1,000 bytes, the number of UDP packets in the same Seg-time is greater than that of TCP packets.

While measuring queuing delay is straightforward without consuming hardly any computational resources since it directly derives from the current queue size, measuring Seg-time takes multiple steps. Furthermore, unlike queuing delay which can be measured almost instantaneously as it happens, current Seg-time calculation is for measuring what has happened in a period of time since it requires the calculation of the overall average queue size and reference points first. Hence, the weakness of Seg-time includes the complexity of calculation with inevitable resource consumption and the fact that it can only be a "hindsight" view at least with the current algorithm.

Table XIV. Comparison data of Seg-time in TCP packet in TCP-only environment

	RED		**L_{map}-RED**	
No. of TCP flows	*Seg-time*	*Seg-time in packet*	*Seg-time*	*Seg-time in packet*
5	0.0651	52.84	0.0542	44.02
10	0.0533	43.23	0.0397	32.22
20	0.0474	38.49	0.0334	27.11
22	0.0477	38.69	0.0332	26.94
23	0.0909	73.78	0.0291	23.61
25	0.0902	73.20	0.0278	22.58
30	0.1777	144.22	0.0258	20.96
40	0.1797	145.83	0.0188	15.22
50	0.1750	142.03	0.0150	12.18
60	0.1687	136.96	0.0140	11.40
70	0.1657	134.47	0.0131	10.64
80	0.1711	138.87	0.0133	10.80
90	0.1541	125.09	0.0130	10.52
100	0.1430	116.05	0.0127	10.34
120	0.1228	99.69	0.0130	10.53
140	0.1177	95.57	0.0130	10.52
180	0.1015	82.35	0.0126	10.23
240	0.1020	82.77	0.0124	10.03

Table XV. Comparison data of Seg-time in TCP packet in TCP and UDP environment

	RED		**L_{map}-RED**	
No. of UDP flows	*Seg-time*	*Seg-time in packet*	*Seg-time*	*Seg-time in packet*
5	0.1031	83.67	0.0503	40.79
10	0.1283	104.15	0.0506	41.10
15	0.1484	120.44	0.0494	40.13
20	0.1980	160.71	0.0362	29.42
25	0.2067	167.80	0.0162	13.16
30	0.1368	111.06	0.0048	3.90
40	0.0944	76.61	0.0043	3.52
50	0.0682	55.37	0.0036	2.94

In light of the experimental results, what are the contributing factors to stabilizing chaotic queue oscillation and improving network performance? The chief difference in the algorithms of RED, AutoRED, and L_{map}-RED is of course the weight parameter, w_q, in EWMA. Fig. 42 illustrates this difference in AutoRED and L_{map}-RED since w_q in RED is a constant value 0.002.

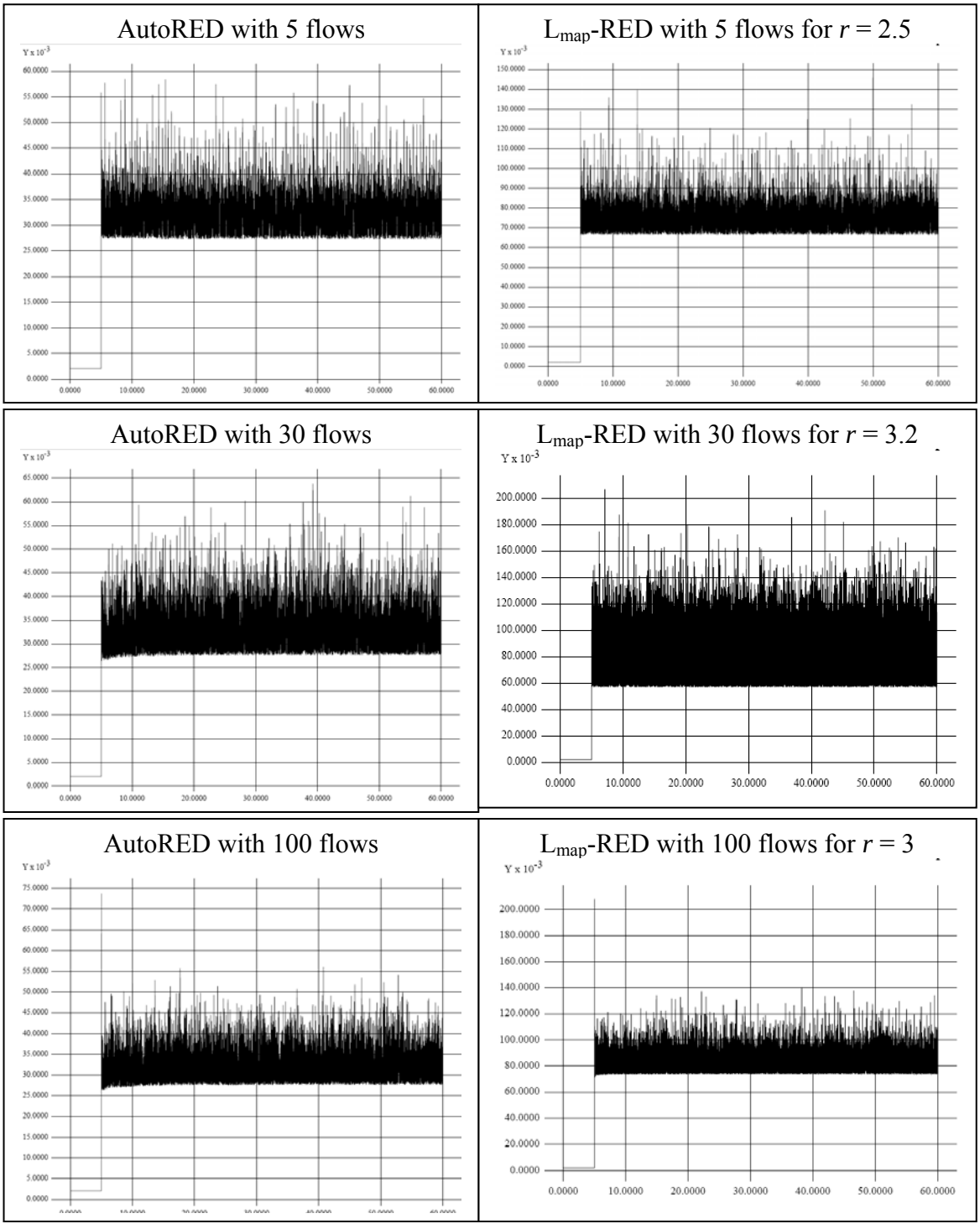

Figure 42. Comparison of w_q value of AutoRED and L_{map}-RED in TCP-only environment

Aside from the fact that w_q values in AutoRED and L_{map}-RED are changing dynamically, their lower bounds show differences. Comparing the algorithms of w_q calculation of AutoRED and L_{map}-RED, it can be seen that the differences in the lower bounds come from the first components, namely, the congestion characteristics in AutoRED, and the logistic map in L_{map}-RED.

In steady states as in the experiments, the congestion characteristics value seems to approach its upper bound, 0.25, rather quickly. In the case of L_{map}-RED, the lowest value of r value, 1.333, produces approximately the same value as the upper bound of the congestion characteristics. This is in fact the lower bound of the logistic map values produced by the rest of r values whose characteristics are presented in the fifth chapter. Fig. 43 shows these differences in each component of the algorithms of AutoRED and L_{map}-RED.

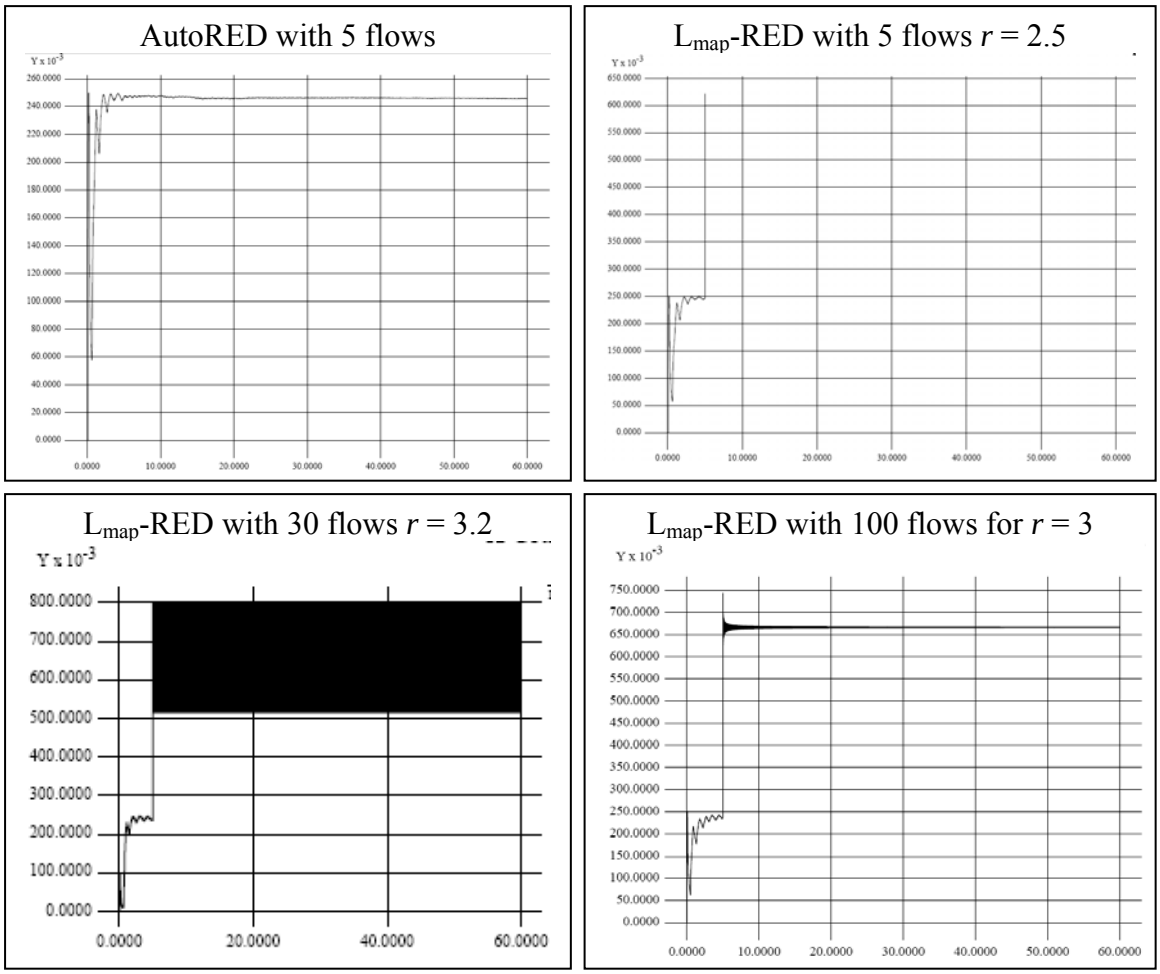

Figure 43. Comparison of congestion characteristics and logistic map values

Eventually, a higher w_q value makes the values of average queue size change as current queue size changes in a higher proportion. As plotting of average queue size over time in Fig. 44 shows, average queue size of AutoRED and L_{map}-RED moves in wider ranges than those of RED.

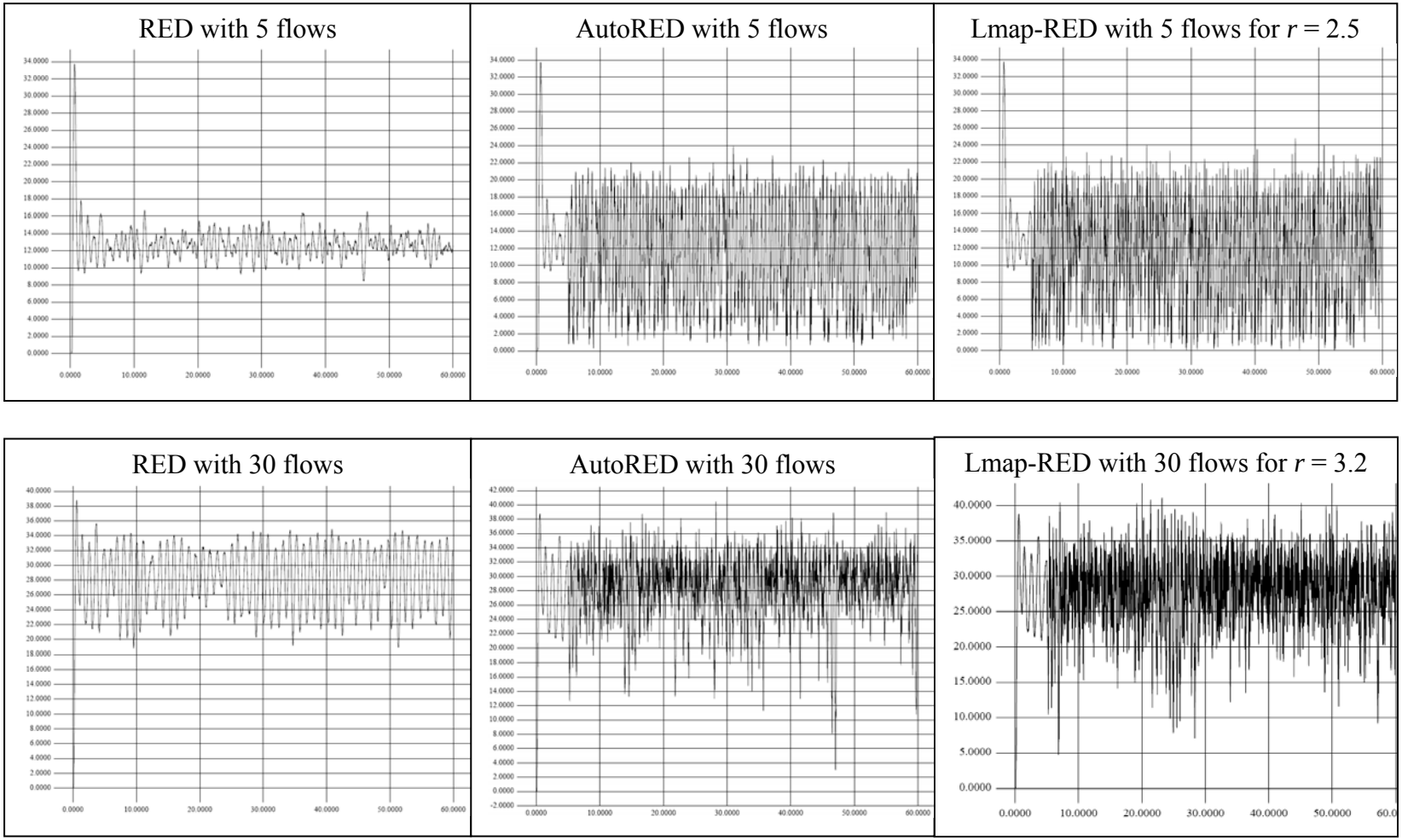

Figure 44. Comparison of average queue size over time in RED, AutoRED, and L_{map}-RED

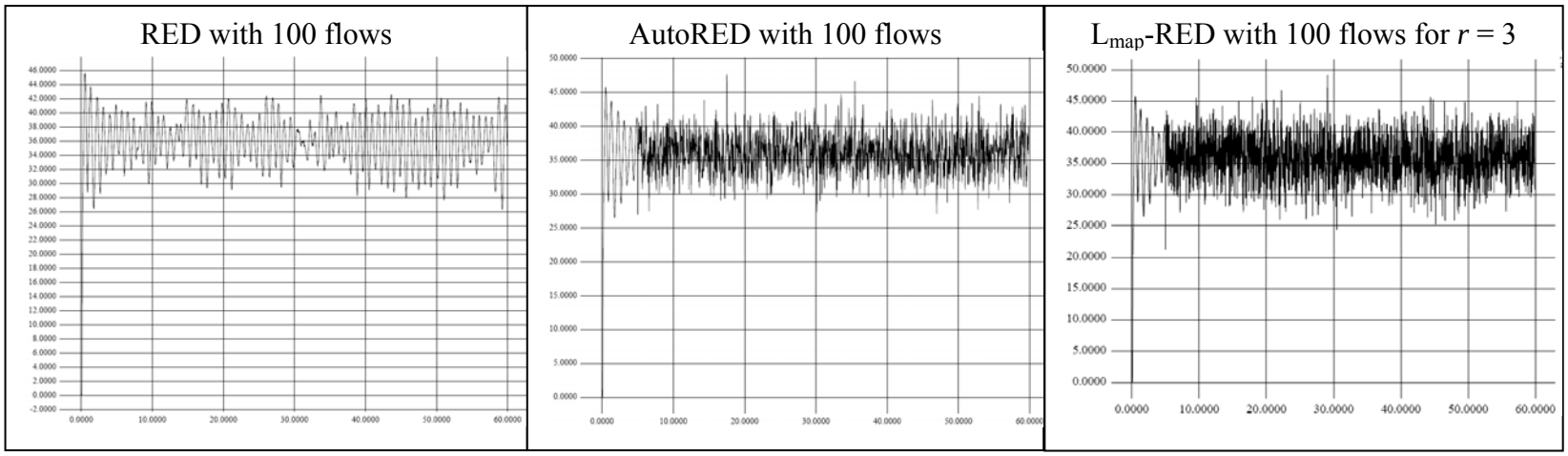

Figure 44. Comparison of average queue size over time in RED, AutoRED, and L_{map}-RED – continued

At this point, it is necessary to discuss certain characteristics of the behavior of the RED mechanism. As the experimental results show, RED seems to have achieved its design goals in that overall average queue size is closer to the maximum threshold and its packet loss rate is lower on a given number of TCP flows as long as the overall average queue size is not greater than the maximum threshold. However, as the overall average queue size becomes close to the maximum threshold, RED appears to drop packets at a much higher rate as its packet-marking probability increases as shown in Fig.12. As pointed out in [16], RED frequently exhibits poor throughput and increased packet loss rates especially when average queue size is larger than the maximum threshold.

In light of this, since average queue sizes on RED move in a smaller scale, when the overall average queue size is around the maximum threshold, calculated average queue sizes tend to stay in that range resulting in a higher packet loss rate and low throughput. On the other hand, in the case of AutoRED and L_{map}-RED, it can be said that a wider range of average queue size works in their advantage in that the sensitive nature of RED's parameters is watered down. Even when the overall average queue size is close to the maximum threshold, for example, in 30 TCP flows in Fig. 44, various average queue sizes are in mixed regions of packet-marking probabilities. L_{map}-RED introduces further randomizing effects through the bifurcations resulted by some r values in calculating w_q.

In terms of the complexity of computation, it is understandable that the algorithms of AutoRED and L_{map}-RED bound to increase their computing time and consume more computing power comparing with the algorithm of RED scheme. But comparing

AutoRED and L_{map}-RED, they should be at the same level because their computation part is virtually the same.

Albeit the favorable experimental results, one of the biggest drawbacks of L_{map}-RED is that the control parameter *r* value introduces the possibility of improvements but also unpredictability. Currently no relationship is observed between *r* values and network traffic characteristics or any aspects of network performance except that the *r* values used in the experiments yield lesser degrees of queue oscillation than those of RED.

CHAPTER VII

CONCLUSIONS

The new metric called Seg-time can definitely be used as a measurement for a type of delay caused by queue oscillation in a gateway that affects the network performance and the QoS. Seg-time answers how long a period of high queue size or low queue size persists with reference to the overall average queue size. It is validated by its direct relationship with the statistical metrics and visual representation of queue oscillation as a measurement of the degree of oscillation in terms of the QoS. However, it is not easy to calculate and it does not provide measurements instantaneously. Therefore, it may be more suitable for an in-depth network analysis rather than for dealing with what's happening currently in the network.

L_{map}-RED mechanism, an AQM scheme newly proposed in this thesis, shows the possibility of further improvements from what has been achieved by AutoRED. By modifying AutoRED's algorithm using the logistic map function, L_{map}-RED improves throughput and stabilizes the chaotic queue oscillation in highly congested networks where RED does not function well. Experimental results display that L_{map}-RED additionally improves packet loss rates in the TCP and UDP combined network environment. The biggest drawback of L_{map}-RED is that no guideline is set for choosing its control parameter, *r*, so that it will yield the best possible results for a given network

scenario. Therefore, AutoRED should be recommended for general purpose due to its advantage of calculating the weight parameter, w_q , automatically. For those users who are able to experiment with L_{map}-RED to find an appropriate r value for the characteristics of their network or for a particular purpose, L_{map}-RED may produce better results.

This leads to the future research topic for L_{map}-RED. Setting a guideline for choosing appropriate r value will require more thorough analysis of the relationship between the value r and the results it produces. In terms of the future work on Seg-time, it will include implementing the Seg-time measurement in live networks for the verification of its applicability to real life situations. Devising an algorithm easier to implement would be an ideal future topic as well.

REFERENCES

[1] S. Suthaharan, "Reduction of queue oscillation in the next generation Internet routers," Computer Communications, vol. 30, pp. 3881–3891 , December 2007.

[2] J. Nagle, "Congestion Control in IP/TCP Internetworks," RFC 896, January 1984.

[3] V. Jacobson, "Congestion Avoidance and Control," ACM SIGCOMM'88, August 1988.

[4] L. Brakmo, S. O'Malley, and L. Peterson, "TCP Vegas: New Techniques for Congestion Avoidance," in Proceedings of ACM SIGCOMM'94, pp. 24-35, Oct. 1994.

[5] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP," ACM Computer Communication Review, vol. 26, no. 3, pp. 5-21, July 1996.

[6] W. Stallings, Computer Networking with Internet Protocols and Technology. Upper Saddle River, NJ: Pearson Prentice Hall, 2004.

[7] B. Braden , D. Clark , J. Crowcroft , B. Davie , S. Deering , D. Estrin , S. Floyd , V. Jacobson , G. Minshall , C. Partridge , L. Peterson , K. Ramakrishnan , S. Shenker , J. Wroclawski , L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet," RFC 2309, April 1998.

[8] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Transactions on Networking, vol. 1, pp. 397–413, August 1993.

[9] R. J. Gibbens and F. Kelly, "Resource Pricing and the Evolution of Congestion Control," Automatica, vol. 35, pp. 1969–1985, 1999.

[10] S. Athuraliya, S. Low, V. H. Li, and Q. Yin, "REM: active queue management," IEEE Network, vol. 15, pp. 48–53, May/June 2001.

[11] S. Kunniyur and R. Srikant, "Analysis and design of adaptive virtual queue algorithm for active queue management," in Proceedings of ACM SIGCOMM, San Francisco, CA, 2001.

[12] M. May, C. Diot, B. Lyles, and J. Bolot, "Reasons not to deploy RED," Seventh International Workshop on 31 May–4 June 1999, IWQoS '99, pp. 260–262, 1999.

[13] D. Lin and R. Morris, “Dynamics of random early detection,” In Proceedings of the ACM Sigcomm, pp. 127–137, New York, NY, September 1997.

[14] T. Ott, T. Lakshman, and L. Wong, “SRED:stabilized RED,” In Proceedings of The IEEE Infocom, vol. 3, pp. 1346–1355, New York, NY, March 1999.

[15] W. Feng, K. Shin, D. Kandlur, and D. Saha, “The BLUE Active Queue Management Algorithm,” IEEE/ACM Transactions on Networking, vol. 10, no. 4, pp. 513-528, August 2002.

[16] S. Floyd, R. Gummadi, and S. Shenker, “Adaptive RED: An algorithm for increasing the robustness of RED’s active queue management,” August 2001. Available from: <http://www.icir.org/floyd/papers/adaptiveRed.pdf>.

[17] S. Liu, T. Başar and R. Srikant, "Exponential-RED: A Stabilizing AQM Scheme for Low- and High-Speed TCP Protocols", IEEE/ACM Transactions on Networking, vol. 13, no. 5, pp.1068-1081, October 2005.

[18] V. Firoiu, and M. Borden, “A study of active queue management for congestion control,” in Proc. of IEEE INFOCOM, Tel Aviv, Israel, 2000, pp. 1435–1444.

[19] P. Ranjan, E.H. Abed, and R.J. La, “Nonlinear instabilities in TCP-RED,” IEEE/ACM Transactions on Networking, vol. 12, pp. 1079-1092, December 2004.

[20] Cisco Systems, Inc., “Congestion Avoidance Overview,” October 2002. Available from:<http://www.cisco.com/en/US/docs/ios/12_0/qos/configuration/guide/qcconavd.html>.

[21] S. Strogatz, Nonlinear Dynamics and Chaos, Cambridge, MA: Perseus Publishing, 2000.

[22] Wikipedia the free encyclopedia, “Logistic map”, Available from: <http://en.wikipedia.org/wiki/Logistic_map>.

[23] E. Altman and T. Jiménez, “NS-2 for beginners,” 2003. Available from: <http://www-sop.inria.fr/maestro/personnel/Eitan.Altman/COURS-NS/n3.pdf>

[24] The Network Simulator: ns-2 [online]. Available from: <http://www.isi.edu/nsnam/ns/>.

[25] S. Floyd, “Recommendation on using the ‘gentle_’ variant of RED,” March 2000. Available from: <http://www.icir.org/floyd/red/gentle.html>.

[26] J.H.C. Nga, H.H.C. Iu, S.H. Ling and H.K. Lam, “Comparative study of stability in different TCP/RED models,” Chaos, Solitons & Fractals, vol. 37, pp. 977-987, August 2008.

Appendix A. L_{map}-RED Simulation Data

1. TCP-only environment

No. of TCP flows	*r value*	*Link utilization*	*Packet loss rate*	*Overall average queue size*	*Queuing delay*	*mini mum*	*maxi mum*	*standard deviation*	*Seg-time*
5	1.333	98.2154%	0.3001%	11.4934	0.01416	0	29	6.0623	0.0671
5	1.5	98.3262%	0.2997%	11.6814	0.01439	0	28	5.8745	0.0592
5	2	98.2933%	0.2998%	11.3659	0.01400	0	28	5.7816	0.0542
5	2.5	98.5645%	0.2990%	11.5176	0.01419	0	27	5.8140	0.0547
5	3	98.2175%	0.3001%	11.4971	0.01416	0	30	5.8521	0.0559
5	3.2	98.3982%	0.2996%	11.6192	0.01431	0	27	5.7462	0.0583
5	3.5	98.3013%	0.2999%	11.4449	0.01410	0	28	5.8640	0.0586
5	3.58	98.5395%	0.2991%	11.5120	0.01418	0	27	5.7322	0.0545
10	1.333	99.4864%	0.3045%	16.5766	0.02042	0	37	7.0797	0.0573
10	1.5	99.5006%	0.3064%	16.4633	0.02028	0	37	6.6938	0.0491
10	2	99.5318%	0.3104%	16.3572	0.02015	0	37	6.6802	0.0483
10	2.5	99.5046%	0.3290%	16.1552	0.01990	0	36	6.6237	0.0472
10	3	99.5109%	0.3228%	16.2579	0.02003	0	36	6.3017	0.0450
10	3.2	99.5580%	0.3083%	16.3604	0.02016	0	36	6.0475	0.0397
10	3.5	99.4923%	0.3332%	16.3985	0.02020	0	36	6.6287	0.0477
10	3.58	99.5232%	0.3125%	16.3327	0.02012	0	35	6.3819	0.0442
20	1.333	99.7847%	1.0984%	22.7821	0.02807	0	43	6.8629	0.0388
20	1.5	99.7512%	1.2734%	22.8735	0.02818	0	46	7.0162	0.0393
20	2	99.7923%	1.3072%	22.8942	0.02821	0	41	6.5926	0.0370
20	2.5	99.7839%	1.3892%	22.8164	0.02811	0	42	6.6539	0.0374
20	3	99.8070%	1.3429%	22.7401	0.02802	0	42	6.6193	0.0373
20	3.2	99.7532%	1.4216%	22.8421	0.02814	0	42	6.6316	0.0380
20	3.5	99.7900%	1.3171%	22.5154	0.02774	0	42	6.5437	0.0358
20	3.58	99.7867%	1.2874%	23.1039	0.02846	0	42	6.3293	0.0334
22	1.333	99.7306%	1.7766%	24.6977	0.03043	0	46	6.5795	0.0353
22	1.5	99.7351%	1.8597%	24.5445	0.03024	0	44	6.5969	0.0361
22	2	99.7327%	1.9910%	24.4856	0.03017	0	43	6.4515	0.0361
22	2.5	99.7306%	1.9705%	24.3206	0.02996	0	46	6.3738	0.0347
22	3	99.7165%	2.0722%	24.2206	0.02984	0	42	6.6446	0.0369
22	3.2	99.7333%	2.0285%	24.3360	0.02998	0	43	6.4329	0.0347
22	3.5	99.7289%	2.0083%	24.5035	0.03019	0	43	6.2666	0.0332
22	3.58	99.7247%	1.9179%	24.0372	0.02961	0	44	6.4758	0.0351
23	1.333	99.5915%	2.5107%	26.0221	0.03206	0	47	6.4624	0.0330
23	1.5	99.5580%	2.5724%	25.5069	0.03142	0	44	6.5705	0.0364
23	2	99.5786%	2.5956%	25.8463	0.03184	0	42	6.0198	0.0291
23	2.5	99.5855%	2.7688%	25.1497	0.03098	0	43	6.4654	0.0353
23	3	99.5909%	2.6755%	25.5044	0.03142	0	43	6.0677	0.0314
23	3.2	99.5704%	2.6307%	25.3496	0.03123	0	42	6.1229	0.0327

No. of TCP flows	*r value*	*Link utilization*	*Packet loss rate*	*Overall average queue size*	*Queuing delay*	*mini mum*	*maxi mum*	*standard deviation*	*Seg-time*
23	3.5	99.5919%	2.6985%	25.3347	0.03121	0	43	6.3615	0.0325
23	3.58	99.5826%	2.7452%	25.2130	0.03106	0	44	6.4114	0.0351
25	1.333	99.8255%	3.3135%	26.1525	0.03222	0	48	6.9822	0.0394
25	1.5	99.8272%	3.1460%	26.3086	0.03241	0	44	6.4909	0.0331
25	2	99.8331%	3.1302%	26.3491	0.03246	0	45	6.0008	0.0300
25	2.5	99.8375%	3.1573%	26.2249	0.03231	0	46	6.0234	0.0285
25	3	99.8358%	3.3224%	26.2274	0.03231	0	44	6.0973	0.0312
25	3.2	99.8270%	3.3975%	25.9988	0.03203	0	43	6.4574	0.0357
25	3.5	99.8228%	3.3070%	26.2978	0.03240	0	45	6.1273	0.0308
25	3.58	99.8375%	2.9871%	26.2461	0.03234	5	44	5.7524	0.0278
30	1.333	99.7902%	4.5615%	27.6870	0.03411	0	46	6.5837	0.0338
30	1.5	99.8024%	4.5200%	27.7270	0.03416	2	46	6.1868	0.0315
30	2	99.7964%	4.5519%	27.6303	0.03404	0	46	6.0621	0.0291
30	2.5	99.8024%	4.5426%	27.8097	0.03426	3	43	5.5004	0.0260
30	3	99.7965%	4.5898%	27.5326	0.03392	0	44	5.9051	0.0302
30	3.2	99.8024%	4.4212%	27.6168	0.03402	1	44	5.7186	0.0277
30	3.5	99.8004%	4.4961%	27.5962	0.03400	0	45	5.7960	0.0271
30	3.58	99.8024%	4.5698%	27.8203	0.03427	1	43	5.5041	0.0258
40	1.333	99.7945%	8.0955%	30.6107	0.03771	5	49	5.5443	0.0239
40	1.5	99.7945%	8.2001%	30.6752	0.03779	6	49	5.1196	0.0225
40	2	99.7945%	7.9801%	30.4930	0.03757	5	49	5.1754	0.0207
40	2.5	99.7944%	7.9799%	30.5068	0.03758	10	46	4.8461	0.0188
40	3	99.7945%	7.9686%	30.2521	0.03727	7	47	5.2359	0.0210
40	3.2	99.7945%	7.9453%	30.1900	0.03719	5	47	5.3449	0.0232
40	3.5	99.7942%	7.8811%	30.2727	0.03730	8	46	5.1043	0.0198
40	3.58	99.7945%	8.0459%	30.4783	0.03755	8	46	5.0171	0.0213
50	1.333	99.8354%	11.5478%	32.4813	0.04002	5	50	4.6669	0.0171
50	1.5	99.8354%	11.2230%	32.2680	0.03975	12	48	4.6849	0.0179
50	2	99.8354%	11.4426%	32.3890	0.03990	13	48	4.5696	0.0164
50	2.5	99.8354%	11.2532%	32.2733	0.03976	12	48	4.5459	0.0155
50	3	99.8354%	11.2287%	32.2532	0.03974	14	46	4.3357	0.0150
50	3.2	99.8353%	11.0888%	32.2248	0.03970	14	47	4.4831	0.0160
50	3.5	99.8354%	11.2922%	32.3149	0.03981	10	48	4.4760	0.0154
50	3.58	99.8352%	11.3890%	32.2358	0.03971	7	47	4.5829	0.0168
60	1.333	99.8785%	13.8500%	33.4388	0.04120	14	52	4.6229	0.0182
60	1.5	99.8785%	14.3094%	33.7162	0.04154	16	49	4.4227	0.0157
60	2	99.8785%	13.8414%	33.4653	0.04123	15	46	4.2903	0.0155
60	2.5	99.8785%	13.9250%	33.5627	0.04135	14	48	4.3038	0.0148
60	3	99.8785%	13.6780%	33.4311	0.04119	16	47	4.1742	0.0140
60	3.2	99.8785%	14.0041%	33.5116	0.04129	18	48	4.2284	0.0147
60	3.5	99.8783%	13.8421%	33.5004	0.04127	16	49	4.1095	0.0140
60	3.58	99.8785%	14.3900%	33.7666	0.04160	14	49	4.1492	0.0143

No. of TCP flows	*r value*	*Link utilization*	*Packet loss rate*	*Overall average queue size*	*Queuing delay*	*mini mum*	*maxi mum*	*standard deviation*	*Seg-time*
70	1.333	99.8969%	15.8019%	34.2865	0.04224	17	51	4.4528	0.0158
70	1.5	99.8970%	16.0015%	34.3697	0.04234	17	49	4.2784	0.0156
70	2	99.8970%	16.2022%	34.4791	0.04248	17	50	4.1071	0.0140
70	2.5	99.8970%	15.8991%	34.3404	0.04231	18	49	4.0114	0.0131
70	3	99.8970%	15.7897%	34.2739	0.04223	16	48	4.2776	0.0147
70	3.2	99.8970%	15.5751%	34.2127	0.04215	16	49	4.1860	0.0148
70	3.5	99.8970%	16.2657%	34.5302	0.04254	15	48	4.0713	0.0140
70	3.58	99.8970%	15.5924%	34.2510	0.04220	17	48	4.0626	0.0134
80	1.333	99.9054%	17.3060%	34.9723	0.04309	17	51	4.3157	0.0173
80	1.5	99.9059%	17.6289%	35.1183	0.04327	19	50	4.1156	0.0144
80	2	99.9055%	17.0856%	34.8445	0.04293	19	50	4.1326	0.0146
80	2.5	99.9055%	17.4894%	35.0315	0.04316	19	49	3.8986	0.0133
80	3	99.9059%	17.3282%	34.9486	0.04306	18	48	3.9393	0.0141
80	3.2	99.9053%	17.0319%	34.8208	0.04290	15	49	3.9804	0.0140
80	3.5	99.9053%	17.1874%	34.9211	0.04302	19	49	3.9529	0.0143
80	3.58	99.9050%	17.4139%	34.9900	0.04311	17	48	3.9475	0.0143
90	1.333	99.8971%	19.0714%	35.6710	0.04395	18	51	4.2034	0.0165
90	1.5	99.8971%	18.7277%	35.5161	0.04376	16	51	4.1626	0.0150
90	2	99.8968%	18.6498%	35.4710	0.04370	18	50	4.0619	0.0141
90	2.5	99.8970%	19.1389%	35.7052	0.04399	18	50	4.0259	0.0143
90	3	99.8970%	18.7626%	35.5644	0.04382	16	49	4.0401	0.0143
90	3.2	99.8971%	18.7610%	35.5569	0.04381	20	50	3.8392	0.0130
90	3.5	99.8971%	18.8012%	35.5342	0.04378	18	49	3.8963	0.0134
90	3.58	99.8971%	18.6737%	35.4993	0.04374	19	50	3.9448	0.0141
100	1.333	99.8950%	19.9875%	36.0736	0.04444	18	50	4.1838	0.0151
100	1.5	99.8948%	20.1378%	36.1269	0.04451	16	50	3.9942	0.0139
100	2	99.8950%	20.4529%	36.3522	0.04479	20	49	3.9269	0.0134
100	2.5	99.8949%	20.1542%	36.1212	0.04450	19	49	3.7829	0.0130
100	3	99.8947%	19.6624%	35.8976	0.04423	20	51	3.8969	0.0142
100	3.2	99.8947%	19.7558%	35.9814	0.04433	21	48	3.9275	0.0145
100	3.5	99.8950%	19.9148%	36.0619	0.04443	20	49	3.8603	0.0134
100	3.58	99.8948%	20.0214%	36.0762	0.04445	21	50	3.7691	0.0127
120	1.333	99.9199%	21.5077%	36.7021	0.04522	18	52	4.1678	0.0160
120	1.5	99.9204%	22.0060%	36.9478	0.04552	21	51	4.0204	0.0159
120	2	99.9208%	21.7931%	36.8780	0.04543	23	50	3.9472	0.0144
120	2.5	99.9202%	21.6454%	36.7562	0.04528	20	50	3.9302	0.0144
120	3	99.9196%	21.4909%	36.6947	0.04521	20	50	3.8391	0.0139
120	3.2	99.9194%	21.2969%	36.5995	0.04509	18	49	3.8322	0.0130
120	3.5	99.9198%	22.2306%	37.0792	0.04568	21	53	3.8528	0.0135
120	3.58	99.9197%	21.4956%	36.7148	0.04523	21	53	3.9715	0.0140
140	1.333	99.9236%	23.1522%	37.3416	0.04600	22	55	4.1534	0.0154
140	1.5	99.9236%	23.1140%	37.3556	0.04602	21	50	4.0530	0.0147

No. of TCP flows	*r value*	*Link utilization*	*Packet loss rate*	*Overall average queue size*	*Queuing delay*	*mini mum*	*maxi mum*	*standard deviation*	*Seg-time*
140	2	99.9236%	23.2014%	37.3985	0.04607	22	50	3.8178	0.0136
140	2.5	99.9236%	23.2357%	37.4722	0.04617	20	51	3.7911	0.0130
140	3	99.9236%	23.1902%	37.3994	0.04608	22	51	3.7963	0.0131
140	3.2	99.9236%	23.0691%	37.3585	0.04603	20	51	3.7985	0.0134
140	3.5	99.9236%	22.9173%	37.3190	0.04598	20	50	3.8410	0.0132
140	3.58	99.9236%	22.6621%	37.1354	0.04575	22	49	3.8004	0.0130
180	1.333	99.9257%	25.0508%	38.1968	0.04706	19	53	4.1636	0.0162
180	1.5	99.9255%	25.0992%	38.1985	0.04706	22	53	3.9808	0.0145
180	2	99.9255%	24.5788%	37.9557	0.04676	21	50	3.9298	0.0149
180	2.5	99.9256%	25.2850%	38.2514	0.04713	23	51	3.8087	0.0137
180	3	99.9259%	25.1782%	38.2574	0.04713	23	51	3.7367	0.0137
180	3.2	99.9260%	24.8863%	38.1301	0.04698	20	51	3.8078	0.0126
180	3.5	99.9255%	24.9036%	38.0727	0.04691	23	50	3.8328	0.0131
180	3.58	99.9266%	24.8275%	38.0770	0.04691	21	51	3.8203	0.0138
240	1.333	99.9321%	27.3763%	39.0643	0.04813	23	53	4.1028	0.0157
240	1.5	99.9321%	27.4620%	39.0846	0.04815	23	54	3.9431	0.0143
240	2	99.9319%	27.1491%	38.9389	0.04797	23	53	3.7243	0.0139
240	2.5	99.9321%	27.1663%	38.9332	0.04797	23	51	3.6962	0.0142
240	3	99.9321%	28.1069%	39.3974	0.04854	25	53	3.6534	0.0127
240	3.2	99.9318%	27.2450%	38.9870	0.04803	24	51	3.7087	0.0149
240	3.5	99.9321%	27.4619%	39.0656	0.04813	26	52	3.6146	0.0127
240	3.58	99.9319%	27.3426%	39.0768	0.04814	23	53	3.5547	0.0124

2. TCP and UDP combined environment

No. of UDP flows	*r value*	*Traffic type*	*Link utilization*	*Packet loss rate*	*Overall average queue size*	*mini mum*	*maxi mum*	*standard deviation*	*Seg-time*
5	1.333	TCP	81.1181%	0.2604%					
5	1.333	UDP	18.5872%	3.1500%					
5	1.333	Total	99.7053%	1.0306%	18.82921	0	39	7.6927	0.0633
5	1.5	TCP	81.0856%	0.2244%					
5	1.5	UDP	18.6016%	3.1333%					
5	1.5	Total	99.6872%	1.0006%	18.76784	0	41	7.0351	0.0559
5	2	TCP	81.2442%	0.2269%					
5	2	UDP	18.5984%	3.1000%					
5	2	Total	99.8426%	0.9922%	18.88889	0	38	6.9599	0.0536

No. of UDP flows	*r value*	*Traffic type*	*Link utilization*	*Packet loss rate*	*Overall average queue size*	*mini mum*	*maxi mum*	*standard deviation*	*Seg-time*
5	2.5	TCP	81.3718%	0.1904%					
5	2.5	UDP	18.5856%	3.1750%					
5	2.5	Total	99.9574%	0.9847%	18.80505	0	39	6.8207	0.0503
5	3	TCP	81.2104%	0.2933%					
5	3	UDP	18.6112%	3.0583%					
5	3	Total	99.8216%	1.0295%	18.60136	0	38	7.1054	0.0545
5	3.2	TCP	81.2630%	0.3354%					
5	3.2	UDP	18.5616%	3.3167%					
5	3.2	Total	99.8246%	1.1288%	18.63506	0	39	7.4639	0.0601
5	3.5	TCP	81.2463%	0.2600%					
5	3.5	UDP	18.6176%	3.0000%					
5	3.5	Total	99.8639%	0.9894%	18.78099	0	38	7.0575	0.0529
5	3.58	TCP	81.2470%	0.2088%					
5	3.58	UDP	18.6512%	2.8583%					
5	3.58	Total	99.8982%	0.9146%	18.56178	0	38	7.0007	0.0523
10	1.333	TCP	65.5156%	0.5385%					
10	1.333	UDP	34.0016%	3.3682%					
10	1.333	Total	99.5172%	1.8156%	19.56807	0	43	8.0888	0.0646
10	1.5	TCP	66.0402%	0.8465%					
10	1.5	UDP	33.7248%	4.1409%					
10	1.5	Total	99.7650%	2.3241%	20.95036	0	41	7.9524	0.0630
10	2	TCP	65.8736%	0.9224%					
10	2	UDP	33.7600%	4.0545%					
10	2	Total	99.6336%	2.3288%	20.70374	0	41	7.5945	0.0561
10	2.5	TCP	66.0403%	0.8287%					
10	2.5	UDP	33.8032%	3.9591%					
10	2.5	Total	99.8435%	2.2333%	20.88060	0	40	7.5170	0.0608
10	3	TCP	66.1200%	0.8129%					
10	3	UDP	33.7200%	4.1500%					
10	3	Total	99.8400%	2.3092%	21.36937	0	41	7.2289	0.0506
10	3.2	TCP	65.7365%	0.8436%					
10	3.2	UDP	33.8160%	3.9227%					
10	3.2	Total	99.5525%	2.2286%	20.91545	0	40	7.5111	0.0597
10	3.5	TCP	65.8903%	1.0028%					
10	3.5	UDP	33.7888%	3.9773%					
10	3.5	Total	99.6791%	2.3377%	20.53946	0	43	7.9181	0.0627
10	3.58	TCP	65.9128%	0.8703%					
10	3.58	UDP	33.8320%	3.8500%					
10	3.58	Total	99.7448%	2.2081%	20.55094	0	40	7.6445	0.0563

No. of UDP flows	*r value*	*Traffic type*	*Link utilization*	*Packet loss rate*	*Overall average queue size*	*mini mum*	*maxi mum*	*standard deviation*	*Seg-time*
15	1.333	TCP	51.1098%	1.9927%					
15	1.333	UDP	48.5520%	5.1219%					
15	1.333	Total	99.6618%	3.8757%	23.00018	0	44	8.2003	0.0609
15	1.5	TCP	51.3577%	2.1827%					
15	1.5	UDP	48.4752%	5.2938%					
15	1.5	Total	99.8329%	4.0504%	23.29369	0	43	8.0386	0.0617
15	2	TCP	51.2257%	1.9375%					
15	2	UDP	48.4768%	5.2719%					
15	2	Total	99.7025%	3.9426%	23.32106	0	42	7.7336	0.0583
15	2.5	TCP	51.1309%	1.7375%					
15	2.5	UDP	48.7360%	4.7781%					
15	2.5	Total	99.8669%	3.5691%	22.73165	0	41	7.5325	0.0556
15	3	TCP	51.2792%	1.9081%					
15	3	UDP	48.6304%	5.0094%					
15	3	Total	99.9096%	3.7727%	23.22719	0	43	7.3833	0.0524
15	3.2	TCP	51.4227%	2.0324%					
15	3.2	UDP	48.3632%	5.4875%					
15	3.2	Total	99.7859%	4.1066%	23.55310	0	43	7.6033	0.0512
15	3.5	TCP	51.2939%	1.8011%					
15	3.5	UDP	48.5504%	5.1219%					
15	3.5	Total	99.8443%	3.7982%	23.59923	0	41	7.1799	0.0494
15	3.58	TCP	51.3947%	2.1066%					
15	3.58	UDP	48.4928%	5.2500%					
15	3.58	Total	99.8875%	3.9933%	23.19219	0	42	7.8523	0.0582
20	1.333	TCP	37.0918%	3.3434%					
20	1.333	UDP	62.8240%	6.4690%					
20	1.333	Total	99.9158%	5.6232%	25.45406	0	45	7.4505	0.0535
20	1.5	TCP	37.1188%	3.9789%					
20	1.5	UDP	62.7616%	6.5690%					
20	1.5	Total	99.8804%	5.8640%	25.35854	0	45	7.9714	0.0574
20	2	TCP	37.6065%	3.8057%					
20	2	UDP	62.3440%	7.1833%					
20	2	Total	99.9505%	6.2570%	26.51647	0	44	6.7626	0.0410
20	2.5	TCP	37.0417%	3.3485%					
20	2.5	UDP	62.8928%	6.3571%					
20	2.5	Total	99.9345%	5.5439%	25.89003	0	42	6.7574	0.0425
20	3	TCP	37.0392%	3.5460%					
20	3	UDP	62.8608%	6.4167%					
20	3	Total	99.9000%	5.6394%	25.48309	0	42	7.0989	0.0496

No. of UDP flows	*r value*	*Traffic type*	*Link utilization*	*Packet loss rate*	*Overall average queue size*	*mini mum*	*maxi mum*	*standard deviation*	*Seg-time*
20	3.2	TCP	37.4189%	3.9405%					
20	3.2	UDP	62.5296%	6.9476%					
20	3.2	Total	99.9485%	6.1252%	26.37617	0	43	6.5278	0.0362
20	3.5	TCP	37.0935%	3.3932%					
20	3.5	UDP	62.8544%	6.4262%					
20	3.5	Total	99.9479%	5.6051%	25.79748	0	43	6.8167	0.0432
20	3.58	TCP	37.4806%	4.3484%					
20	3.58	UDP	62.4064%	7.0976%					
20	3.58	Total	99.8870%	6.3422%	26.38697	0	44	7.0383	0.0455
25	1.333	TCP	25.1516%	7.9805%					
25	1.333	UDP	74.8496%	9.9885%					
25	1.333	Total	100.0012%	9.6352%	30.52464	0	45	4.4333	0.0192
25	1.5	TCP	25.4796%	8.6211%					
25	1.5	UDP	74.5216%	10.3923%					
25	1.5	Total	100.0012%	10.0756%	30.59180	2	45	4.8453	0.0222
25	2	TCP	25.7445%	8.7545%					
25	2	UDP	74.2560%	10.7019%					
25	2	Total	100.0005%	10.3503%	30.85828	3	44	4.3661	0.0184
25	2.5	TCP	24.8771%	8.2319%					
25	2.5	UDP	75.1248%	9.6865%					
25	2.5	Total	100.0019%	9.4324%	30.17606	0	44	4.6207	0.0203
25	3	TCP	25.5580%	8.8674%					
25	3	UDP	74.4400%	10.4865%					
25	3	Total	99.9980%	10.1956%	30.89379	0	42	4.0769	0.0162
25	3.2	TCP	24.3309%	6.9557%					
25	3.2	UDP	75.6704%	9.0346%					
25	3.2	Total	100.0013%	8.6823%	29.62063	5	42	4.5361	0.0215
25	3.5	TCP	25.5844%	8.7106%					
25	3.5	UDP	74.4160%	10.5231%					
25	3.5	Total	100.0004%	10.1977%	30.84437	5	43	4.2688	0.0167
25	3.58	TCP	24.7760%	8.2748%					
25	3.58	UDP	75.2256%	9.5769%					
25	3.58	Total	100.0016%	9.3502%	30.21555	8	45	4.4064	0.0212
30	1.333	TCP	16.3806%	16.0773%					
30	1.333	UDP	83.6224%	15.7016%					
30	1.333	Total	100.0030%	15.7442%	34.13364	24	46	2.9847	0.0053
30	1.5	TCP	16.0198%	14.2801%					
30	1.5	UDP	83.9808%	15.3371%					
30	1.5	Total	100.0006%	15.2219%	33.86068	19	46	3.0682	0.0068

No. of UDP flows	*r value*	*Traffic type*	*Link utilization*	*Packet loss rate*	*Overall average queue size*	*mini mum*	*maxi mum*	*standard deviation*	*Seg-time*
30	2	TCP	16.2781%	15.4427%					
30	2	UDP	83.7232%	15.6016%					
30	2	Total	100.0013%	15.5838%	34.06685	23	44	2.8788	0.0048
30	2.5	TCP	16.6321%	15.1565%					
30	2.5	UDP	83.3696%	15.9581%					
30	2.5	Total	100.0017%	15.8669%	34.21761	24	45	2.8710	0.0050
30	3	TCP	16.4926%	15.4584%					
30	3	UDP	83.5104%	15.8145%					
30	3	Total	100.0030%	15.7742%	34.13299	24	45	2.9631	0.0058
30	3.2	TCP	15.4032%	14.9252%					
30	3.2	UDP	84.5968%	14.7161%					
30	3.2	Total	100.0000%	14.7383%	33.64231	19	44	3.0014	0.0060
30	3.5	TCP	15.3194%	14.9863%					
30	3.5	UDP	84.6816%	14.6371%					
30	3.5	Total	100.0010%	14.6740%	33.62001	22	44	2.8979	0.0054
30	3.58	TCP	14.8566%	13.9678%					
30	3.58	UDP	85.1456%	14.1629%					
30	3.58	Total	100.0022%	14.1431%	33.37986	22	43	2.9505	0.0059
40	1.333	TCP	3.2981%	27.0433%					
40	1.333	UDP	96.7200%	26.2793%					
40	1.333	Total	100.0181%	26.2959%	38.83069	33	47	2.4673	0.0044
40	1.5	TCP	3.6827%	28.0135%					
40	1.5	UDP	96.3184%	26.5878%					
40	1.5	Total	100.0011%	26.6230%	39.07895	33	48	2.5074	0.0044
40	2	TCP	3.5460%	28.1578%					
40	2	UDP	96.4544%	26.4866%					
40	2	Total	100.0004%	26.5264%	39.10552	33	48	2.4952	0.0044
40	2.5	TCP	2.7750%	28.4345%					
40	2.5	UDP	97.2304%	25.8927%					
40	2.5	Total	100.0054%	25.9403%	38.83135	33	48	2.4818	0.0044
40	3	TCP	3.0581%	28.1323%					
40	3	UDP	96.9424%	26.1098%					
40	3	Total	100.0005%	26.1514%	38.97079	33	47	2.4651	0.0043
40	3.2	TCP	4.2218%	27.9344%					
40	3.2	UDP	95.7792%	27.0000%					
40	3.2	Total	100.0010%	27.0263%	39.43389	33	48	2.5224	0.0043
40	3.5	TCP	5.1868%	28.0780%					
40	3.5	UDP	94.8144%	27.7329%					
40	3.5	Total	100.0012%	27.7448%	39.80092	33	48	2.5250	0.0044

No. of UDP flows	*r value*	*Traffic type*	*Link utilization*	*Packet loss rate*	*Overall average queue size*	*mini mum*	*maxi mum*	*standard deviation*	*Seg-time*
40	3.58	TCP	4.5120%	28.7222%					
40	3.58	UDP	95.4880%	27.2134%					
40	3.58	Total	100.0000%	27.2591%	39.51225	33	49	2.5129	0.0044
50	1.333	TCP	0.8798%	40.7654%					
50	1.333	UDP	99.1216%	39.2637%					
50	1.333	Total	100.0014%	39.2725%	43.97220	38	52	2.2863	0.0036
50	1.5	TCP	0.5108%	45.6973%					
50	1.5	UDP	99.5488%	39.0029%					
50	1.5	Total	100.0596%	39.0250%	44.00059	38	51	2.2909	0.0039
50	2	TCP	0.7078%	41.8367%					
50	2	UDP	99.2960%	39.1569%					
50	2	Total	100.0038%	39.1697%	44.21833	39	51	2.2987	0.0039
50	2.5	TCP	0.7428%	42.4901%					
50	2.5	UDP	99.2752%	39.1676%					
50	2.5	Total	100.0180%	39.1840%	44.28871	39	51	2.2870	0.0038
50	3	TCP	0.6700%	42.0259%					
50	3	UDP	99.3328%	39.1343%					
50	3	Total	100.0028%	39.1474%	44.29510	39	51	2.2869	0.0038
50	3.2	TCP	1.2230%	37.2751%					
50	3.2	UDP	98.7952%	39.4627%					
50	3.2	Total	100.0182%	39.4462%	44.38905	39	52	2.2962	0.0038
50	3.5	TCP	1.1417%	38.2716%					
50	3.5	UDP	98.8880%	39.4069%					
50	3.5	Total	100.0297%	39.3988%	44.38463	39	51	2.2874	0.0038
50	3.58	TCP	0.6804%	41.8947%					
50	3.58	UDP	99.3200%	39.1422%					
50	3.58	Total	100.0004%	39.1549%	44.30534	39	52	2.2885	0.0038