Self Organizing Map Based RED Parameter

Estimation for Congestion Avoidance

ÖZEN YELBAŞI

Ph.D. Dissertation

Graduate School of Sciences

Electrical and Electronics

Engineering Program

May 2011

ANADOLU ÜNİVERSİTESİ

JÜRİ VE ENSTİTÜ ONAYI

Özen Yelbaşı'nın **Self Organizing Map Based RED Parameter Estimation for Congestion Avoidance** başlıklı **Elektrik-Elektronik Mühendisliği** Anabilim Dalı **Kontrol ve Kumanda Sistemleri** Bilim Dalındaki Doktora Tezi 17/03/2011 tarihinde aşağıdaki jüri tarafından Anadolu Üniversitesi Lisansüstü Eğitim-Ögretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

	Adı-Soyadı	**İmza**
Üye (Tez Danışmanı)	**: Yrd. Doç. Dr. Emin GERMEN**	.....
Üye	**: Doç. Dr. Yusuf OYSAL**	.....
Üye	**: Yrd. Doç. Dr. Hakan Güray ŞENEL**	.....
Üye	**: Yrd. Doç. Dr. Cüneyt AKINLAR**	.....
Üye	**: Yrd. Doç. Dr. Erol SEKE**	.....

Anadolu Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..... **tarih ve** ..... **sayılı kararıyla onaylanmıştır.**

Enstitü Müdürü

ABSTRACT

Ph.D. Dissertation

Self Organizing Map Based RED Parameter Estimation for Congestion Avoidance

Özen Yelbaşı

Anadolu University
Graduate School of Sciences
Electrical and Electronics Engineering Program

Supervisor: Asst. Prof. Dr. Emin GERMEN

2011, 108 pages

In data communication networks, congestion avoidance in routers is one of the favorite research topics. In this dissertation, new queue management approaches are proposed on the Random Early Detection (RED) algorithm by monitoring the global congestion status of an autonomous system. In order to view the level of global congestion, a traffic flow is generated between routers and a centralized observation unit. Internet Protocol (IP) routers are specialized in order to send information packets, regarding current output queue lengths, to the observation unit. By this way, it becomes possible to produce a global picture of congestion. A Self Organizing Map (SOM) structure is used by the observation unit, to make predictions on the future congestion behaviour of the network. After some improvements, routers gain the ability to update their RED parameters according to the congestion notifications sent by the observation unit. In this work, the benchmarks of new queue management approaches are investigated by simulations on OPNET Modeler and comparisons with recent queue management technologies: Drop Tail (DT) and RED, are performed. The results for throughput, packet loss, end-to-end delay and delay variation are observed.

Keywords: Congestion avoidance, Random Early Detection, Drop Tail, Self Organizing Map.

ÖZET

Doktora Tezi

Tıkanıklık Önlemede Öz Düzenlemeli Ağ Kullanılarak RED Parametrelerinin Kestirimi

Özen Yelbaşı

Anadolu Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman: Yard. Doç. Dr. Emin GERMEN

2011, 108 sayfa

Veri iletim ağlarında, yönlendiriciler üzerinde meydana gelen tıkanıklığın önlenmesi en popüler araştırma konularından birisidir. Bu çalışmada, bir ağ yapısındaki genel tıkanıklık durumu gözlemlenerek RED algoritmasına dayanan yeni kuyruk yönetim modelleri üretilmiştir. Tıkanıklık seviyesinin anlık durumunun izlenebilmesi için merkezi gözlem birimi ile yönlendiriciler arasında trafik akışı yaratılmıştır. Yönlendiriciler, kuyruk uzunluğu değerlerini taşıyan bilgi paketlerini merkezi birime gönderebilecek şekilde özelleştirilmiştir. Bu bilgi trafiği sayesinde tıkanıklığın genel durumunu gözlemlemek mümkün olmaktadır. Merkezi gözlem birimi, tıkanıklık seviyesinin gelecekteki yönelimini öngörebilmek amacıyla öz düzenlemeli ağ kullanmaktadır. Yönlendiriciler üzerinde başka bazı iyileştirmeler de yapılmıştır, öyle ki, merkezi birimden gelen tıkanıklık durum bilgisine bağlı olarak istenen RED parametreleri güncellenebilmektedir. Yeni kuyruk yönetim modelleri OPNET Modeler programı kullanılarak denenmiş, Drop Tail ve RED yönetim modelleriyle kıyaslanmıştır. Paket kayıpları ve zaman gecikmeleri gibi değerlerin zamanla değişimi incelenmiştir.

Anahtar Kelimeler: Tıkanıklık önleme, RED kuyruk yönetim modeli, DT kuyruk yönetim modeli, öz düzenlemeli ağ.

TEŞEKKÜR

Sahip olduğu bilgi birikimi, azmi ve enerjisiyle bu tezi tamamlamamda en çok emeği olan, tez danışmanım Yrd. Doç. Dr. Emin Germen'e,

Jürimde yer alarak bu tezin oluşumuna önemli katkı sağlayan Doç. Dr. Yusuf Oysal'a, Yrd. Doç. Dr. Hakan Güray Şenel'e, Yrd. Doç. Dr. Cüneyt Akınlar'a ve Yrd. Doç. Dr. Erol Seke'ye,

Tez önerisi aşamasında verdiği değerli fikirlerden ötürü Prof. Dr. Altuğ İftar'a,

saygılarımı ve teşekkürlerimi sunuyorum...

Okul ve iş hayatında karşılaşılan problemlerin çözümsüz olmadığını anlamamı sağlayan değerli çalışma arkadaşlarım Hakkı Ulaş Ünal ve Murat Başaran'a,

Sahip olduğu incelikli ruh ve yaşama sevinciyle bana destek olan canım arkadaşım Elif Eren'e,

Karşılaştığım tüm zorluklara benimle birlikte göğüs geren, hiçbir fedakarlıktan kaçınmayan, beni yüreklendirerek daima yanımda olduğunu hissettiren, varlığıyla beni yaşama bağlayan canım aileme,

Ve bu tezin tamamlanmasını en az benim kadar yürekten isteyen, tanımaktan mutluluk duyduğum tüm güzel insanlara,

sevgilerimi ve teşekkürlerimi sunuyorum...

İyi ki varsınız...

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

IP	Internet Protocol
IETF	Internet Engineering Task Force
IntServ	Integrated Services
DiffServ	Differentiated Services
QoS	Quality of Service
SOM	Self Organizing Map
RED	Random Early Detection
DT	Drop Tail
EWMA	Exponentially Weighted Moving Average
AIMD	Additive Increase Multiplicative Decrease
AQM	Active Queue Management
ERED	Enhanced RED
REM	Random Exponential Marking
RAQM	Rate-based Active Queue Management
PI	Proportional and Integral
FTP	File Transfer Protocol
HTTP	Hyper Text Transfer Protocol
OPNET	Optimized Network
UDP	User Datagram Protocol
RIP	Routing Information Protocol
OSPF	Open Shortest Path First
ICMP	Internet Control Message Protocol
WRED	Weighted RED
FIFO	First-In First-Out
BMU	Best Matching Unit
PQM	Passive Queue Management

1 INTRODUCTION

The evolution of the Internet has been continuing since its development in the early 1970s [1]. The data communication in the Internet is provided by the protocols that organize the functionality of network components such as routers, end users, transmission links. The name 'TCP/IP' represents the collective operation of Transmission Control Protocol (TCP) and Internet Protocol (IP). TCP [2] is a transport layer protocol which provides process-to-process communication between end user applications such as electronic mail and file transfer in the Internet. The data transfer service provided by TCP is connection oriented and reliable. IP is the network layer protocol of the Internet and its main considerations are routing and path determination for packets [3].

'Best effort' is the traditional service model of the Internet and is still being used for packet delivery. No special service guarantees are provided by the best effort service, on orderly and error free delivery of segments. Internet Engineering Task Force (IETF) working group has proposed architectures like Integrated Services (IntServ [4]) and Differentiated Services (DiffServ [5, 6]) for modifying Internet infrastructure to support real-time Quality of Service (QoS) requirements. In [7], Internet QoS is handled as a puzzle that contains many pieces such as terminology, architecture, traffic engineering and marketing. Another work, [8] is a review on the available mechanisms used to enable QoS guarantees in packet switched networks and to offer trade off among performance, functionality, complexity, etc. IP QoS is about providing different services for different users in the network layer of the Internet, while taking necessities of users and availability of network resources into account. In [9], three logical planes are defined as building blocks for QoS: control plane, data plane and management plane. Each plane contains a different set of mechanisms: admission control, QoS routing, resource reservation (control plane); buffer management, congestion avoidance, packet marking, queueing &

scheduling, traffic shaping, traffic policing, traffic classification (data plane); metering, service level aggreement, service restoration (management plane). This dissertation is mainly focused on the congestion avoidance mechanism in data plane.

Congestion occurs in the Internet when the demand for a resource, i.e. buffer space in routers or bandwidth of a link, exceeds the available capacity of the resource [10]. Packet losses, long end-to-end delays, delay variations (jitters) are unwanted results of sustained network overload. As defined in [3], end-to-end delay is the accumulation of transmission, processing and queueing delays in routers; propagation delays in the links and processing delays at end users. Propagation delay is affected by the distance between source and destination, while queueing delay varies with respect to the amount of traffic load at the routers [11]. As a result of variable queuing delays in the routers, IP datagrams experience different delays in travelling from source to destination and jitter occurs. Amounts of jitter, end-to-end delay and packet loss are so important for multimedia applications such as Internet phone and real-time video conferencing [11].

In TCP/IP networks, there are two main indicators for a congestion problem in the network: appearance of a timeout or receipt of triple duplicated acknowledgement for a packet previously sent by a TCP source [10,12-13]. The algorithms against congestion that are introduced in [12] and [13] are based on the adjustment of sending rate of a TCP source when it is notified of congestion. Congestion control and congestion avoidance are different approaches used for handling congestion problem. As described in [10], congestion control is a reactive approach that is applied to already congested networks while congestion avoidance is used as a prevention against congestion.

There are plenty of passive/active queue management schemes proposed to handle the congestion of recent IP infrastructure. In this dissertation, congestion avoidance is examined under the scope of queue management algorithms. The mostly cited studies and some of the recent work on queue

management are reviewed; their advantages/drawbacks are investigated. As a starting point, global congestion behaviour of IP networks with multiple bottlenecks is considered. A centralized observation method is presented to monitor global congestion behaviour of an IP network. An observation unit is designed to identify the congestion level of the network by collecting queue length data from all router interfaces. Collected data are used to train a Self Organizing Map (SOM), which is a special class of artificial neural networks. By the help of the trained SOM, it is possible to visualize global congestion and to make predictions on the future congestion behaviour of the network. The information obtained by the SOM is of concern to all IP routers in the network. A congestion notification traffic is supplied to the network by the observation unit, and by this way, routers are informed about the level and tendency of congestion.

New queue management approaches, which are proposed as a part of this dissertation, aim for the treatment of end-to-end delays/delay variations, by controlling the number of packets queued in router output interfaces. A new queue management technique is produced due to each new approach and then it is activated by router output interfaces. As a result, necessary updates are performed on the new queue management technique, upon receiving the congestion notification sent by the observation unit. Performances of new approaches are verified by the OPNET Modeler simulation program. Simulation results are compared with that of Drop Tail (DT) and RED algorithms. The following section presents the outline of this dissertation.

1.1 Dissertation Outline

This dissertation is organized as follows:

Chapter 2 is a motivation for the readers of this dissertation. Principles of passive and active queue management are explained by referencing to well known algorithms such as Drop Tail and RED.

An object-oriented simulation program, OPNET Modeler, is described in Chapter 3. Steps of building an IP network and producing network components (such as routers, links, end users, etc.) with desired functionality are summarized in this chapter. Global congestion problem in multi-bottleneck IP networks, queue management algorithms for congestion avoidance and SOM training are important considerations of this dissertation. Main problem statement, utilization of SOMs in monitoring tendency of global congestion, definition of some parameters used in statistical data analysis and development of new queue management approaches are presented in Chapter 3. Details about simulations, which are carried out by the OPNET Modeler program for investigating the performances of new queue management approaches, are given and results are discussed.

Last chapter presents concluding remarks and topics of ongoing studies.

2 QUEUE MANAGEMENT ALGORITHMS

Congestion avoidance mechanisms presented in [12] are based on reducing the amount of traffic generated by TCP sources during congestion. As a result of the growth in the Internet, together with the increase in the number of Internet users and applications such as ftp, http, e-mail, video, etc., the effect of TCP congestion avoidance mechanisms had to be strengthened by queue management algorithms in IP routers. Queue management algorithms control the number of packets in queues of router interfaces and decides whether to drop/mark packets or not. There are two main classes of queue management algorithms which are explained in the following sections.

2.1 Passive Queue Management

Passive Queue Management (PQM) is the traditional technique that supports acceptance of packets until router buffer limit is reached. It does not employ any preventive early packet drops [11]. DT is the most widely used PQM technique. It lets the router queue accept packets until getting full and then newcomers are dropped till some empty place is gained with transmissions. It has been widely used although having important drawbacks. The problems about DT are lock-out and full queue behaviours. Lock-out is the situation that appears when the router queue is fully used by packets of only a limited number of connections, so network resources are not shared fairly. Full queue problem causes long queueing delays. 'Random drop on full' and 'drop front on full' are other PQM techniques which are effective for lock-out problem, but full queue situation is still a problem [14].

2.2 Active Queue Management

Active Queue Management (AQM) is a preventive queue management technique that detects congestion early and informs end users to decrease the

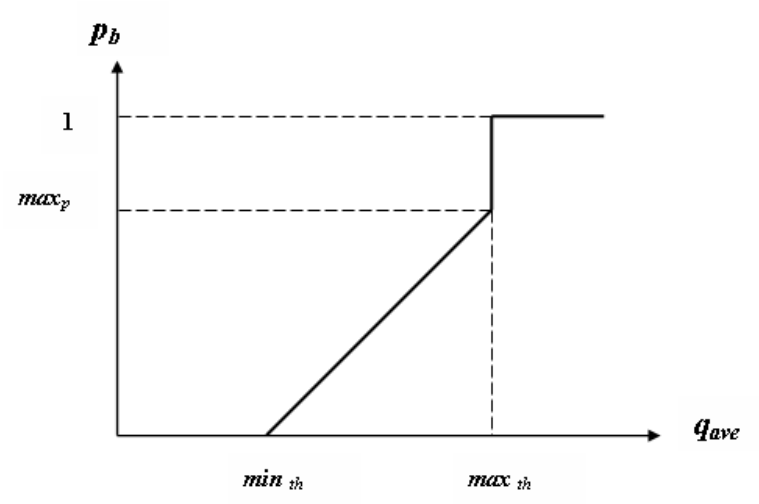

Figure 2.1: Probability p_b vs average queue length in RED

amount of traffic, for preventing packet losses due to buffer overflow [15]. As a result of the need to improve queue management and congestion avoidance in the Internet, [14] is presented. In [14], Internet community is suggested to use RED algorithm [16] for providing AQM in routers. In [16], where congestion avoidance ability of random early detection was proposed, packet switched networks are taken into consideration. The design goals studied in [16] are as follows:

- congestion avoidance by controlling average queue length and maintaining an upper bound on the average queue length,
- avoidance of biases against bursty traffic.

In RED, marking probability for the new arrival, p_a, is calculated as follows:

$$p_a = p_b/(1 - count.p_b) \quad , \tag{2.1}$$

where p_b is the marking probability and *count* is the number of arrivals since the last marking. The variation of p_b with respect to the average queue length, q_{ave}, is shown in Fig.2.1, whose partial expression is given in (2.2):

$$p_b = \begin{cases} 0, & q_{ave} < min_{th} \\ max_p \left[\dfrac{q_{ave} - min_{th}}{max_{th} - min_{th}} \right], & min_{th} \leq q_{ave} < maxth \\ 1, & max_{th} \geq q_{ave} \end{cases} \quad . \tag{2.2}$$

The value of q_{ave} is updated at packet arrival times by the following equation:

$$q_{ave} = q_{ave_p} (1 - w_q) + w_q q \quad , \tag{2.3}$$

where q_{ave_p} is the previous value of the average queue length, q is the current value of the queue length and w_q is the queue weight. (2.3) is a low pass filter equation, known as Exponentially Weighted Moving Average (EWMA). The decision on the values of min_{th}, max_{th} and w_q is an important problem. A few rules are given in [16] for the solution of this problem:

- For efficiency of implementation, negative power of two, preferably a value greater than 0.001 should be used for w_q. If it is so low, a delay occurs in following the increases in the instantaneous queue length,
- bursty nature of networks should be taken into consideration, average queue length should be kept high enough for preventing underutilization of the output link,
- the probabilistic behaviour of packet marking is a precaution against reduction of transmission rates at once by multiple hosts in response to packet marking. Additionally, $[min_{th} \;\; max_{th}]$ interval should be long enough to avoid global synchronization. $\max_{th} > 2 \min_{th}$ should be satisfied.

In [17], design guidelines are presented for providing stable operation of a linear feedback control system which is represented by a combined TCP and AQM model implementing RED. Besides its benefits in fairness and avoiding global synchronization, RED suffers from low throughput, fluctuations in the queue length and large queueing delay variation. These disadvantages of

RED has attracted attention of many network researchers and various RED variants are proposed. Some of these variants and examples to AQM algorithms are explained below:

Self configuring RED [18] is about improving RED, by following an adaptive algorithm for setting RED parameter max_p due to the characteristics of network traffic:

$$\begin{gathered}
\text{if } (min_{th} < q_{ave} < max_{th}) \\
status = Between; \\
\text{if } (q_{ave} < min_{th} \text{ and } status! = Below) \\
status = Below; \\
max_p = \frac{max_p}{\alpha} \\
\text{if } (q_{ave} > max_{th} \text{ and } status! = Above) \\
status = Above; \\
max_p = max_p * \beta;
\end{gathered}$$

As seen in the above algorithm, average queue length is observed, if it is oscillating around min_{th}, then the drop mechanism is said to be too agressive; if it is oscillating around max_{th}, then it is too conservative. Simulation results presented in [18], show that the adaptive RED algorithm reduces packet losses and improves link utilization.

In another adaptive queue management technique proposed in [19], maximum drop probability is updated dynamically while average queue length is kept close to a desired level.

The design purpose for NRED [20] is truely estimating the number of active flows and keeping the queue length close to a target value. NRED aims to stabilize router queue length and proposes an estimator for active flow numbers together with an algorithm for updating the value of max_p. If the estimated number of active flows, N, is unchanged, max_p is updated by the following equation:

$$max_p(new) = max_p(old) + \frac{2 * max_p(old)}{max_{th} - min_{th}} * (q(new) - T) \quad , \tag{2.4}$$

where $q(new)$ is the instantaneous queue length and T is the target queue length (T value is taken as $\frac{max_{th} - min_{th}}{2}$ in simulations). If N is changed, then $max_p(new)$ is calculated by the following equation:

$$max_p(new) = 2 * \left\{ 1 + \frac{R^2C^2}{4N^2} - \sqrt{\frac{R^2C^2}{4N^2} + \frac{R^4C^4}{16N^4}} \right\} \quad , \tag{2.5}$$

where R is the round trip time and C is the link capacity.

TL-RED [21] is a traffic load adaptive algorithm, such that max_p value is varied due to changes in traffic load. The algorithm, which is given below, is used to stabilize the queue length in various traffic conditions, by means of changing maximum drop probability as a result of changing load conditions:

$$\text{traffic}_{ave}=(1\text{-w}_t)*\text{traffic}_{ave}+\text{w}_t*\text{traffic}$$

When $\text{traffic}_{ave} < \text{T}_{min}$ or $q_{ave} < min_{th}$: $max_p = max_p - \alpha * (\text{past}/\text{traffic}_{ave})^2$

When $\text{traffic}_{ave} > \text{T}_{max}$ or $q_{ave} > max_{th}$: $max_p = max_p + \beta * (\text{traffic}_{ave}/past)^2$

$$\text{past}=\text{traffic}_{ave}$$

where

$past$: previous value of average traffic load

traffic: instantaneous value of the traffic load

traffic_{ave} : average value of the traffic load

w_t : weight, $\text{w}_t = 0.002$

$\alpha = 0.00025$ and $\beta = 0.0025$

T_{min}, T_{max} : boundaries indicating whether traffic load is light or heavy.

In [22], maximum drop probability is changed with respect to the performance variations. Performance is a measure of ability for keeping instantaneous queue length between minimum and maximum threshold values.

A load adaptive queue management scheme is presented in [23]. By the usage of this scheme, queue thresholds (and packet drop probabilities) are varied dynamically as the network load changes, while packet loss rate is kept close to a target value.

Calculation of the average queue length is an important process since RED uses this value as a measure of congestion. Alternative calculation methods are presented in [24-26]:

[24] is a recent study on assigning optimal weight values, w_q, to RED. A RED variant, named fuzzy RED, is proposed in [25]. In fuzzy RED scheme, EWMA procedure is different from RED. An estimate of the queue length is obtained by using a fuzzy EWMA and the amount of error, between real and estimated values, is used to calculate new weight, w_q. Stability of the queue length and robustness against variations in the network load are two goals of [25]. Fuzzy RED scheme improves RED performance in packet loss rate, average queueing delay and link utilization. Enhanced RED (ERED) is introduced in [26]. For calculating average queue length, the most appropriate equation is chosen among the equations produced for different queue length intervals.

Each of the studies [27-37] presents a formulation for determining packet drop/mark probability:

In DSRED ([27], [28]), average queue length (q_{ave}) is a factor in determining drop probability (p_d) and it is calculated in the same way as in RED. However, unlike in RED, a piecewise linear model is used to determine p_d. The graphics for p_d vs q_{ave} is shown in Fig.2.2 and the dropping behaviour is expressed by the following equations [27]:

$$p_d(q_{ave}) = \begin{cases} 0, & q_{ave} < K_l \\ \alpha(q_{ave} - K_l), & K_l \leq q_{ave} < K_m \\ 1 - \gamma + \beta(q_{ave} - K_m), & K_m \leq q_{ave} < K_h \\ 1, & K_h \leq q_{ave} \leq N \end{cases} , \tag{2.6}$$

where

$$\alpha = \frac{2(1-\gamma)}{K_h - K_l} , \tag{2.7}$$

$$\beta = \frac{2\gamma}{K_h - K_l} . \tag{2.8}$$

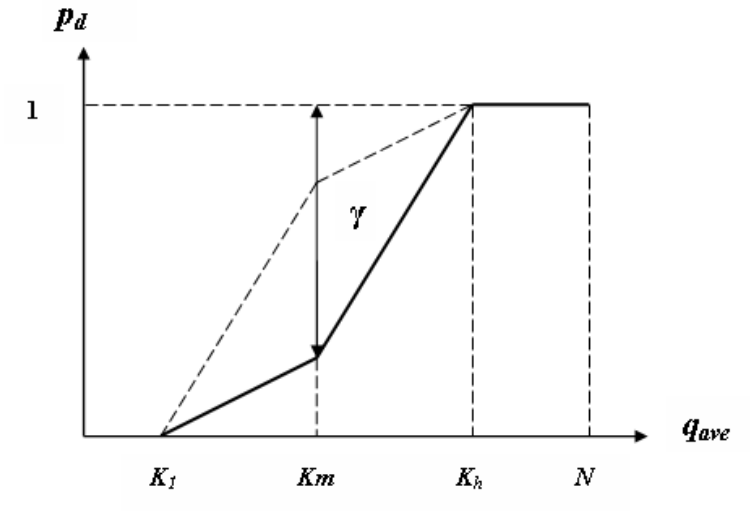

Figure 2.2: Packet drop probability vs average queue length in DSRED

γ gives the rate of change in packet drop for the case $K_m \leq q_{ave} < K_h$.

Simulation results in [27] show that DSRED decreases packet drop rate and average queueing delay while improving throughput.

SRED [29], is presented for stabilizing queue length. Unlike in RED, average queue length is of no interest. The packet drop probability depends on the instantaneous queue length and the estimated number of active flows.

DRED, which is described in [15] and [30], calculates packet drop probability in a different manner: samples of instantaneous queue length, q_n, is taken every Δt seconds and drop probability, $p_d(n)$ is calculated by following the procedure below:

- Computation of the error signal by using the target queue length value, $T(n)$:

$$e(n) = q(n) - T(n) \quad , \tag{2.9}$$

- Computation of the filtered error signal:

$$\hat{e}(n) = (1 - \beta)\hat{e}(n-1) + \beta e(n) \quad , \tag{2.10}$$

  where β is the filter gain,

- Computation of the current drop probability:

$$p_d(n) = min\left\{max\left[p_d(n-1) + \alpha\frac{\hat{e}(n)}{B}, 0\right], \theta\right\} \quad , \tag{2.11}$$

where α is the control gain, θ is the upper bound on the drop probability, which is less than or equal to 1, and B is the buffer size.

Simulation results show that DRED is able to keep the queue length close to target value, preventing overflows and underflows.

BLUE [31] may be defined by the following algorithm:

For a packet loss (or $queuelength > L$) event:

$$\begin{gathered}\text{if } ((now - last\_update) > freeze\_time) \\ p_m := p_m + \delta_1 \\ last\_update := now \quad ,\end{gathered}$$

For a link idle event:

$$\begin{gathered}\text{if } ((now - last\_update) > freeze\_time) \\ p_m := p_m - \delta_2 \\ last\_update := now \quad ,\end{gathered}$$

where

L: a limiting value for the queue length

p_m: packet marking probability

$freeze\_time$: minimum time interval between two successive updates

$last\_update$: time of last update for p_m

δ_1: amount of increase in marking probability

δ_2: amount of decrease in marking probability

Simulations show that when the queue management is changed from RED to BLUE, throughput increases, packet loss decreases, an improvement is observed in link utilization.

PRED, which is proposed in [32], is another RED variant that aims to achieve higher throughput and lower average delay. The algorithm, as summarized below, updates the value of packet drop probability based on the instantaneous queue length and the threshold values:

$$\text{If } q_{ave<min_{th}} \Rightarrow \text{enqueue all packets}$$

$$\text{If } q_{ave} > \frac{max_{th} + q}{2} \Rightarrow \text{drop all packets}$$

$$\text{If } min_{th} \leq q_{ave} < \frac{max_{th} + q}{2} \Rightarrow \text{apply the progressive adjustment method to find the drop probability.}$$

Another AQM method is introduced in [33], where drop probability is adjusted with respect to the average queue length and the estimated packet arrival rate.

A method for estimating future congestion level and calculation of drop probability is presented in [34]. The purpose of the work is to achieve higher link utilization and queue length stability.

Random Exponential Marking (REM), which is explained in [35], is an AQM techique which uses link prices as a congestion measure. Its name comes from the behaviour against arrival packets: a packet is marked with an exponentially increasing probability as link price increases.

A queue management algorithm away from the probabilistic behaviour of packet dropping is proposed in [36]. A virtual buffer capacity, less than the link capacity, is introduced for a queue, and an arriving packet is marked or dropped when the buffer is full. At each packet arrival, virtual buffer capacity is updated with respect to the arrival rate.

In [37], the importance of packet dropping behaviour (as a linear, convex, concave, step function of queue length), in achieving AQM stability, is studied.

There are comparative studies, such as [38], in which simulation results for a number of AQM algorithms are presented.

Robust control principles are considered in some AQM algorithms. The studies [39] and [40] are on designing controllers by using Proportional (P) control and Proportional-Integral (PI) control techniques where stability of the queue length is considered. A recent study, [41], is about designing rate-based PI controllers. In [42], a nonlinear model of TCP/RED is presented, then the stability region for TCP/RED is obtained in terms of round trip propagation delay and bottleneck link capacity. In [43], a Proportional-Derivative (PD) control methodology is followed to produce a new AQM scheme. Another robust AQM design, satisfying robust stability under uncertain network conditions, is presented in [44]. A PI-PD controller is proposed in [45] to provide proactive congestion avoidance in the Internet. A formula (based on input rate of the queue, capacity of the link, instantaneous queue length and the target length) for determining the packet drop rate is proposed in [46]. RAQM, which is described in [47], is a rate-based AQM scheme, which updates its packet drop probability with respect to either the input traffic rate or the instantaneous queue length (for regulation of these values to the expected ones). In [48], a robust H^{∞} controller is developped and its advantages over RED and PI [39] schemes are presented.

3 A STUDY ON CONGESTION NOTIFICATION AND AQM

The performances of most queue management schemes in Chapter 2 were tested by researchers on single bottleneck networks of dumbbell topology. In Fig.3.1, a representation of dumbbell topology is shown. As seen in the figure, R_1 and R_2 are routers, s_i and d_i are source and destination pairs, ($i = 1, 2, ..., n$. n is the total number of pairs). The traffic flow (ftp, http, video, etc.) is directed from sources to destinations except the acknowledgement packets' flow from destinations to sources. Queue lengths, packet losses, end-to-end delays, delay variations, link utilizations and throughputs are observed during simulations to test the performance of proposed queue management algorithm and compare it with other algorithms. The reason for choosing this topology is its simplicity in studying congestion behaviour of single bottleneck networks.

In this dissertation, the congestion behaviour in multi-bottleneck networks is studied and some improvements are proposed for AQM. It is well-known that congestion problems arise at bottlenecks as a result of heavy traffic load and limitation in link/queue capacities. For observing the congestion

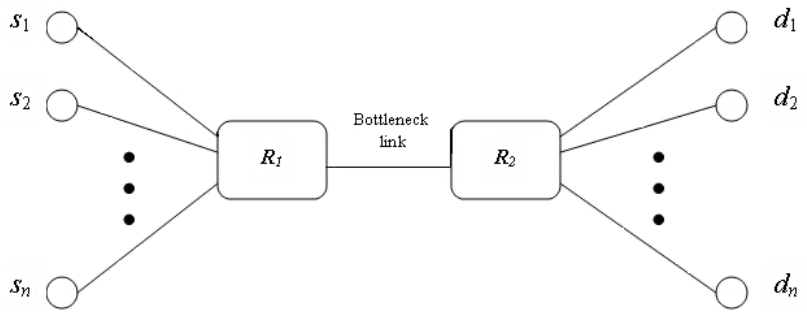

Figure 3.1: Dumbbell network topology

behaviour of bottlenecks all over the network, a centralized observation unit is designed as a part of this dissertation. Since queue lengths at the bottlenecks are indicators for congestion, their values are collected by the observation unit. These data are then used to train a Self Organizing Map (SOM). After training, the resulting map is embedded in the observation unit so that the future behaviour of congestion could be predicted by observing current state. Then, due to predicted congestion state of the network, AQM schemes applied to routers (and to bottleneck queues) are improved. The observation unit sends reply packets to routers for congestion notification. Upon receiving the notification, each router updates its AQM parameters. Details of this procedure and modeling is given in Section 3.2. Simulation results are also discussed in the same section.

3.1 A Network Simulation Tool: OPNET (OPtimized NETwork) Modeler

Network simulation tools, such as programs named Network Simulator (NS) and Optimized Network (OPNET) Modeler, provide an environment for researchers to design networks by using elements with specialized functionalities, develop new algorithms and test their performances. OPNET Modeler is one of the most commonly used tools in network design and simulation. In this section, OPNET Modeler is going to be explained, by the help of a network model which is used during simulations.

In this dissertation,

- a commonly used IP router model is improved,
- an observation unit is generated,
- operation of the observation unit is simulated on a network with improved routers.

As explained above, an IP router model is generated by improving the router model 'ethernet4_slip8_gtwy_adv' in OPNET. Firstly, let's study the properties of an ethernet4_slip8_gtwy_adv router:

- It is an IP based gateway, supporting 4 Ethernet hub interfaces (for 10BaseT/100BaseT connections) and 8 serial line interfaces (for IP connections).
- IP, UDP, Ethernet, RIP, OSPF, SLIP are supported protocols.
- RIP (Routing Information Protocol) or OSPF (Open Shortest Path First) protocol may be used to dynamically and automatically create its routing tables and select routes in an adaptive manner.
- It can not be used as a source or a final destination node.

In addition to having the first three properties from the list above, improved router model should have the ability to produce special IP packets when needed and to receive the corresponding reply messages. This improvement is necessary for collecting the queue length data of all interfaces in an observation unit and monitoring the network congestion level.

Fig.3.2 is the project model of a network which is produced by using the *Project Editor* of OPNET Modeler. Data sources, routers, links and any other components of a network are easily selected from the 'Object Palette' of the *Project Editor* and the network is built. Statistics, that will be collected during simulation, are chosen and simulation is started. If a new network component (link, node, packet,...) is to be generated, then the associated *Editor* (*Link Model Editor*, *Node Editor*, *Packet Format Editor*, ...) is activated. The node model of an IP router is given in Fig.3.3. There are modules - named 'ip', 'ip_encap', etc... - and wires with arrows showing the direction of packet stream, logical association or message transmission. Each module is designed for a different purpose and has a different process model.

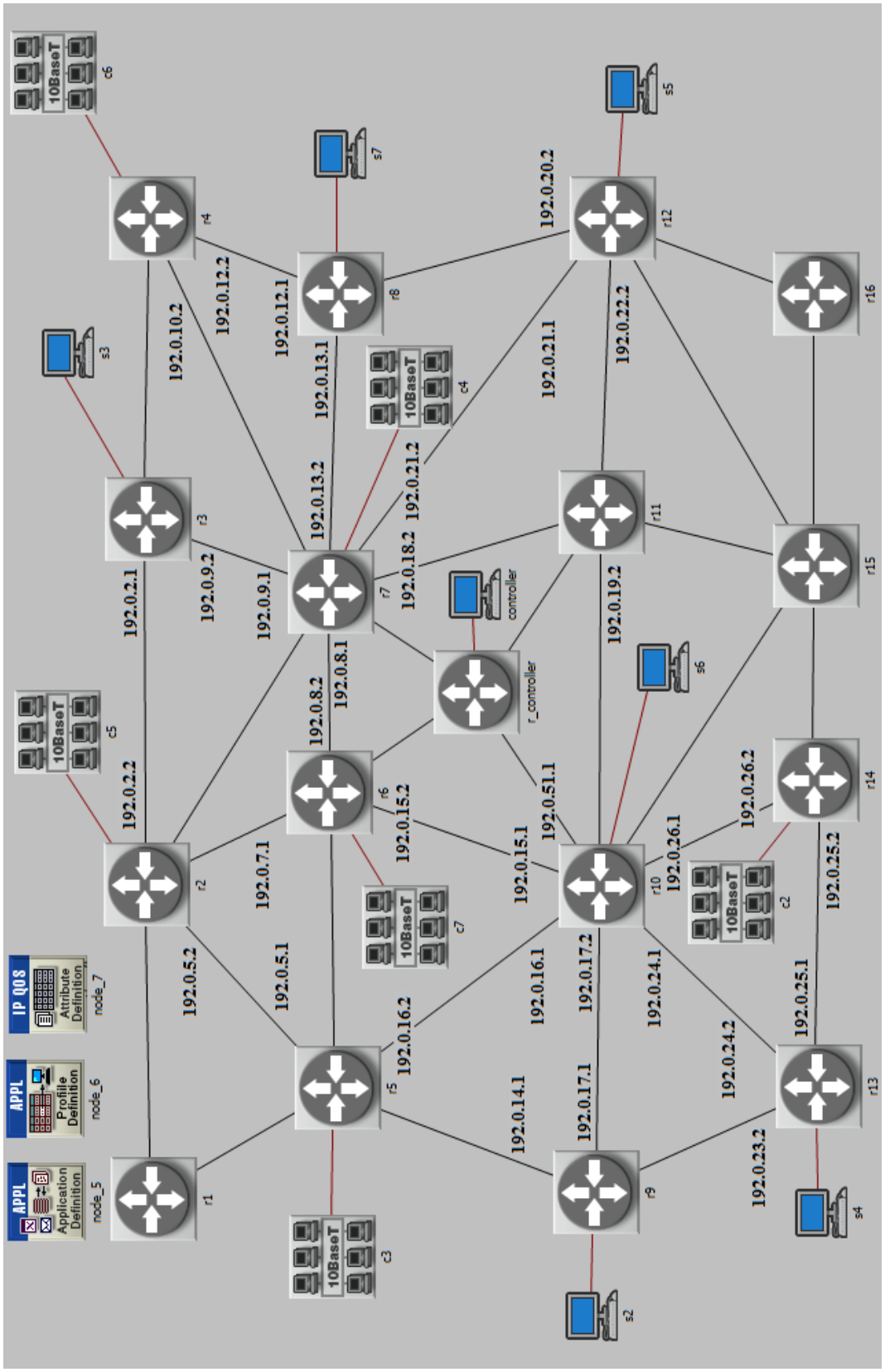

Figure 3.2: Network model designed for Approach 1

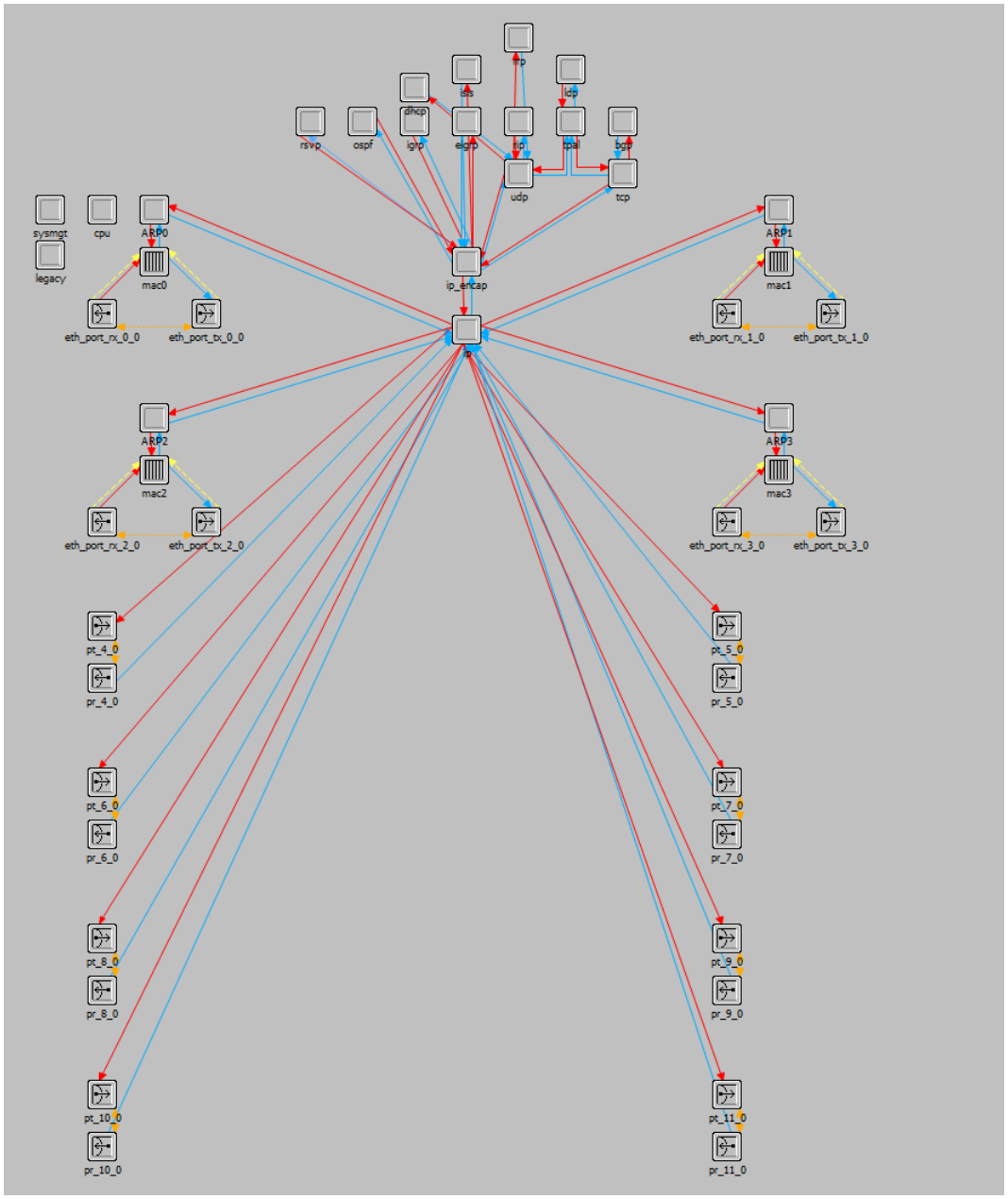

Figure 3.3: Node model of an IP router

Process models are handled in the *Process Editor*. A process model is expressed in 'Proto-C' language. Proto-C is based on a combination of state transition diagrams, a library of high-level commands known as 'Kernel Procedures' and the C or C++ programming language. Process model for an 'ip' module is shown in Fig.3.4. This model is an improved version of the standard model given in OPNET. As seen in Fig.3.4, states are represented by circles. States may contain a code that is performed immediately after the state is entered (Enter Executives) or just before the state is left (Exit Executives). States in color 'red' are unforced states; an unforced state is blocked immediately after executing the Enter executives and waits for an interrupt before executing the Exit Executives and leaving the state. States in color 'green' are named 'forced' states [49]. Enter Executives for newstate1 and newstate2 are given in Fig.3.5 and Fig.3.6, respectively.

Process models may have 'children'. Children are invoked during the execution of parent processes. The process model, whose state transition diagram is seen in Fig.3.4, has child processes and two of them are studied in Fig.3.7 and Fig.3.8.

When an IP packet reaches the router, ip module is activated and the child process 'ip_rte_central_cpu' is invoked. Another child process, named 'ip_output_iface', is invoked when a packet is passed from the 'ip_rte_central_cpu' to be queued in one of the output interfaces. The queue lengths may be observed in 'ip_output_iface', and the decision for packet dropping is also made here. Before leaving the router, packets are again handled by the parent process 'ip_dispatch'. By the help of this information, the router model could be modified. New process models named 'my_ip_dispatch.pr.m', 'ip_rte_central_cpu_mart11.pr.m', 'ip_output_iface_h2dc1_wl_minth.pr.m' are produced by modifying the process models in OPNET Modeler: 'ip_dispatch.pr.m', 'ip_rte_central_cpu.pr.m', 'ip_output_iface.pr.m'. Steps of modification are explained below:

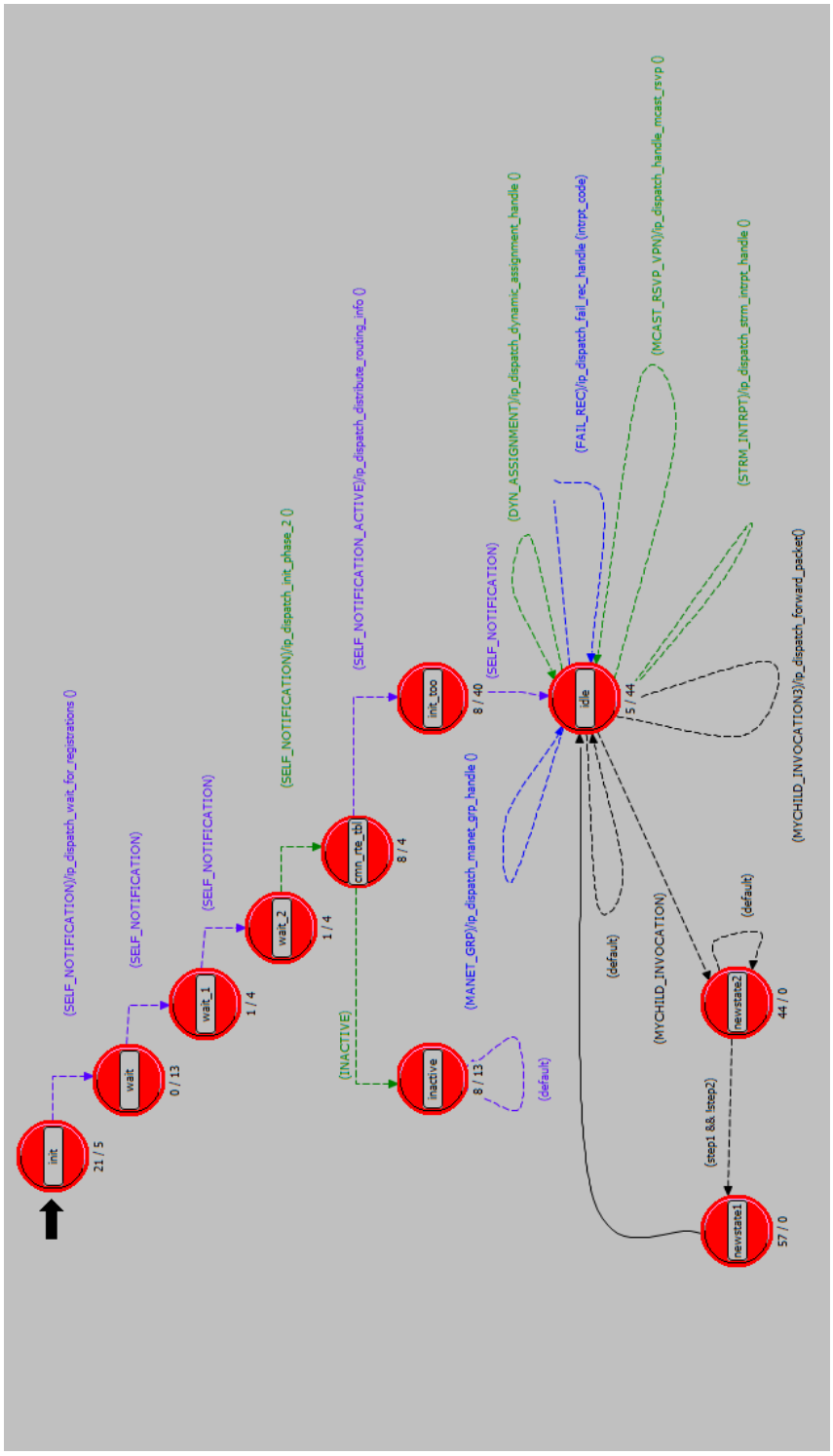

Figure 3.4: Process model for ip module

Step 1: Code is added to the function, named 'enqueue_packet', which is defined in 'ip_output_iface.pr.m', for capturing queue length values and related interface adddresses.

Step 2: A child process for 'ip_output_iface_h2dc1_wl_minth.pr.m' is generated. This child will transfer the interface address and the corresponding queue length values to 'my_ip_dispatch.pr.m'.

Step 3: The state transition diagram of 'ip_dispatch.pr.m' is enlarged and additional code is written (Fig.3.4 shows the enlarged model with new states: newstate1 and newstate2) to produce 'my_ip_dispatch.pr.m'.

Step 4: A new packet format is generated. In doing this, Internet Control Message Protocol (ICMP, [50]) is studied. ICMP is an upper layer protocol for IP, and is used for communicating information between hosts, routers, etc. Examples to ICMP applications are sending 'ping' packets (called 'echo request' and 'echo reply') between hosts and sending error messages like 'Destination is unreachable' from routers to source hosts. In OPNET Modeler, the child process, named 'ip_icmp', is responsible for a limited ICMP application (only echo request\echo reply is modeled). Fig.3.9 and Fig.3.10 show the structures of a ping packet and an IP packet which has a field for encapsulation of a ping packet, respectively.

The packet model generated in Step 4 is an icmp-like packet, which is carried in an IP packet. In our network model, queue length and interface info will be transferred to the observation unit, therefore a new ping-packet like model is defined in the *Packet Format Editor*. The new packet model is seen in Fig.3.11. In this model,

- iface_info (8 bits) field carries the IP address code of the interface,
- queue length (10 bits) field carries the queue length value of the output interface whose IP address code is written in the 'iface_info' field of the packet,
- type (8 bits) field carries the name 'IpC_Icmp_Echo_Request' which in-

dicates that a reply is expected from the destination.

```
intrpt_type = op_intrpt_type ();
intrpt_code = op_intrpt_code ();

invoke_prohandle = op_pro_invoker (module_data.ip_root_prohandle, &invoke_mode);

if ((invoke_mode != OPC_PROINV_INDIRECT) && (invoke_mode != OPC_PROINV_DIRECT))
    {
    ip_dispatch_error ("Unable to determine if how IP process got invoked.");
    }


if (intrpt_code==1313)
    {
    step1=1;
    step2=0;
    }

else
    {
    if (intrpt_code==1314)
        {
        step1=0;
        step2=1;
        }
    else
        {
        if (intrpt_code==0)
            {
            step1=0;
            step2=0;
            }
        else
            {
            step1=1;
            step2=1;
            }
        }
    }
```

Figure 3.5: Enter executives for state 'newstate1'

```
Prohandle my_icmp_call_handle2;
Objid                   node_objid              = OPC_OBJID_INVALID;
Packet*         qisend_pkptr;

kuyrukum_ptr = op_pro_argmem_access ();

uzunluk = kuyrukum_ptr->myqueuelength;

arayuzadi = kuyrukum_ptr->myinterface;

qisend_pkptr = op_pk_create_fmt ("my_identifier4new_h2");

op_pk_nfd_set (qisend_pkptr, "type", IpC_Icmp_Echo_Request);

my_objid = op_id_self ();

op_pk_nfd_set (qisend_pkptr, "queue length", uzunluk);

op_pk_nfd_set (qisend_pkptr, "iface_info", arayuzadi);

op_pk_nfd_set (qisend_pkptr, "source module objid", (double) my_objid);

my_ip_dgram_pkptr = my_ip_icmp_pkt_encapsulate (qisend_pkptr);

if (my_ip_dgram_pkptr != OPC_NIL)
    {
    op_pro_invoke (routing_prohandle, my_ip_dgram_pkptr);
    }

module_data.ip_ptc_mem.child_pkptr=my_ip_dgram_pkptr;
```

Figure 3.6: Enter executives for state 'newstate2'

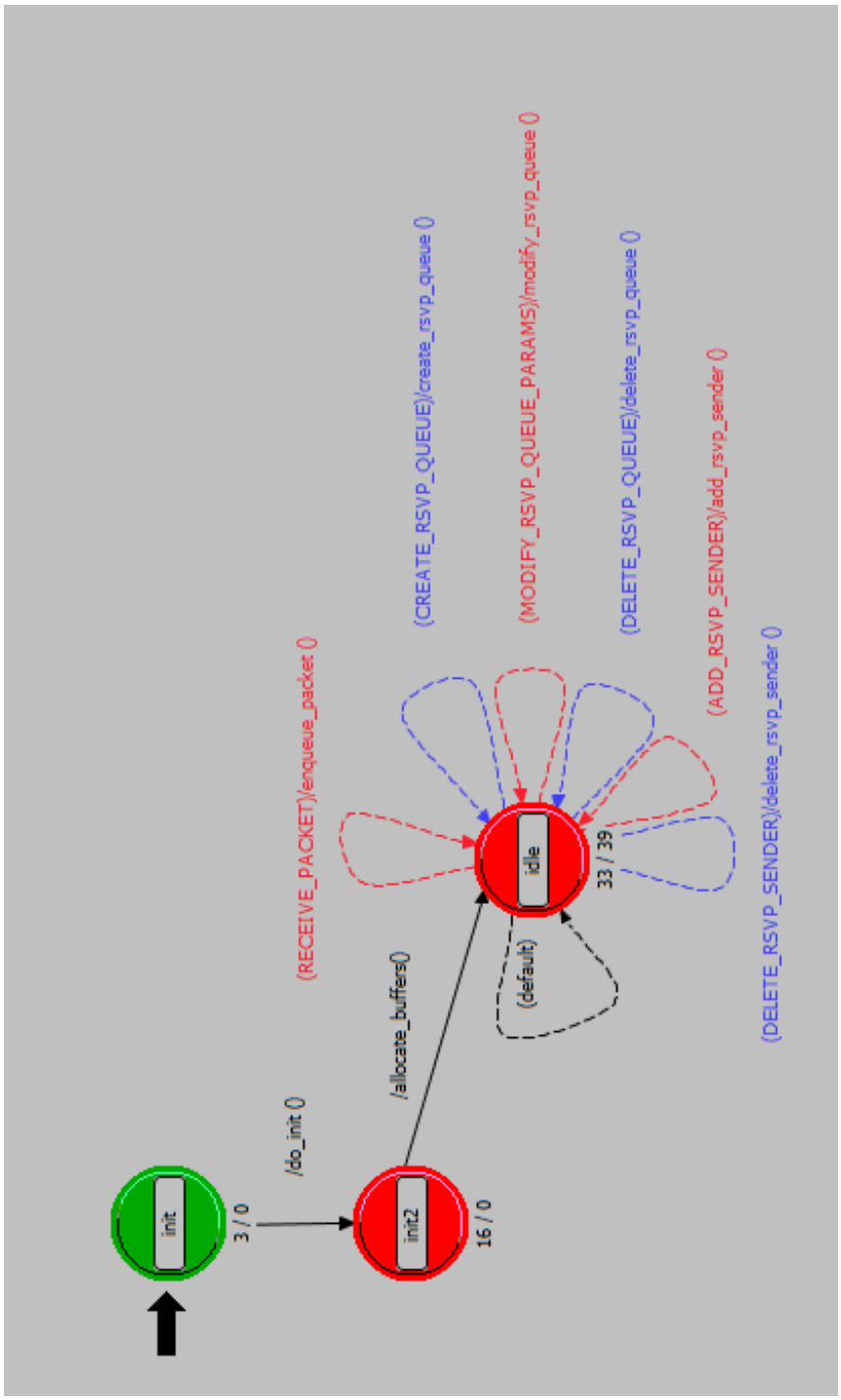

Figure 3.7: Process model 'ip_output_iface_h2dc1_wl_minth.pr.m'

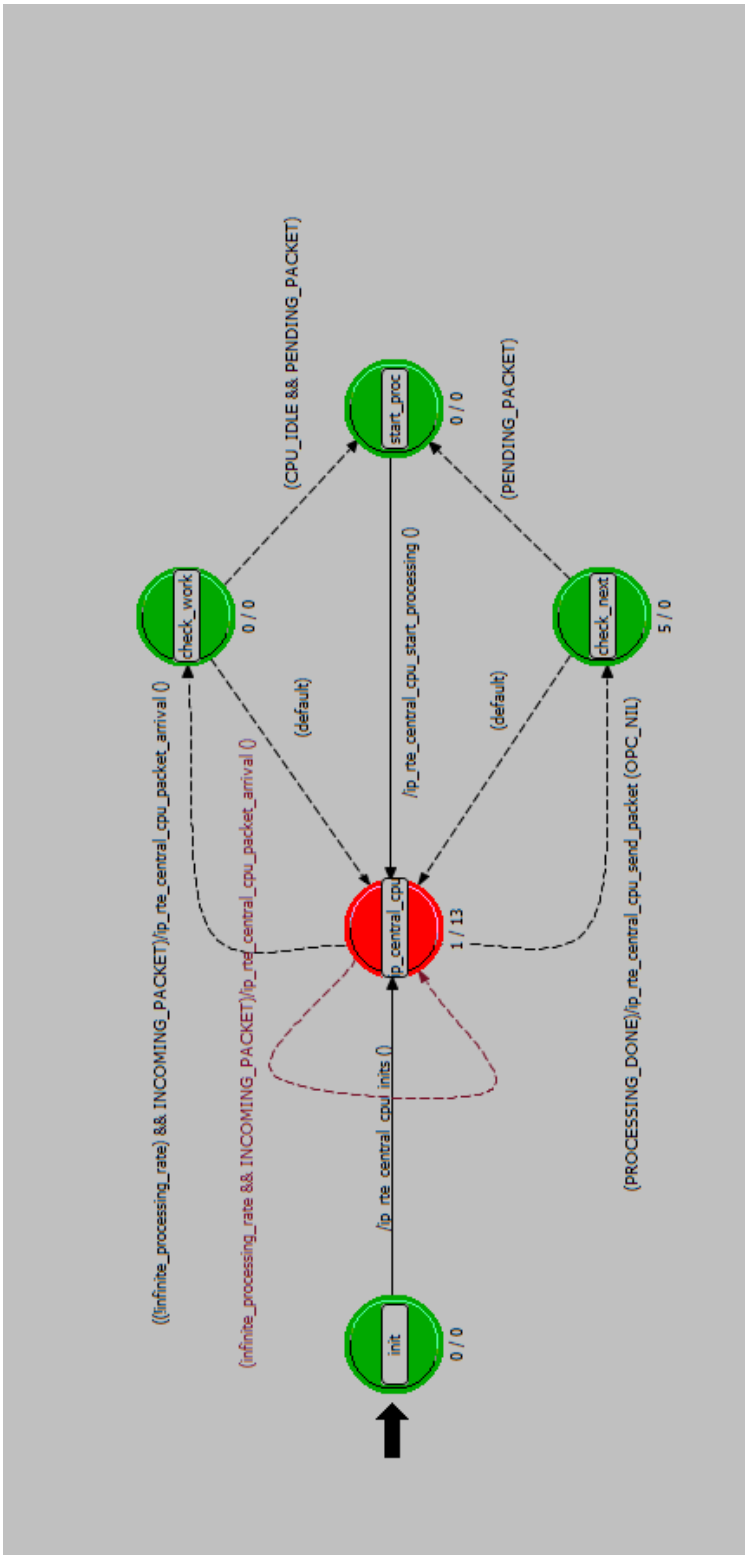

Figure 3.8: Process model 'ip_rte_central_cpu_mart11.pr.m'

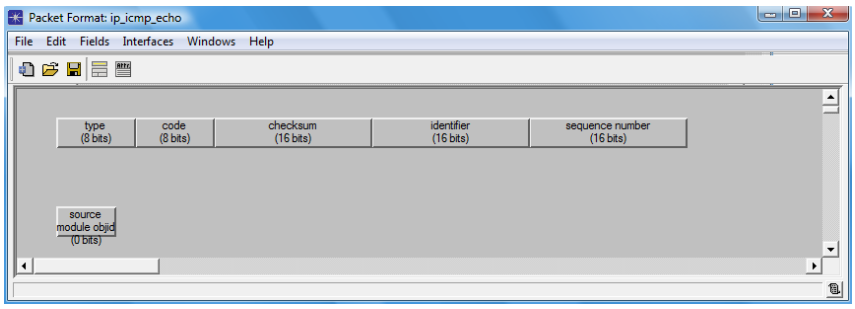

Figure 3.9: Packet model of a ping packet

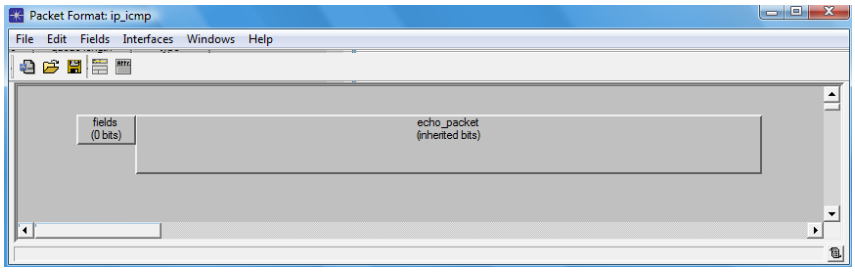

Figure 3.10: Packet model of an IP packet carrying ping packets

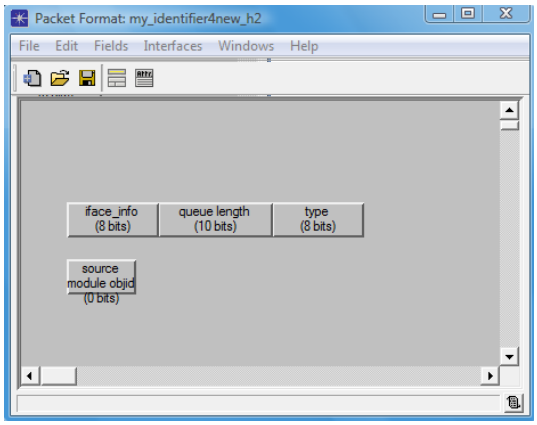

Figure 3.11: Packet model 'my_identifier4new_h2.pk.m'

The centralized observation unit, which is responsible for monitoring global congestion level, is built by making some changes on a commonly used ethernet end user model in OPNET: 'ethernet_wkstn_adv'. The changes could be summarized as follows:

- When a packet is received, the observation unit detects the data field of the packet and follows a special procedure if the packet is of type 'IpC_Icmp_Echo_Request' and carrying an interface information (queue length and IP address code)
- If the mission of the observation unit is only collecting interface information, then queue length and IP address code are written into a file together with the receipt time (the time when the packet is received).
- If the observation unit is used to determine the congestion level of the network by observing the incoming information and send a reply back to the router for avoiding congestion, then the code needs to be a bit more complicated.

The node model for the centralized observation unit is seen in Fig.3.12. 'controller_ip_dispatch.pr.m' is the process model of the ip module of the observation unit and it is seen in Fig.3.13.

When the type of the incoming packet is IpC_Icmp_Echo_Request, it is forwarded to a child process whose process model is given in Fig.3.14. If the observation unit is to be used for congestion avoidance, it has to sense the congestion level of the network and send a reply (a congestion notification) back to the router. Contents of the fields in the request packet are changed to produce the reply packet. The content of the 'type' field in the packet model is changed: IpC_Icmp_Echo_Request is replaced with IpC_Icmp_Echo_Reply. In addition to this replacement, there is a change in the 'queue length' field of the packet: this field is used to write the value of congestion level indicator, 'iface_info' field of the packet remains unchanged.

Now, it is time to make another improvement in the IP router:

- The congestion indicator should be used only by the corresponding router (which has sent the IpC_Icmp_Echo_Request).
- When an IpC_Icmp_Echo_Reply is received by a router, a special 'interrupt' signal is generated in the process so that the packet content is handled in a different way than the contents of other ip packets.
- The value in the iface_info field is compared with the output interface addresses of the router; if it is not one of these addresses, congestion indicator is ignored and the packet continues its way.
- If the congestion indicator is accepted, router uses this value to update its RED parameters.

In Fig.3.2, client side of the traffic is represented by '10BaseT LAN's that are ethernet local area networks, with 10 clients and a server, in switched topology ('eth_switched_lan_adv.nd.m' is the node model for clients). Servers use the node model 'ethernet_wkstn_adv.nd.m', which is mainly used for client-server applications running over TCP/IP and UDP/IP. Clients and servers are using TCP Reno, as transport layer protocol. Each client, each server and the observation unit is connected to its edge router by a '10BaseT' link. The '10BaseT' is a duplex link which represents an ethernet connection operating at 10 Mbps. The links connecting routers are duplex 'PPP_E1' links which operate at 2.048 Mbps. Fig.3.15 shows 'PPP_E1' definition in the *Link Editor*. In Fig.3.2, there are 3 configuration objects for attribute definition, profile definition and IP QoS configuration. The type of the traffic (video conferencing), packet size, packet interarrival time are defined in the attribute definition object; while the settings like traffic start and stop times, repeatibility of the traffic profile are made in profile definition object. IP QoS configuration object is used for QoS requirements (link-scheduling disciplines: FIFO, priority queuing, etc. and AQM algorithms: RED,WRED,...) on the output interfaces of the routers. The routing protocol used in the routers is 'RIP'. RIP uses a

distance vector algorithm, called 'Bellman-Ford algoritm'. The properties of this algorithm may be summarized as follows [3]:

- It is a distance vector algorithm. It provides routers the information about the cost of directly attached links, the cost of the least-cost path and the knowledge received from directly connected neighbours.
- It is load-insensitive, link costs does not change with the variations in the congestion level.
- Routing updates are exchanged between the routers approximately every 30 seconds by using a 'RIP response message'.
- RIP messages are sent over UDP in a standard IP packet.

In this section, a brief information is given about OPNET Modeler and one of the network models used in simulations. The following section contains details about congestion avoidance and proposes new queue management approaches.

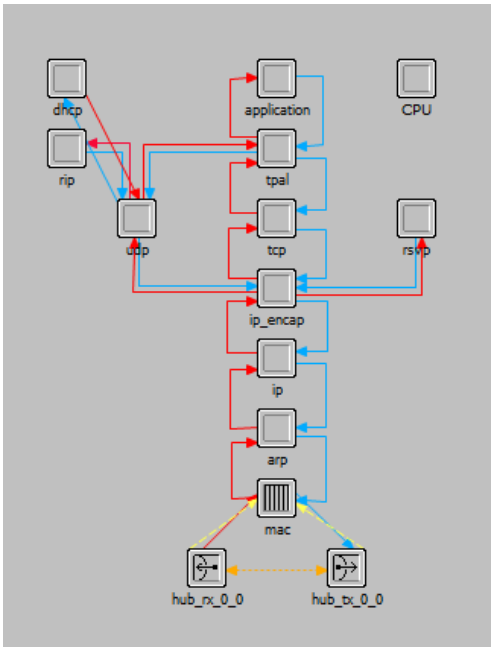

Figure 3.12: Node model of the centralized observation unit

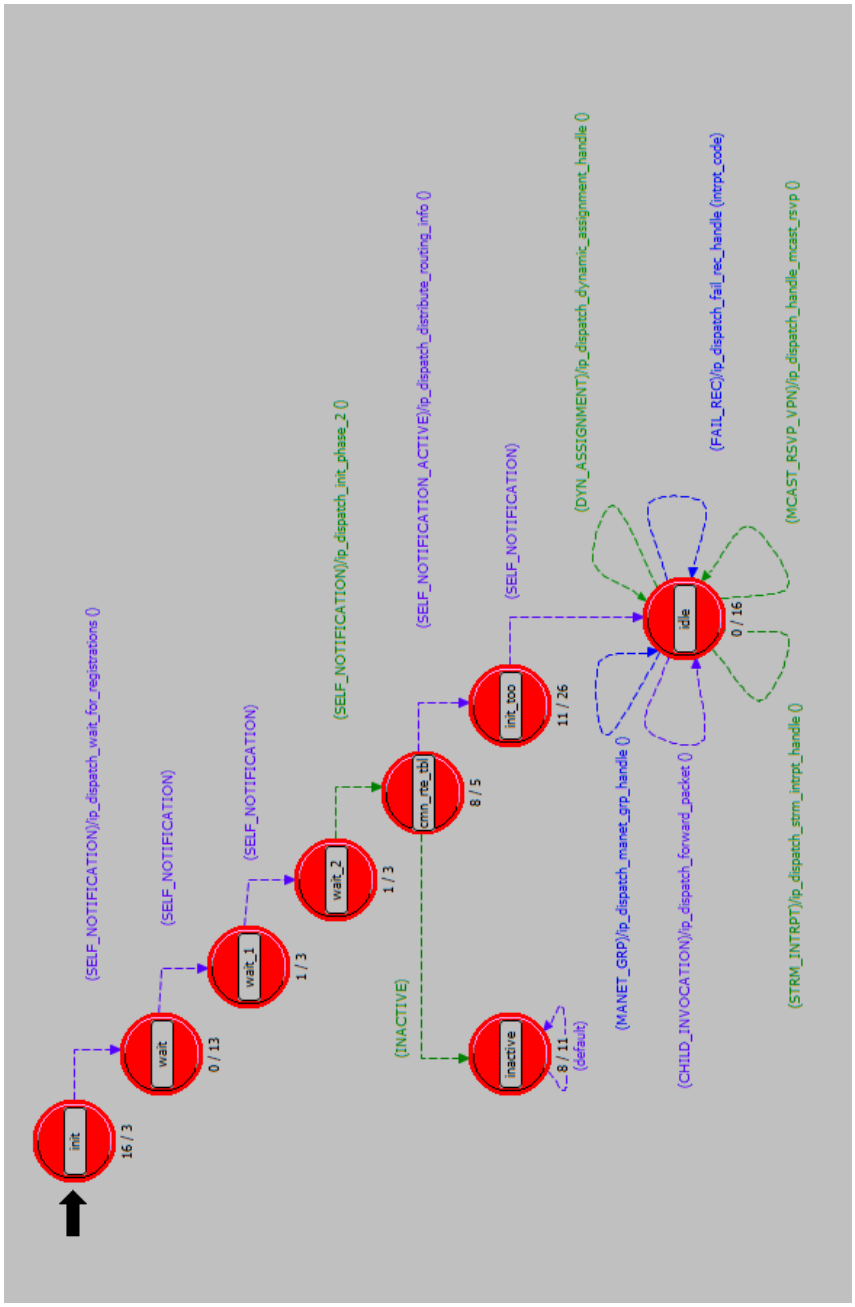

Figure 3.13: Process model 'controller_ip_dispatch.pr.m'

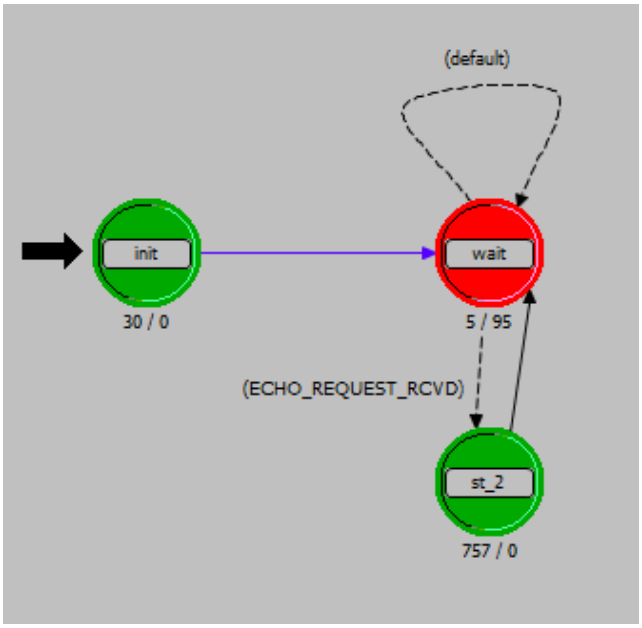

Figure 3.14: Process model for child process 'controller_icmp.pr.m'

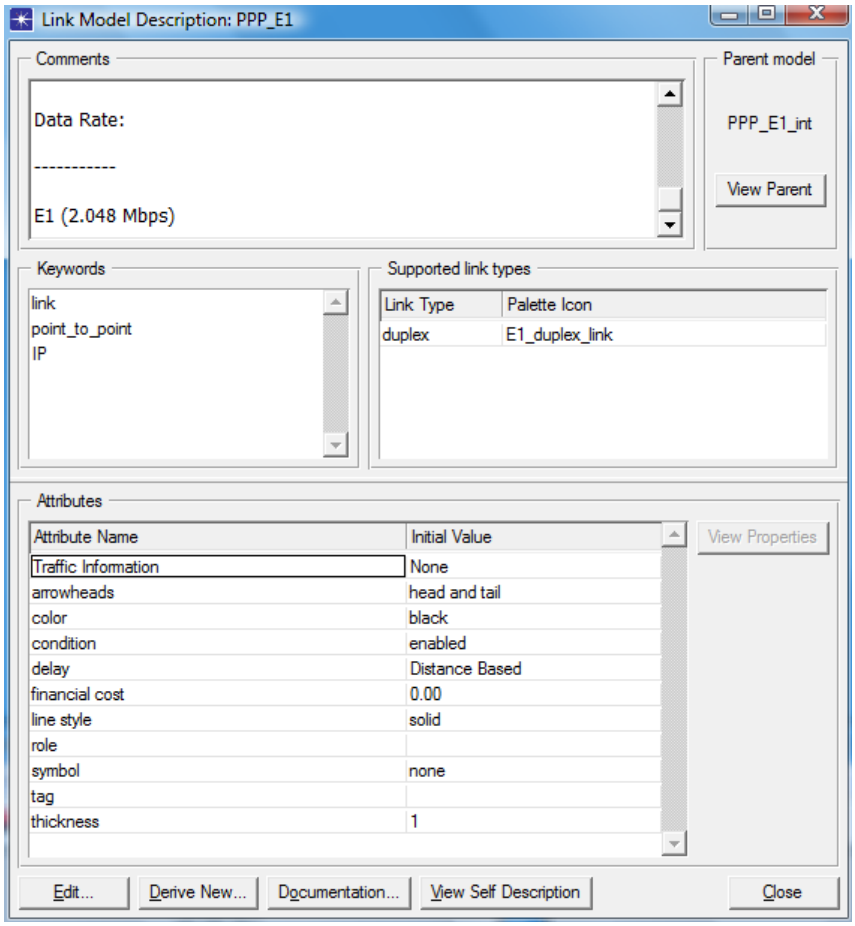

Figure 3.15: Link Editor

3.2 Global Congestion Problem in Multi-Bottleneck Networks and New Approaches for AQM

Fig.3.2 shows an autonomous system that is composed of end users, routers and connection links. A bottleneck problem arises, when the demand of multiple traffic flows using the same link goes beyond the available link transmission capacity. As a result, some packets are queued or dropped at router output interfaces. For handling the queue lengths appropriately and providing congestion avoidance, queue management algorithms are necessarily used at these interfaces. In this work, we are focused on the global congestion problem which leads us to observe changes on all router output queues. First of all, a centralized observation unit is designed to provide communication with IP routers. Then routers are specialized for communicating with the observation unit and for updating queue management schemes due to global congestion notifications. Handling global congestion and making necessary changes on the queue management schemes is a procedure with various steps. These steps are explained in this section.

The idea of following the global congestion level is developped by the fact that if there is an increasing congestion somewhere in the network, other regions are also in danger of being affected. The changes in global congestion level is considered as a sign of future congestion status by all interfaces.

In this section, the procedure for observation and avoidance of global congestion is proposed. First of all, it is useful to have some information about SOMs.

3.2.1 Self organizing maps

Self Organizing Map (SOM [51,52]) is a special class of artificial neural networks. SOMs are used to perform a mapping from the input data space R^n onto a one or two dimensional array of neurons. A parametric reference vector $m_i = [\mu_{i1}, \mu_{i2}, \ldots, \mu_{in}]^T \in R^n$ is associated with neuron i where μ_{ij} are

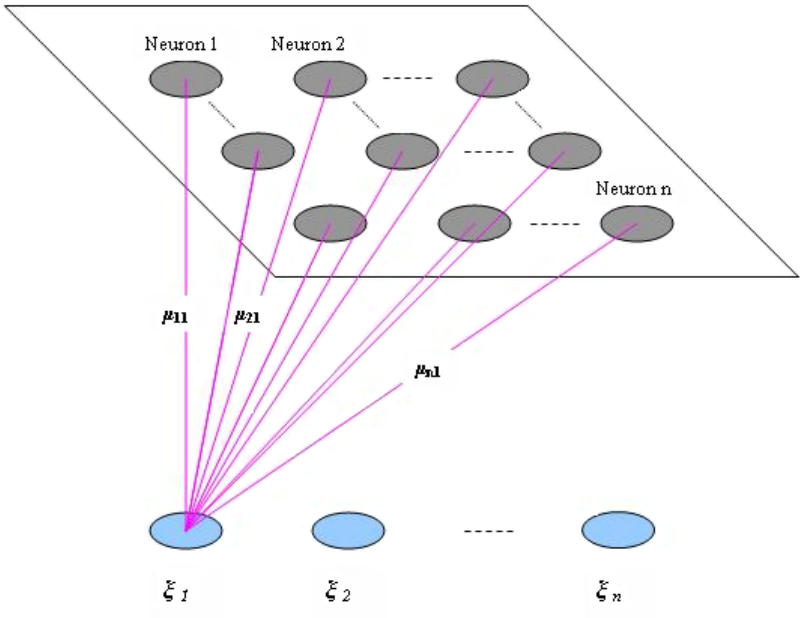

Figure 3.16: Configuration of a SOM

variable scalar weights. An input vector $x = [\xi_1, \xi_2, \ldots, \xi_n]^T$ is connected to all neurons in parallel via weights μ_{ij}. Configuration of a SOM is shown in Fig.3.16. For preventing visual complexity in the figure, only the connections for the first input element are drawn. The formation of the map is defined below [53]:

Step 1 (Initialization): Initial values are chosen for synaptic weight vectors, m_i.

Step 2 (Sampling): An input vector, x, is chosen randomly from the input vectors space.

Step 3 (Similarity matching): x is compared with all m_i values and the neuron with maximum similarity is defined as the Best Matching Unit (BMU). In many applications, Euclidean distances, $||x - m_i||$ are used to find similarities and the neuron with the smallest Euclidean distance to the input vector is defined as the BMU.

Step 4 (Updating): Synaptic weights are updated:

$$m_i(t+1) = m_i(t) + h_{ci}(t)\left[x(t) - m_i(t)\right] \ , \tag{3.1}$$

where $h_{ci}(t)$ is the neighbourhood function centered around the winning neuron. It can be defined by a Gaussian function:

$$h_{ci}(t) = \alpha(t).\left(-\frac{\|r_c - r_i\|^2}{2\sigma^2(t)}\right) , \tag{3.2}$$

where

$\alpha(t)$: a learning rate factor $0 < \alpha(t) < 1$

r_c, r_i: location vectors of neurons c and i, respectively

$\sigma(t)$: width of the neighbourhood function.

Step 5: Repeatition of Step 2 and the rest, until no noticeable changes are observed in the map.

The SOM algorithm is used in many areas, such as speech analysis and recognition, signal processing, telecommunications, process control... There are also researchers working on network applications. In studies [54] and [55], SOM based analysis of IP network traffic is proposed, data measured in a real environment is used to analyze dynamical properties of the traffic. The usage of SOMs in visualizing the QoS level of VoIP communications is studied in [55]. In that work, the real environment measurements of QoS-related parameters, such as end-to-end delay and packet loss rate are used to train the map; by this way QoS level of the communication could be estimated. Another work [56], introduces an adaptive RED algorithm, KRED, which uses Kohonen neural network model to solve the stability problem in queue lengths. An example for the usage of neural networks in AQM: L-RED is proposed in [57]. It is a variant of RED with prediction ability for increasing network utilization rate.

In this dissertation, two different AQM approaches are proposed and simulations are performed for testing their performances. These approaches (Approach 1 and Approach 2) depend on redefinition of some RED parameters by investigating the general effects of local congestions on global congestion. SOMs are used to study the tendency of global congestion and make future predictions. Approach 1 and Approach 2 are explained in Subsection 3.2.2 and Subsection 3.2.3, respectively.

3.2.2 Approach 1

In this approach, the minimum threshold ($\min_{th}$) and maximum probability denominator ($max_{p_{den}}$) parameters of RED have been redefined, using the global congestion notifiers which are collected from a trained SOM. Steps of Approach 1 could be summarized as given below:

- Observing the global congestion status of a network by using the queue length information from router output interfaces.
- Training a SOM by the help of global congestion status information.
- Using the trained SOM for estimating future congestion behaviour of the network.
- Updating some RED parameters in order to avoid future congestion

Approach 1 will be explained in detail, by the help of an autonomous network model, which is seen in Fig.3.2. This network model is designed by using OPNET Modeler and it is used during simulation of two different cases for different traffic models. The network has seventeen IP routers (r1, r2, ..., r16, r_controller), six pairs of clients&servers (c1&s1, c2&s2, ..., c6&s6) and a centralized observation unit (controller). A video conferencing traffic is generated between clients and servers. The IP routers are similar to 'ethernet4_slip8_gtwy_adv', besides, some changes have to be made in the model for providing the router an additional function: managing a communication packet traffic between the centralized observation unit and itself. The details about the router is given in Section 3.1. Clients and servers in Fig.3.2 are using the node model 'ethernet_wkstn_adv'. The contol unit model is similar to 'ethernet_wkstn_adv', but again as in the router's improvement, the functionality of the model is enriched. Simulations are performed for a number of scenarios under two main cases, Case1 and Case 2. In Fig.3.2, IP addresses of some output interfaces are given. These interfaces are the ones which suffer

from congestion and packet drops in at least one of the simulation cases. Case 1 and Case 2 are similar in the following ways:

- Both cases are simulated through 7 scenarios: Scenario 1 is for data collection and SOM generation and training; Scenarios 2-7 are for data collection, investigation on the trained SOM, future congestion estimation and RED improvement by parameter updates.
- Simulations are performed on the same network model.
- Traffic type (video conferencing) is the same in both cases.
- The kind of data (queue lengths of router output interfaces) used for training a SOM is the same.
- The idea behind RED improvement is the same.

Besides these similarities, there are differences between the two cases:

- Amounts of traffic flow are different.
- The interfaces where congestion occur are not the same.
- Formulas used for RED parameter estimation are similar, but not the same.

After this brief information about both simulation cases, it will be easier to understand the details which are explained below:

CASE 1:

In this case, seven scenarios are generated for inspecting congestion problem on the network model shown in Fig.3.2:

Scenario 1: The purpose of this scenario is obtaining data about the congestion behaviour of the network by means of the queue lengths on the router output interfaces. The simulation results will give a general information about the system behaviour when there is no control input destined to affect the

AQM algorithms of routers. Scenario 1 is followed by 6 more scenarios which will use the SOM that is obtained at the end of this scenario. Therefore, Scenario 1 starts with data collection and ends with the generation of SOM based on these data. In this scenario, in addition to the video conferencing traffic generated between clients and servers, there is a traffic between routers and the observation unit: ip packets carrying the queue length values (that belong to the queues at various output interfaces) are sent to the centralized observation unit and reply packets are produced. Each of these special packets are originated in one of the routers; carries information about an output interface of this router (the address code of the output interface and the queue length at this interface) and is sent in the first 2ms of each 10ms. In the observation unit, these packets are received, the queue length values are collected and (together with the receipt time) written to a file with respect to the address code of the interface. Then, reply packets are generated by the observation unit and sent back to the routers in the first 100ms of each second. In the following scenarios, the function of reply packets will be explained in detail, for now it is enough to know that reply peckets are for readjusting some QoS parameters in the routers. First-In First-Out mechanism (FIFO) with enabled RED criteria is used as a QoS implementation in the routers. In Scenario 1, RED parameters (maximum threshold, minimum threshold, maximum probability denominator) are kept constant whatever reply is received. The values of these parameters are as follows:

• minimum threshold: $min_{th} = 100$ packets

• maximum threshold: $max_{th} = 200$ packets

• maximum probability denominator: $max_{p_{den}} = 10$

• exponential weight factor (ewf): 9 (The queue weight, w_q, is found by the formula: $w_q = 2^{-ewf}$)

To give more details about this scenario, it must be added that the evolutions in the queue lengths are noticeable on only 28 of the interfaces (IP addresses for all -and more- of these interfaces are shown in Fig.3.2), the rest of the interfaces had no more than 20 packets in their queues during simulation.

Table 3.1: Data collection table

Time(seconds)	Queue length at output interface 1	Queue length at output interface 2	...	Queue length at output interface n
0.00				
0.01				
0.02				
⋮				
Stop time				

Collected data are stored in 28 different files, each file for a different interface data. At the end of the simulation, the information in these files are used to build a matrix of queue lengths, as shown in Table 3.1.

The matrix of data shows the evolution of queue lengths during periods of 10ms at various interfaces, till the end of the simulation. The purpose of obtaining this matrix is producing a SOM and make observations on the map after training. The 'som toolbox' for MATLAB is used for initialization and training phases.

The queue length data during the time interval 8.24 sec.-69.64 sec. is used for training (the queue length values are all zero in the time interval 0 to 8.23 seconds). The size of the data matrix is 6141 x 28. As will be seen later, each row in this matrix is an input vector for the map. Now, the data are ready for use, the following procedure is an explanation of the initialization and training:

Step 1: Producing a SOM data structure (named 'som_data') by using the data matrix.

Step 2: Normalizing each column (queue length values of an interface) of the matrix due to variance criteria. 28 different normalizations with different mean and variance values took place. The mean and variance values of each column are saved in a 28x1 cell.

Field	Value
name	'dm'
type	'som_data'
comp_names	<28x1 cell>
comp_norm	<28x1 cell>
labels	<6141x1 cell>
data	<6141x28 double>
label_names	[ ]

Figure 3.17: Input data structure for SOM

Step 3: Appending 'label's to input vectors, in order to group them with respect to the congestion level they represent:

Remembering that the maximum threshold level is 200 packets, an interface with a queue length that is greater than 200 packets may be defined as 'highly congested'. The number of highly congested interfaces is an indicator of the network's congestion level. Due to this definition, each row of 'som_data' is labeled such that each label value is the number of highly congested interfaces at that moment. As a summary, there are 6141 input vectors (1x28) to be used as an input for the map and each has a label value representing the number of vector elements whose value are greater than 200. Until now, the procedure of reshaping the vectors in order to make them appropriate inputs for a SOM is explained. The next phase is the definition of SOM structure and the training process.

Step 4: When the command 'som_gui' is typed on the command window of Matlab, a user-friendly menu window is opened. There are necessary parts in the menu to be filled appropriately. Here are the settings for these parameters:

• map size: 10x20 (a map of 200 neurons)

• lattice: hexagonal (The neurons are hexagonal, allowing a neighbourhood of maximum 6 neurons)

• shape: sheet

There are also some settings that have to be made before starting the training phase:

• type of training: sequential (finetune training is preceded by rough training)

Field	Value
type	'som_map'
codebook	<200x28 double>
topol	<1x1 struct>
labels	<200x11 cell>
neigh	'gaussian'
mask	<28x1 double>
trainhist	<1x3 struct>
name	'SOM 22-Jan-2010'
comp_names	<28x1 cell>
comp_norm	<28x1 cell>

Figure 3.18: SOM data structure

- initial training radius: 10 (rough training), 2 (finetune training)
- final training radius: 2 (rough training), 1 (finetune training)
- training length: 500 epochs (rough training and finetune training)
- initial learning rate: 0.5 (rough training), 0.05 (finetune training)

Step 5: As a preliminary work for Scenarios 2 to 7, it is necessary to obtain the codebook for the data that are used to train SOM. After training, we end up with a SOM whose structure is shown in Fig.3.18. The codebook matrix is defined in SOM data structure and it is composed of codebook vectors of size 1x28 for the neurons 1 to 200. Each row of the codebook matrix is the codebook vector of a different neuron. In Fig.3.19, the black points in the neurons are indicators of how often these neurons are hit, the empty neurons are never hit, full black neurons are the most hit ones.

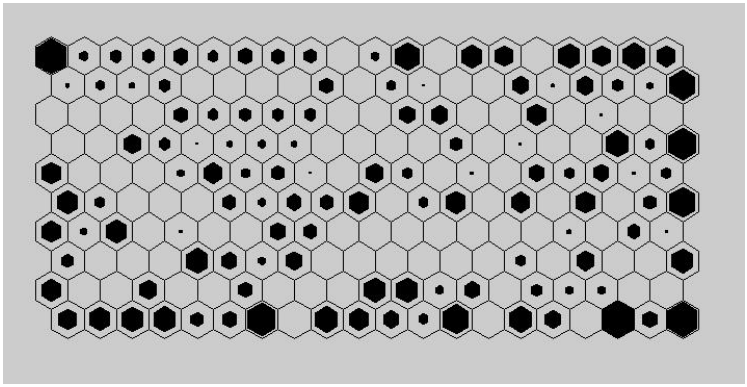

Figure 3.19: Hit points on the SOM (Case 1)

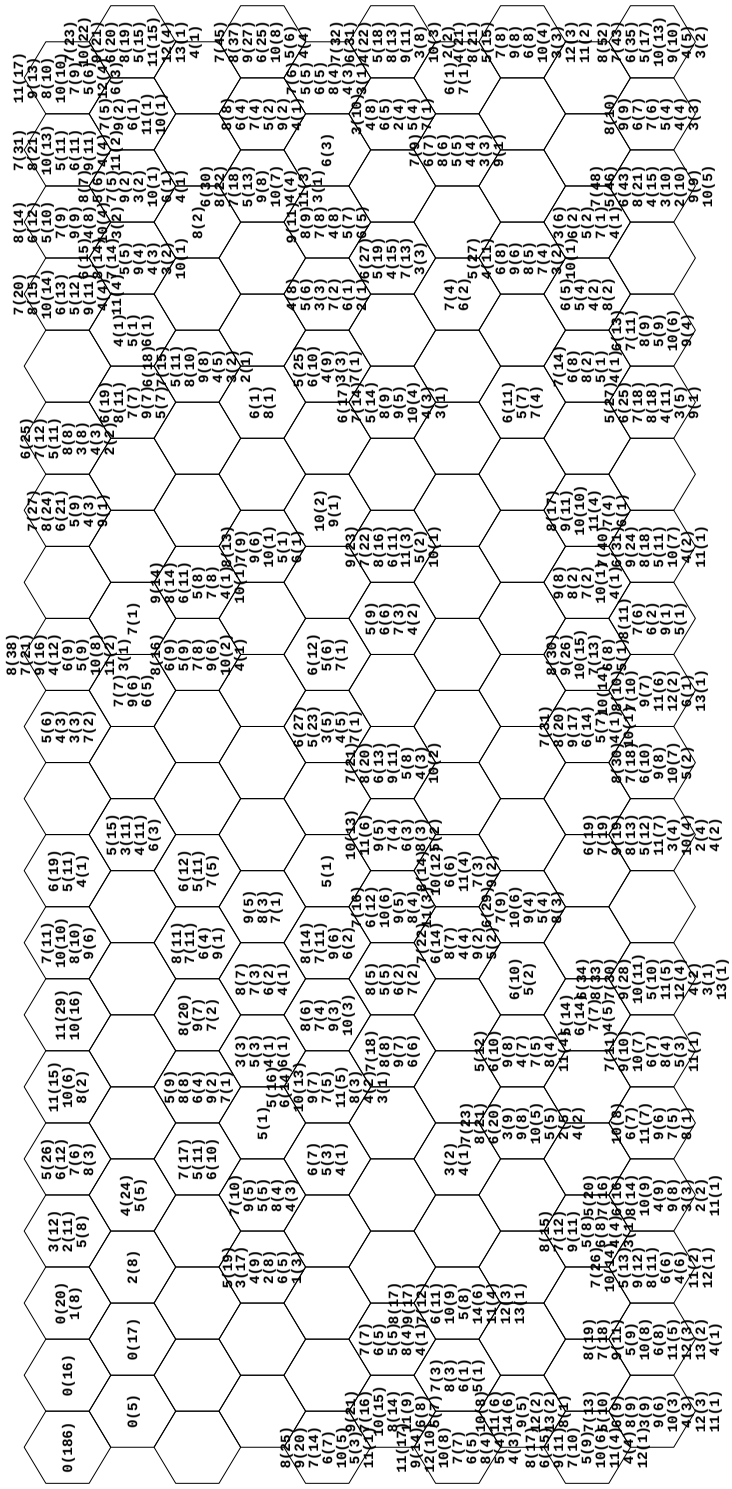

Figure 3.20: Hit and frequency values of the neurons for Case 1 (map is rotated counterclockwise by 90°)

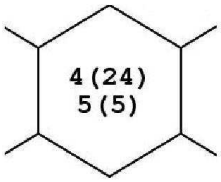

Figure 3.21: Hit and frequency values for neuron 32 (Case 1)

In Scenarios 2 - 7, not only the hit values but also the frequencies of the hits are taken into consideration while studying the congestion behaviour. In Fig.3.20, a counterclockwise rotated (by 90°) version of the map, these values are shown inside the neurons, most of the neurons carry the following information:

$$hit_1(frequency_1)$$
$$hit_2(frequency_2)$$
$$\vdots$$
$$hit_n(frequency_n)$$

which means that the neuron is hit as the BMU for inputs with n different labels (hit_1, hit_2, ..., hit_n). The frequency values in paranthesis show how many times the neuron is hit for that label. Let's observe neuron 32: in Fig.3.21, it is seen that the neuron has taken 29 hits, 24 of the hits have label value 4, 5 of them have label value 5.

The trajectory of BMUs is also plotted to obtain an idea about the reflection of congestion behaviour of the traffic on the map. The trajectory of the BMUs (the path of BMUs corresponding to the input vectors, from the 1^{st} to the 6141^{th}) is shown in Fig.3.22. It is noticeable that the trajectory finds its way among the neighboring neurons, except rarely occuring regional jumps. The neurons on the top left corner represent the vectors with label value 0, meaning no heavy congestion. The trajectory then changes its region, towards the right side of the map where label values are greater than 0.

Scenarios 2 to 7 are preceded by Scenario 1 because the resulting map structure of Scenario 1 is used in their simulations. The observation unit in

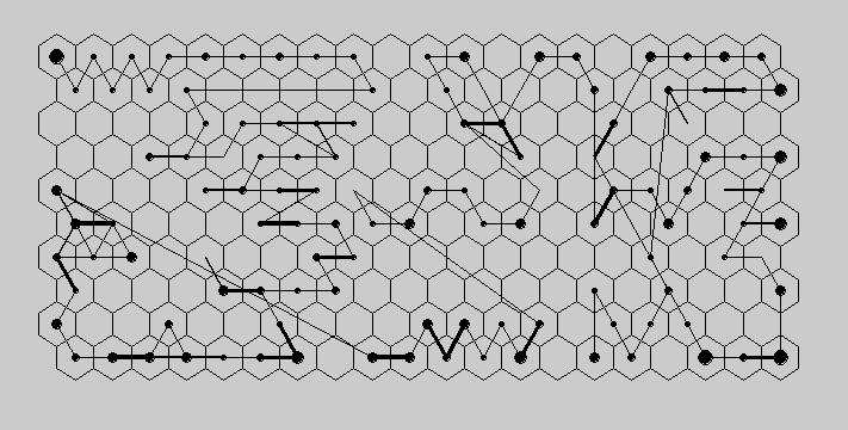

Figure 3.22: Best matching unit trajectory (Case1)

Scenario 1 send reply packets to routers; however, these packets do not carry any congestion notifications. In Scenarios 2 - 7, the reply packets sent back to the routers are worthy of more attention because the data carried in these packets are used to update one or two RED parameters, the minimum threshold value, maximum probability denominator or both. The main topic of discussion is that how appropriately the parameters are updated. The idea behind the update procedure is simple: increase the number of dropped packets if there is a future congestion notification and decrease otherwise. Approach 1 presents an early drop procedure, helping to decrease end-to-end delays and jitters while preventing a decrease in throughput and link utilization.

Scenario 2: A similar video conferencing traffic is generated as in Scenario 1, therefore similar congestion behaviour is expected. The data collection procedure is continued also in this scenario. At specific times, the vector of queue lengths for the 28 interfaces, which are the interfaces in Scenario 1, is to be applied as an input to the SOM obtained before. This vector is compared with each SOM codebook vector and the euclidean distances are found. The neuron with the least distance is the BMU for this vector. The BMU carries information about the congestion level of the network not only for now but also for the nearest future. Here, we have to turn back to our analysis in the map about hits and hit frequencies of the neurons. When a neuron is hit, we

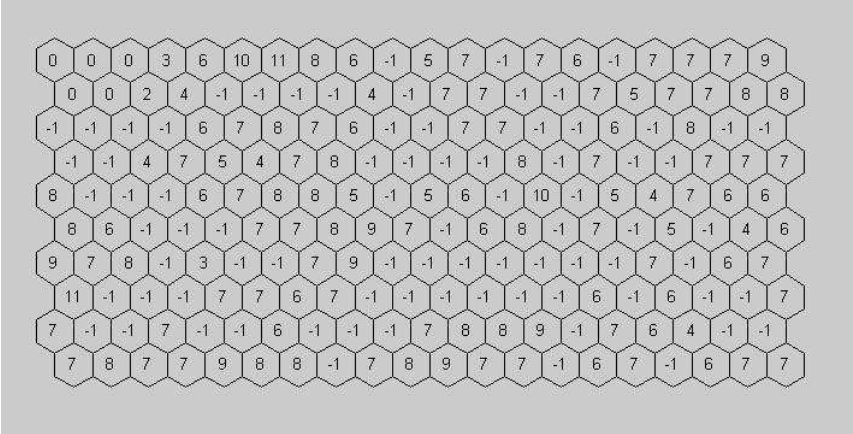

Figure 3.23: Labels obtained by using weighted averaging method (Case 1)

are faced with a list of label values with different weights (hit frequencies). For obtaining the congestion state of the network, the label value information on the BMU and on the neighbours have to be taken into account. Let's say, the hit and frequency values for the BMU are as follows:

10(2) 4(7) 8(1)

The BMU label value $label_{BMU}$ is calculated by using a weighted average:

(10*2 + 4*7 + 8*1)/(2+7+1)=5.6

If this value is rounded up, $label_{BMU}$ is obtained as 6.
This value is not enough for our control process because we have to predict the congestion status in the future. Additionally, the label values of the neighboring neurons also have to be calculated in the same manner. (The exact label values for the neurons are shown in Fig.3.23. The neurons with -1 value are the ones which had no hits.)

The calculations that are made so far is a preparation for finding $label_{final_1}$. $label_{final_1}$ is the value that the observation unit sends to the routers, as a notification of the network congestion level and it is obtained in the following way:

$$label_{final_1} = \beta * label_{BMU} + (1-\beta) * ((label_{n1} + label_{n2} + \ldots + label_{nN})/N) \ , \quad (3.3)$$

where $label_{ni}$ is the label value calculated for neighbour i among N neighbours $(i = 1, 2, \ldots, N)$.

A greater weight is assigned for the BMU than the one assigned for the neighbouring neurons. Therefore, β is taken as 0.75 in order to make $label_{BMU}$ more effective on the calculation. So far, we assumed that neither $label_{BMU}$ nor $label_{ni}$ is -1, in other words, we assumed every neuron has at least one hit value. What happens if this assumption is not true? Firstly, let's assume BMU has no hit values, but at least one of the neighbours has. Then $label_{final_1}$ is calculated by dividing the sum of the labels different from -1 by the number of neighbors with labels different from -1:

$$label_{final_1} = (label_{p1} + label_{p2} + \ldots + label_{pP})/P\,, \tag{3.4}$$

where $label_{pl}$ is the label value calculated for neighbour l among P neighbours $(l = 1, 2, \ldots, P)$ whose label values are different from -1.

The problem arises when not only the BMU but also the neighbours have -1 values? There may be neurons such that neither itself nor the neighbors have valid label values. Here is the solution for this problem: we should use the secondary neighboring layer, (the layer of neurons that are two neurons away, the neighbours used so far were only one neron away) and use their label values:

$$label_{final_1} = (label_{m1} + label_{m2} + \ldots + label_{mM})/M\,, \tag{3.5}$$

where $label_{mi}$ is the label value calculated for secondary neighbour j among M neighbours $(j = 1, 2, \ldots, M)$. In Case 1, we are not faced with such a problem. (But in Case 2, the secondary neighboring layer is necessarily used.)

Whenever a router receives the reply packet with the $label_{final_1}$ value encapsulated in, it forwards the packet to the ip layer, where the RED parameter minimum threshold (min_{th}) is recalculated due to the following formula:

$$min_{th} = 100 - \alpha_1 * label_{final_1} \tag{3.6}$$

'How did we obtain this formula?' The answer is simple; when the hit values shown in Fig.3.20 are investigated, it is seen that '14' is the maximum

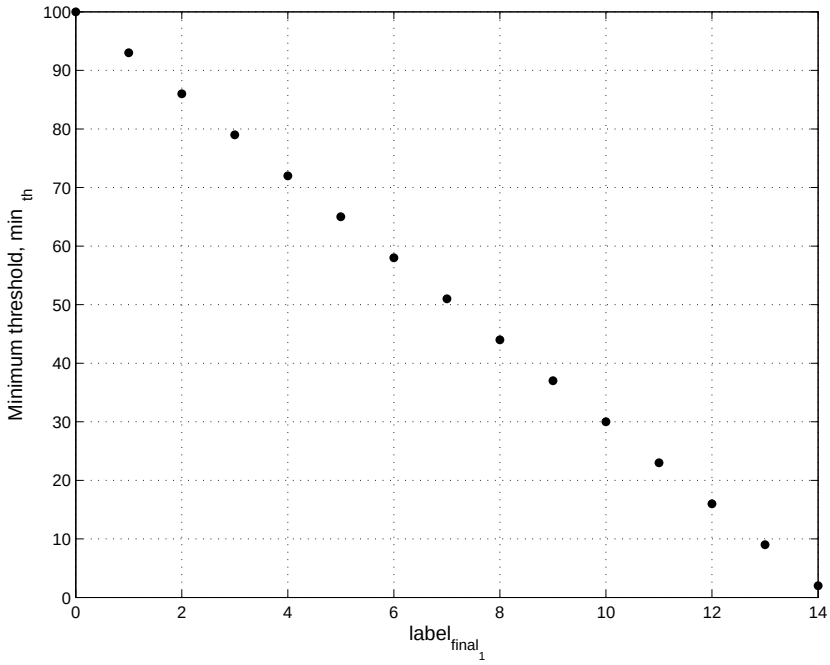

Figure 3.24: min_{th} vs $label_{final_1}$ (Case 1)

label value, meaning there are at most 14 congested interfaces at a time. This is the most dramatic case we expect for this type of traffic, therefore it must cause the minimum value in the minimum threshold, almost 0 packets. In addition to this, if the label value is '0', no need to worry about congestion throughout the network, normal settings of RED parameters will be enough for congestion avoidance. As a result, the formula will cause a linear decrease in the value of minimum threshold as $label_{final_1}$ value increases. In our simulation, $\alpha_1 = 7$ is used. As a result, min_{th} takes a value in the region [2 100] packets; until the acceptance of the next min_{th} value, the packets in the queue are not dropped until the average queue length reaches this value. In Fig.3.24, min_{th} vs $label_{final_1}$ graphics is shown.

Scenario 3: In Scenario 2, the $label_{final_1}$ value is sent to a router by the observation unit and the receiving router uses this value to update the minimum threshold level of its output interfaces. The only difference between Scenarios 2 and 3 is the way the receiving router uses the $label_{final_1}$ value. In Scenario 3, minimum threshold level is kept constant, maximum probabil-

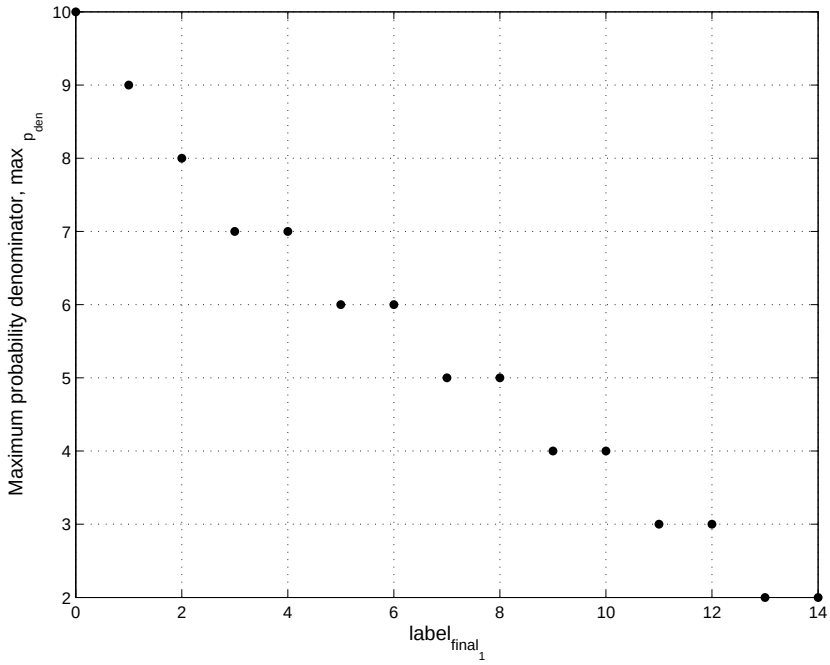

Figure 3.25: $max_{p_{den}}$ vs $label_{final_1}$ (Case 1)

ity denominator ($max_{p_{den}}$) is changed instead, due to the value of $label_{final_1}$. Fig.3.25 shows the relation between $label_{final_1}$ and $max_{p_{den}}$, while Fig.3.26 is for max_p vs $label_{final_1}$.

$$max_p = 1/max_{p_{den}} \tag{3.7}$$

Scenario 4: In this scenario, $label_{final_1}$ value that is sent by the observation unit is used to update the values of min_{th} and $max_{p_{den}}$ at the same time. min_{th} vs $label_{final_1}$ is shown in Fig.3.24; $max_{p_{den}}$ vs $label_{final_1}$ is shown in Fig.3.25.

Scenario 5: The distinction between the two scenarios, Scenario 2 and 5, is the way that is followed to find the label value associated with the BMU ($label_{BMU}$), and its neighbours. In the observation unit a vector is produced with the 28 queue length values collected from output interfaces. By using the SOM generated in Scenario 1 and calculating the euclidean distances, the BMU is found. Let's use the same example in Scenario 2 for better understanding and assume that the hit and frequency values for the BMU neuron

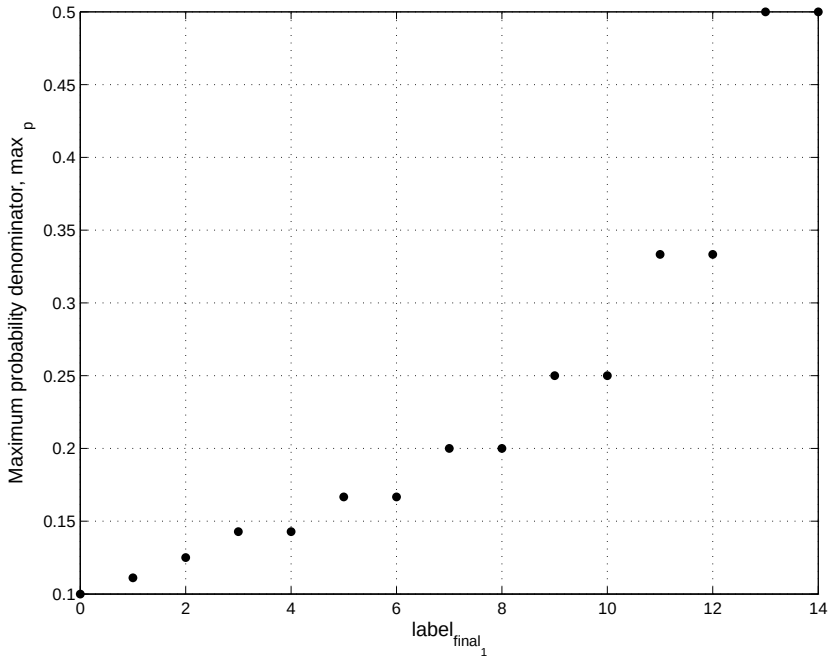

Figure 3.26: max_p vs $label_{final_1}$ (Case 1)

are given as: 10(2) 4(7) 8(1) (If we make a prediction among the values 10, 4 and 8; the probability of taking a hit with label 10 is 2/10, with label 4 is 7/10 and 8 is 1/10). A random number is generated between numbers 0 and 10 (=2+7+1); if the number is between 0 and 2, the label value is taken to be 10, if it is between 2 and 9, the label value is 4, otherwise (between 9 and 10) the label value is 8.

As mentioned in Scenario 2, the labels for the primary or secondary (for a special case) neighbours are also important for us. The procedure we used for BMU, that is based on the hitting probabilities, is repeated for the neighbours. Then the label value is found by using the suitable one among the following methods:

Method 1: This method is used when BMU label $\neq -1$ and at least one of the primary neighbours' labels $\neq -1$.

$$label_{final_2} = \beta * label_{BMU} + (1-\beta) * ((label_{n1} + label_{n2} + \ldots + label_{nN})/N), \quad (3.8)$$

where N is the number of primary neighbours whose label $\neq -1$.

In this method, a greater weight is assigned for the BMU than the one assigned for the neighbouring neurons. β is taken as 0.75 in order to make $label_{BMU}$ more effective on the calculation.

Method 2: This method is used when BMU label = -1 and at least one of the primary neighbours' labels $\neq -1$.

$$label_{final_2} = (label_{p1} + label_{p2} + \ldots + label_{pP})/P, \tag{3.9}$$

where P is the number of primary neighbours whose label $\neq -1$.

Method 3: This method is used when BMU label = -1, primary neighbours' labels =-1 and at least one of the secondary neighbours' labels $\neq -1$.

$$label_{final_2} = (label_{m1} + label_{m2} + \ldots + label_{mM})/M, \tag{3.10}$$

where M is the number of secondary neighbours whose label $\neq -1$. $label_{final_2}$ is used to determine min_{th}, just as in Scenario 2:

$$min_{th} = 100 - \alpha_1 * label_{final_2} \tag{3.11}$$

In simulation of Scenario 5, $\alpha_1 = 7$ is used.

Scenario 6: This scenario is similar to Scenario 5 in the way, the label values are obtained, a probabilistic method is followed. $label_{final_2}$ is used to determine the new value for $max_{p_{den}}$, the relation between these two parameters are seen in Fig.3.25.

Scenario 7: As in Scenarios 5 and 6, a probabilistic method is used in the observation unit to obtain $label_{final_2}$ value. When this value is sent to a router, the router uses it to update both min_{th} and $max_{p_{den}}$ at the same time by using the relations given in Fig.3.24 and Fig.3.25.

The properties of Scenarios 1 - 7 are summarized in Table 3.2 and Table 3.3.

Table 3.2: Summary of Scenarios 1 - 4 of Case 1

Property	**Scenario 1**	**Scenario 2**	**Scenario 3**	**Scenario 4**
Queue management technique	RED	Approach1	Approach1	Approach1
label for congestion notification	—	$label_{final_1}$	$label_{final_1}$	$label_{final_1}$
label determination method	—	weighted averaging	weighted averaging	weighted averaging
buffer limit (pkts)	300	300	300	300
min_{th} (pkts)	100	updated due to (3.6)	100	updated due to (3.6)
max_{th} (pkts)	200	200	200	200
$max_{p_{den}}$ (pkts)	10	10	updated as in Fig.3.25	updated as in Fig.3.25

Table 3.3: Summary of Scenarios 5 - 7 of Case 1

Property	**Scenario 5**	**Scenario 6**	**Scenario 7**
Queue management technique	Approach1	Approach1	Approach1
label for congestion notification	$label_{final_2}$	$label_{final_2}$	$label_{final_2}$
label determination method	probabilistic	probabilistic	probabilistic
buffer limit (pkts)	300	300	300
min_{th} (pkts)	updated due to (3.11)	100	updated due to (3.11)
max_{th} (pkts)	200	200	200
$max_{p_{den}}$ (pkts)	10	updated as in Fig.3.25	updated as in Fig.3.25

A simulation of 70 seconds is performed on each of Scenarios 1 - 7. In Fig.3.27 - Fig.3.32, the results of the simulations are given. In each figure, there are 7 graphics, each graphic belongs to a different scenario. The graphics are for Scenario 1 (plotted in black line), Scenario 2 (plotted in red line), Scenario 3 (plotted in red dashed line), Scenario 4 (plotted in red pluses), Scenario 5 (plotted in blue line), Scenario 6 (plotted in blue dashed line), Scenario 7 (plotted in blue pluses). Fig.3.27 - Fig.3.31 are for observation of some global statistics and Fig.3.32 shows the evolution of queue length at interface with IP address: 192.0.2.2.

As seen in Fig.3.27, maximum end-to-end delay is observed for Scenario 1 but its performance for delay variation is better than most of other scenarios. Minimum end-to-end delay and delay variation are observed in Scenario 7.

In all the simulations, the traffic load was similar but not the same, therefore observing the ratio of traffic received to traffic sent may be more helpful for comparing throughputs. By using the results in Table 3.4, that are based on the graphics in Fig.3.30 and Fig.3.31, it is seen that throughput performances of Scenario 3, 4 and 7 are better than that of Scenario 1.

Simulation results show that, for Case 1, Scenarios 3, 4 and 7 present good performance in providing trade-off between throughput and delay.

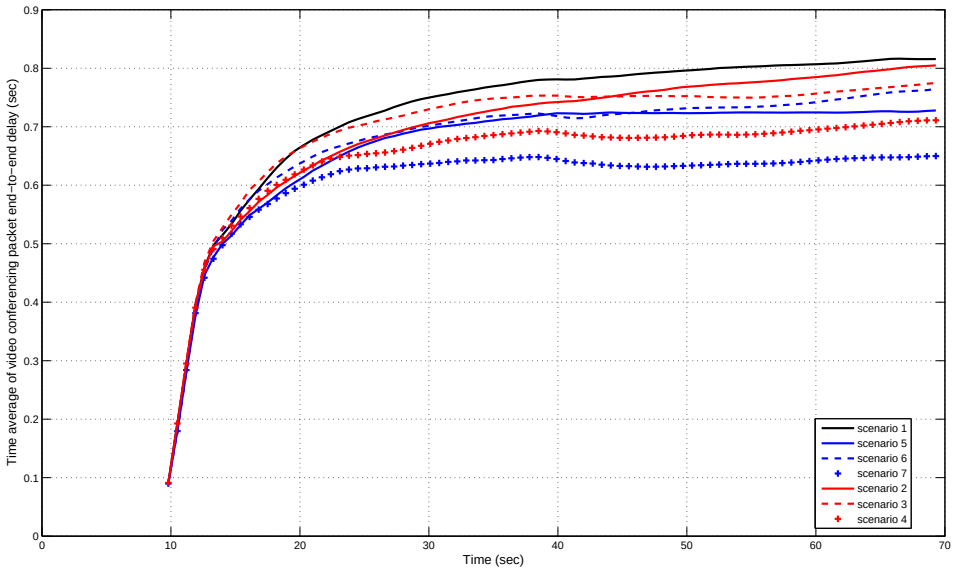

Figure 3.27: Time average of video conferencing packet end-to-end delay (Case 1)

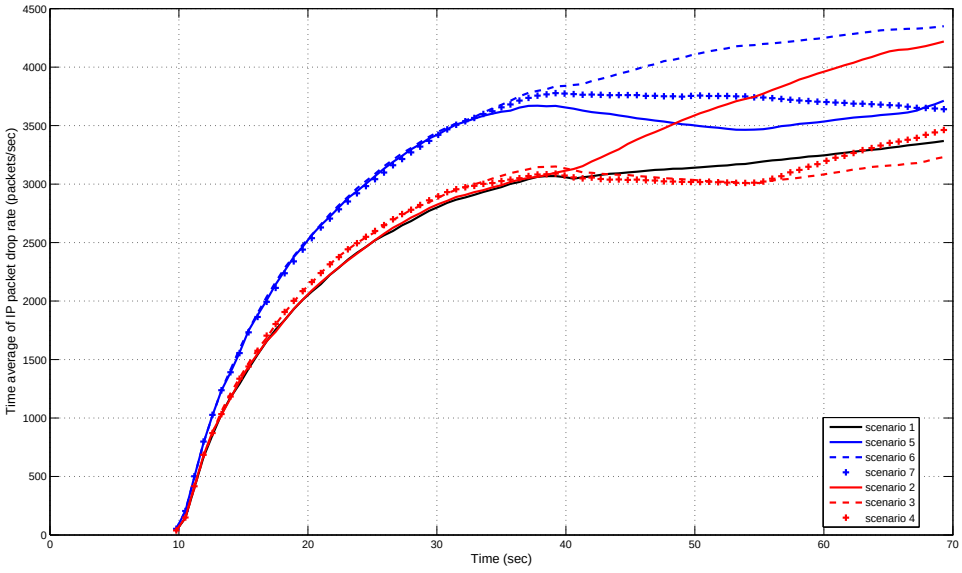

Figure 3.28: Time average of IP packet drop rate (Case 1)

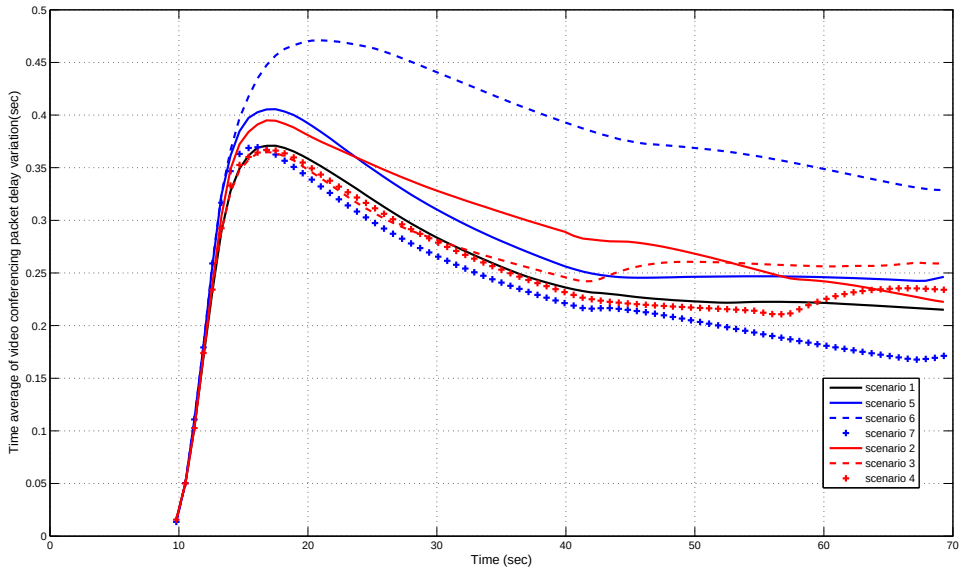

Figure 3.29: Time average of video conferencing packet delay variation (Case 1)

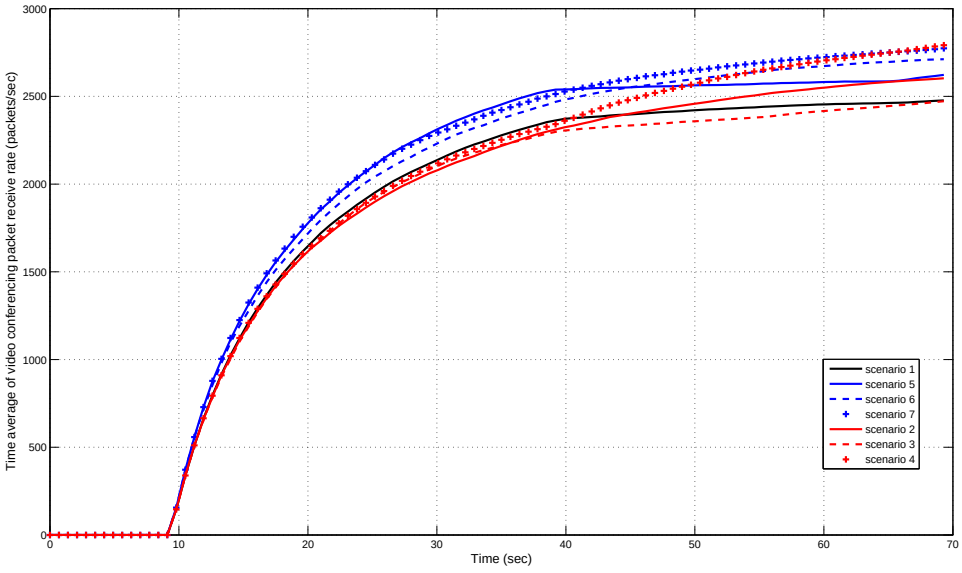

Figure 3.30: Time average of video conferencing packet receive rate (Case 1)

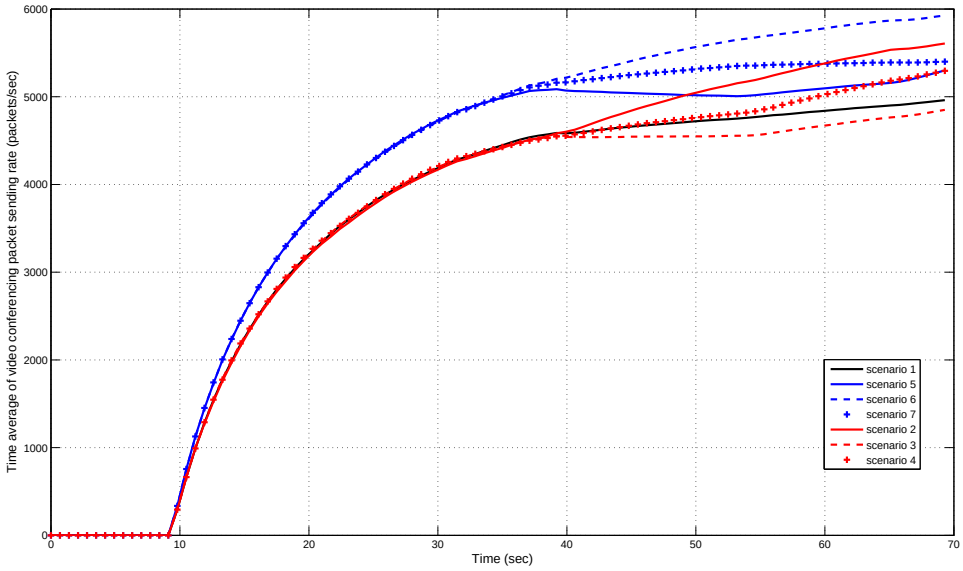

Figure 3.31: Time average of video conferencing packet sending rate (Case 1)

Table 3.4: Video conferencing packets received (r) to sent (s) ratio (Case 1)

Scenario no	Time average of traffic sending rate (pkts/sec) at 70^{th} second	Time average of traffic receiving rate(pkts/sec) at 70^{th} second	$\frac{r}{s}$
1	4961	2476	0.50
2	5606	2602	0.46
3	4850	2472	0.51
4	5299	2791	0.53
5	5299	2622	0.49
6	5927	2711	0.46
7	5399	2773	0.51

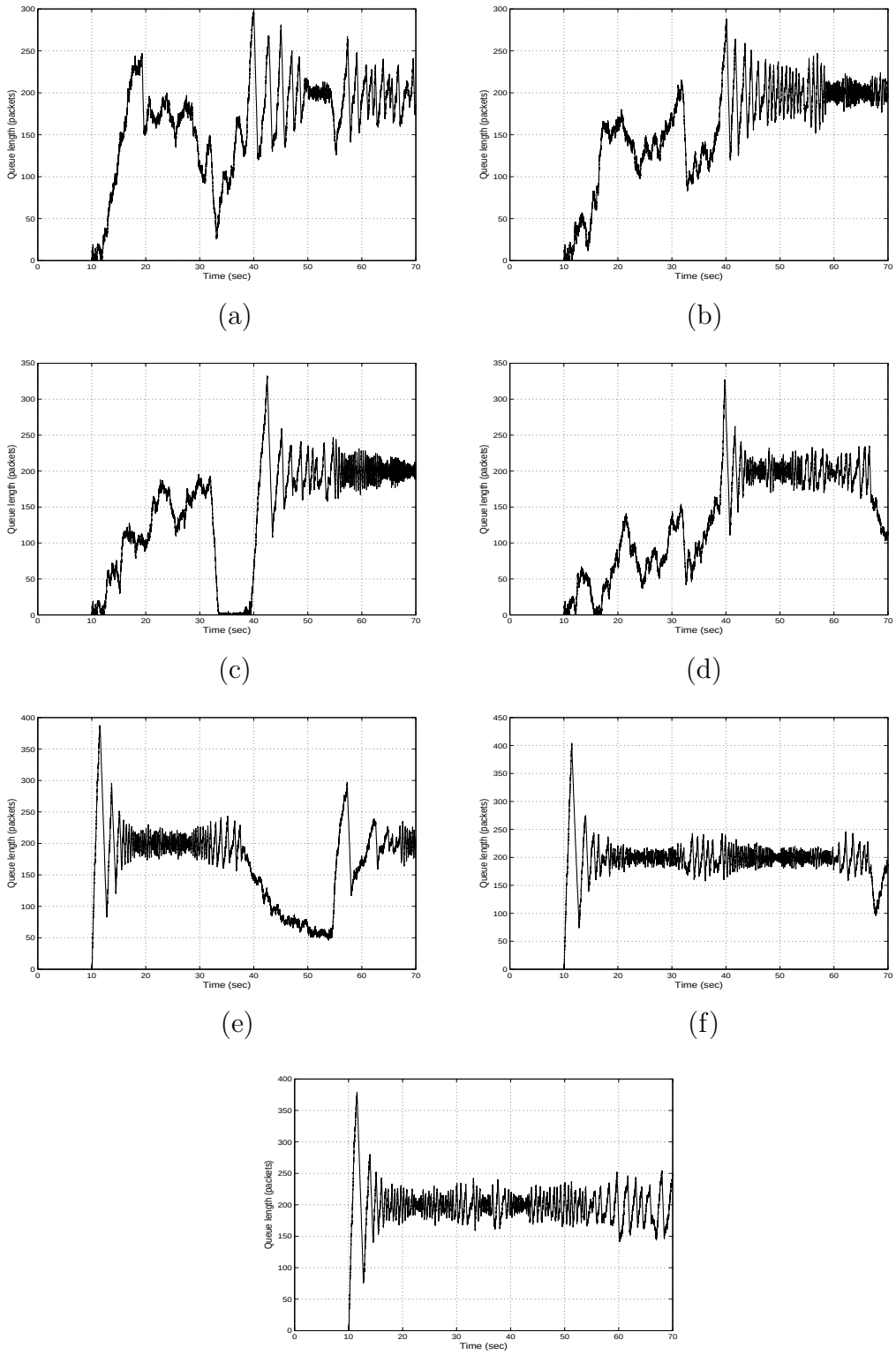

Figure 3.32: Queue length at the interface with IP address: 192.0.2.2 for Case 1. The graphics are for (a) Scenario 1, (b) Scenario 2, (c) Scenario 3, (d) Scenario 4, (e) Scenario 5, (f) Scenario 6, (g) Scenario 7.

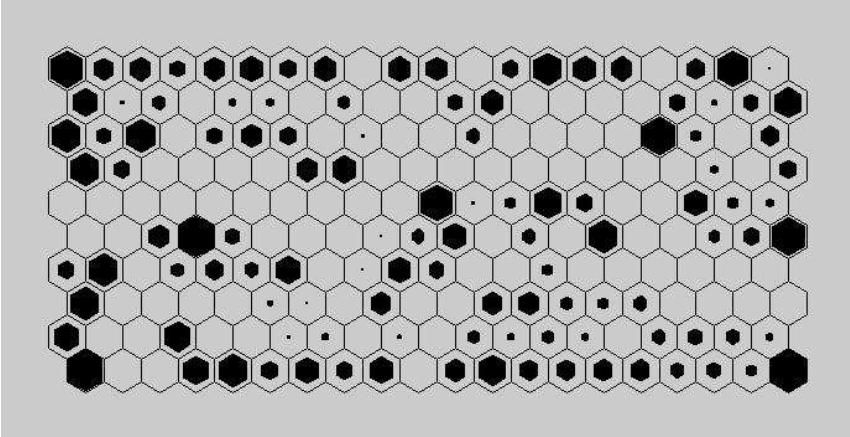

Figure 3.33: Hit points on the SOM (Case 2)

CASE 2:

The network model used for simulation of Case 2 is the same as the model for Case 1, as seen in Fig.3.2. Case 2 differs from Case 1 in the characteristics of the video conferencing traffic generated in the network. As in Case 1, the network is analyzed under seven scenarios:

Scenario 1: The values of RED parameters are the same as in Case 1. The values of SOM parameters used in initialization and training are also the same. There are 29 interfaces where noticeable queue variations are observed. These interfaces and their IP addresses are shown in Fig.3.2. The map size of trained SOM is 10x20 and the size of the resulting codebook matrix is 200*29.

In Fig.3.33, the hits that each neuron get are indicated by black points, empty neurons are the ones with no hits. 6169 input vectors are used to train the map and BMU trajectories are drawn. It is easy to follow the trajectory since it finds a way among the neighbours most of the time, as seen in 3.34.

Scenario 2: The label values are calculated as described in second scenario of Case 1. Hit and frequency values, which are seen in Fig.3.35, are used to calculate a weighted average value. The label values calculated for each neuron is shown on the map in Fig.3.36. For neuron 133 and neuron 143 there

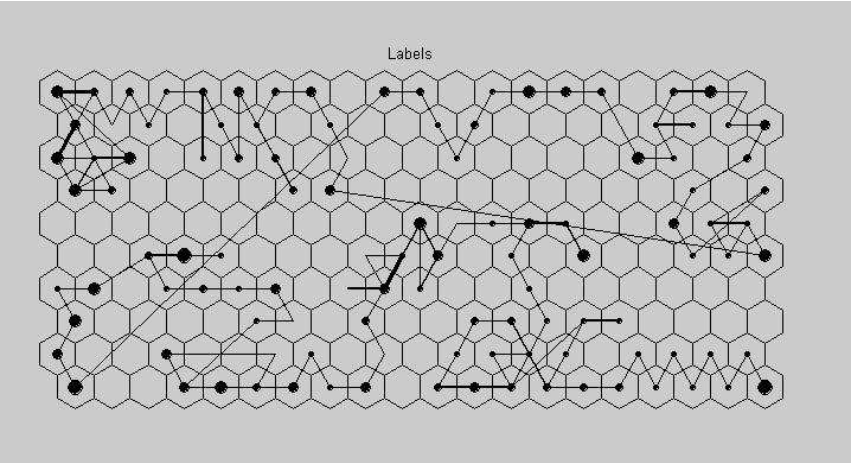

Figure 3.34: Best matching unit trajectory (Case 2)

is a special situation: neither themselves nor the primary neighbors have valid label values. As described in Case 1, the solution is using the label values of the secondary neighbours.

In Fig.3.35, maximum hit value is 16. This value is important for finding the formula defining the relation between the label value and the minimum threshold value. When a router receives a reply packet, the minimum threshold value is updated due to the formula:

$$min_{th} = 100 - \alpha_2 * label_{final_1} \tag{3.12}$$

This formula shows that min_{th} takes values in the region [4 100] packets when α_2 is chosen as 6. The relation between minimum threshold and $label_{final_1}$ is seen in Fig.3.37.

Scenario 3: Minimum threshold value is not affected by the $label_{final_1}$ value, its value is kept constant at 100 packets level. $label_{final_1}$ value is used by the receiving router, to determine max_p. $max_{p_{den}}$ vs $label_{final_1}$ is given in Fig.3.39.

Scenario 4: When $label_{final_1}$ value is obtained by the observation unit and is sent to a router, not only min_{th} but also max_p value changes. The amount of changes in min_{th} and max_p may be determined from Fig.3.37 and Fig.3.39.

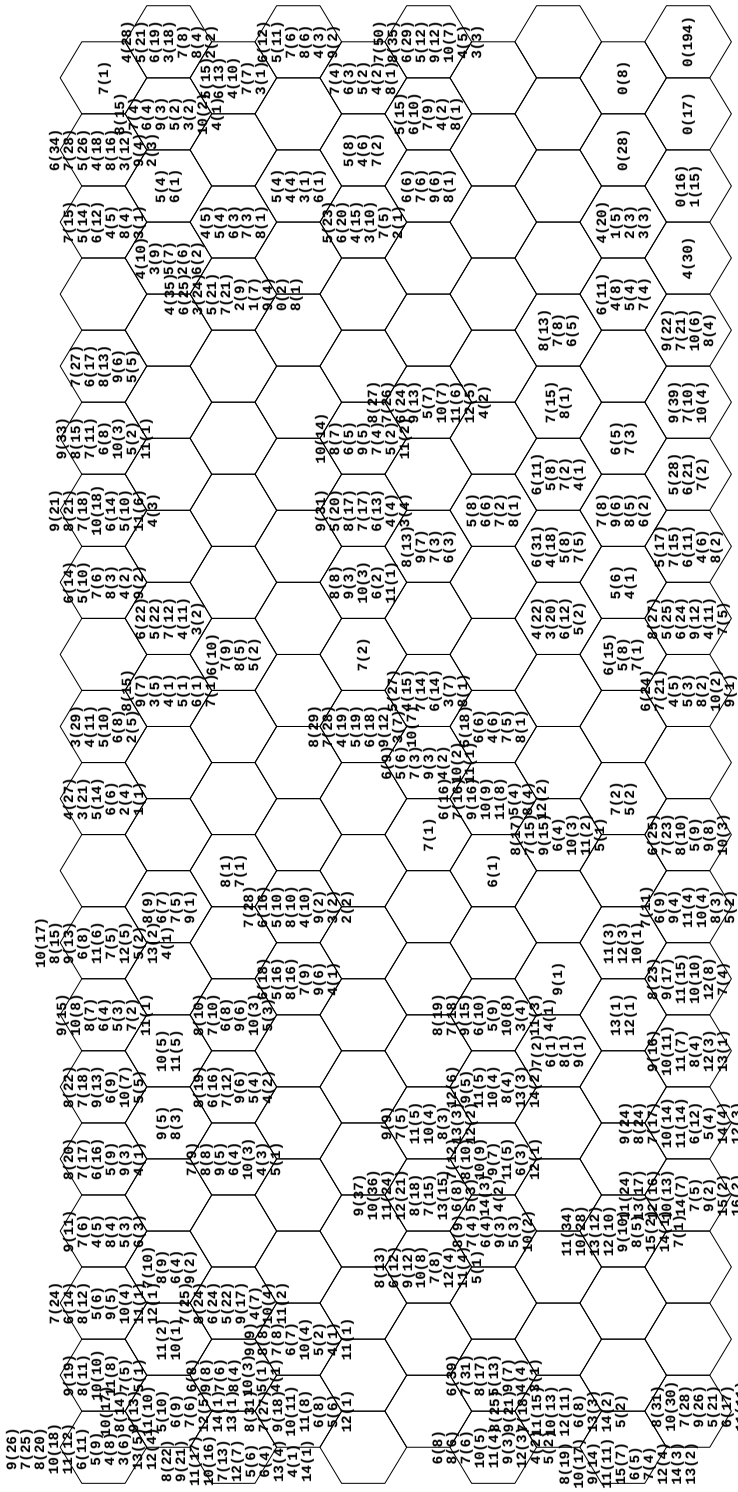

Figure 3.35: Hit and frequency values of the neurons for Case 2 (map is rotated counterclockwise by 90°)

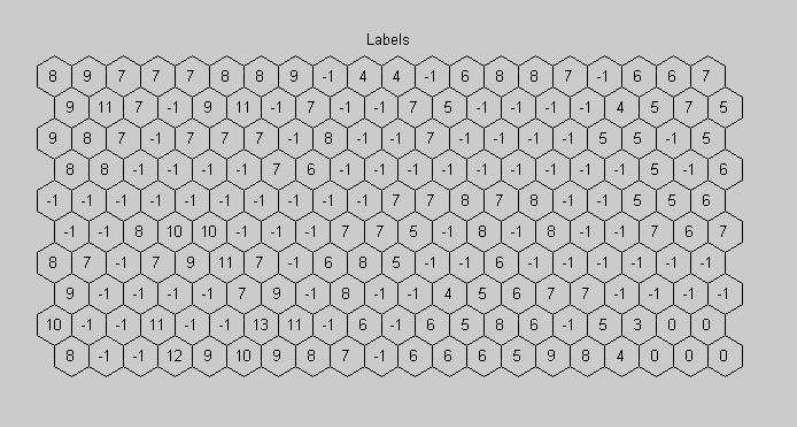

Figure 3.36: Labels obtained by using weighted averaging method (Case 2)

Scenario 5: Just as in the fifth scenario in Case 1, a probabilistic way is followed to obtain the label values and the final label value, $label_{final_2}$, which will be sent to routers from the observation unit. When $label_{final_2}$ is obtained by a router, min_{th} is calculated by using the formula:

$$min_{th} = 100 - 6 * label_{final_2} \tag{3.13}$$

Scenario 6: At the observation unit, a probabilistic way is followed to obtain the label values and $label_{final_2}$. At the router, max_p value is updated due to the value of $label_{final_2}$ as seen in Fig.3.39. min_{th} is kept constant during simulation.

Scenario 7: Just as in Scenarios 5 and 6, label values and $label_{final_2}$ are obtained in a probabilistic way. $label_{final_2}$ is used to update both of min_{th} and max_p (Fig.3.37 and Fig.3.39 show the updated values).

As in Case 1, a simulation of 70 seconds is performed for each scenario. Results of the simulations are given in Fig.3.40 - Fig.3.45. Fig.3.40 - Fig.3.44 are for observation of some global statistics, while Fig.3.45 shows how the length of a queue changes with time. There are 7 graphics in each figure and each graphic belongs to a different scenario.

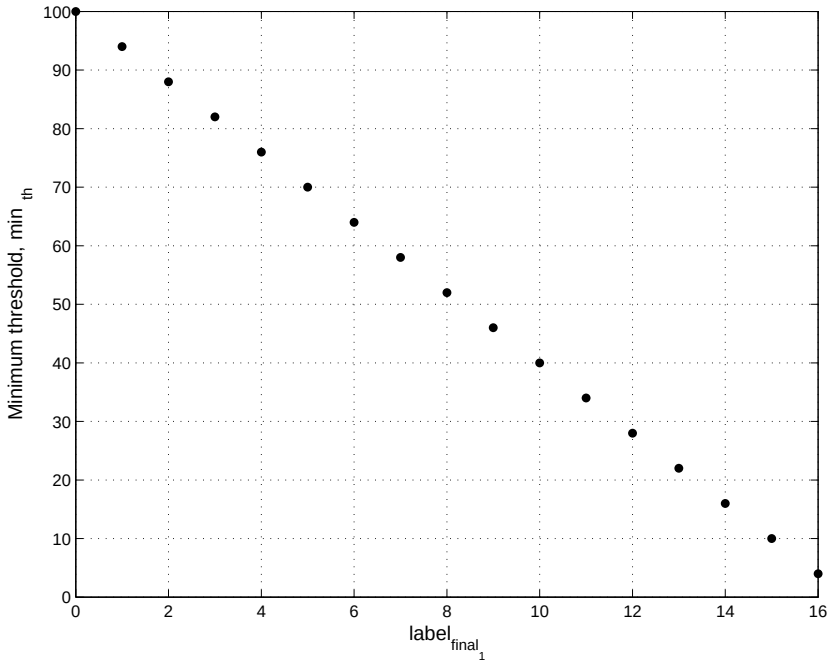

Figure 3.37: min_{th} vs $label_{final_1}$ (Case 2)

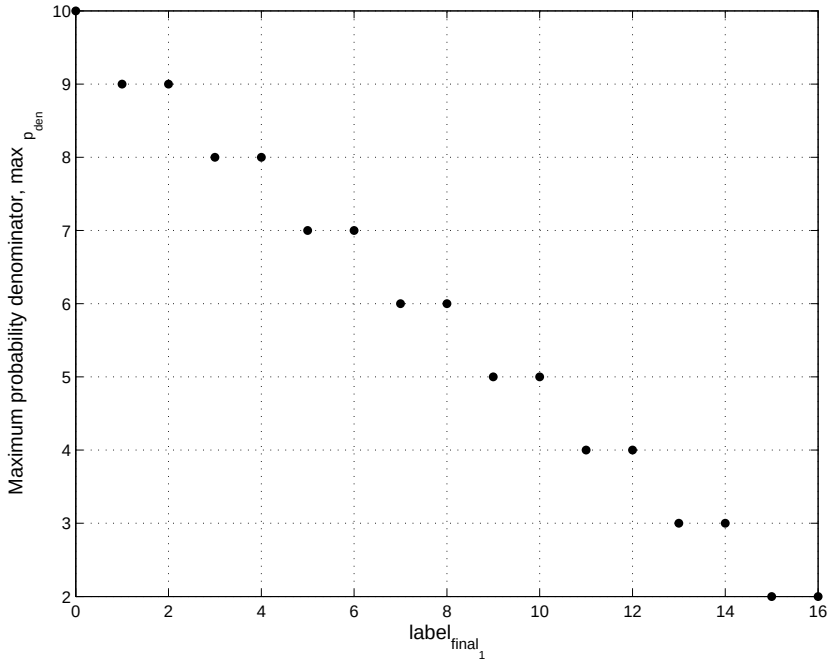

Figure 3.38: $max_{p_{den}}$ vs $label_{final_1}$ (Case 2)

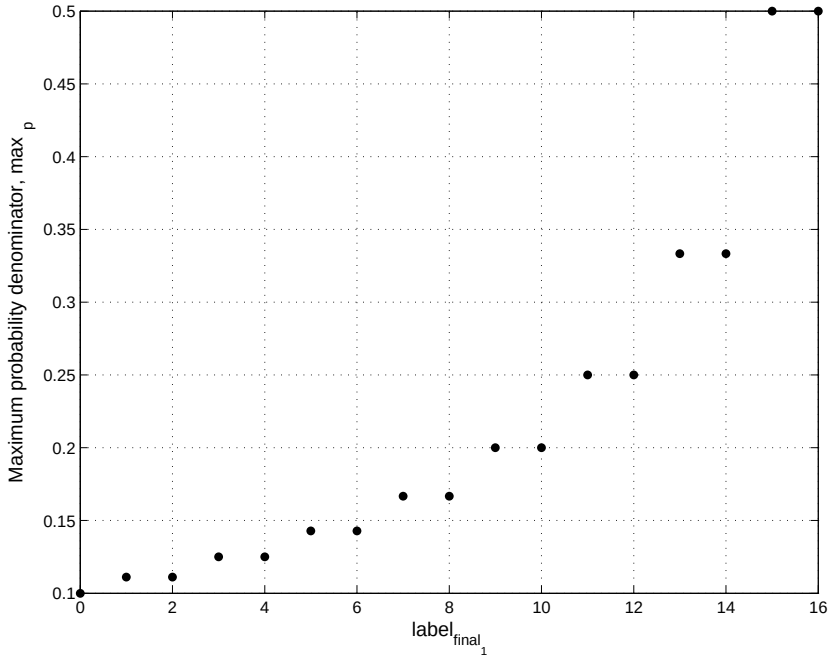

Figure 3.39: max_p vs $label_{final_1}$ (Case 2)

Table 3.5: Summary of Scenarios 1 - 4 of Case 2

Property	**Scenario 1**	**Scenario 2**	**Scenario 3**	**Scenario 4**
Queue management technique	RED	Approach1	Approach1	Approach1
label for congestion notification	—	$label_{final_1}$	$label_{final_1}$	$label_{final_1}$
label determination method	—	weighted averaging	weighted averaging	weighted averaging
buffer limit (pkts)	300	300	300	300
min_{th} (pkts)	100	updated due to (3.12)	100	updated due to (3.12)
max_{th} (pkts)	200	200	200	200
$max_{p_{den}}$ (pkts)	10	10	updated as in Fig.3.38	updated as in Fig.3.38

Table 3.6: Summary of Scenarios 5 - 7 of Case 2

Property	**Scenario 5**	**Scenario 6**	**Scenario 7**
Queue management technique	Approach1	Approach1	Approach1
label for congestion notification	$label_{final_2}$	$label_{final_2}$	$label_{final_2}$
label determination method	probabilistic	probabilistic	probabilistic
buffer limit (pkts)	300	300	300
min_{th} (pkts)	updated due to (3.13)	100	updated due to (3.13)
max_{th} (pkts)	200	200	200
$max_{p_{den}}$ (pkts)	10	updated as in Fig.3.38	updated as in Fig.3.38

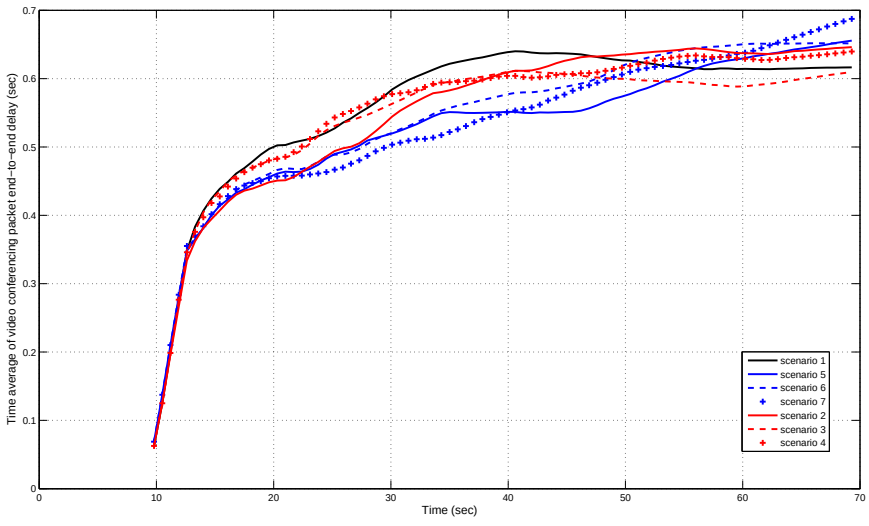

Figure 3.40: Time average of video conferencing packet end-to-end delay (Case 2)

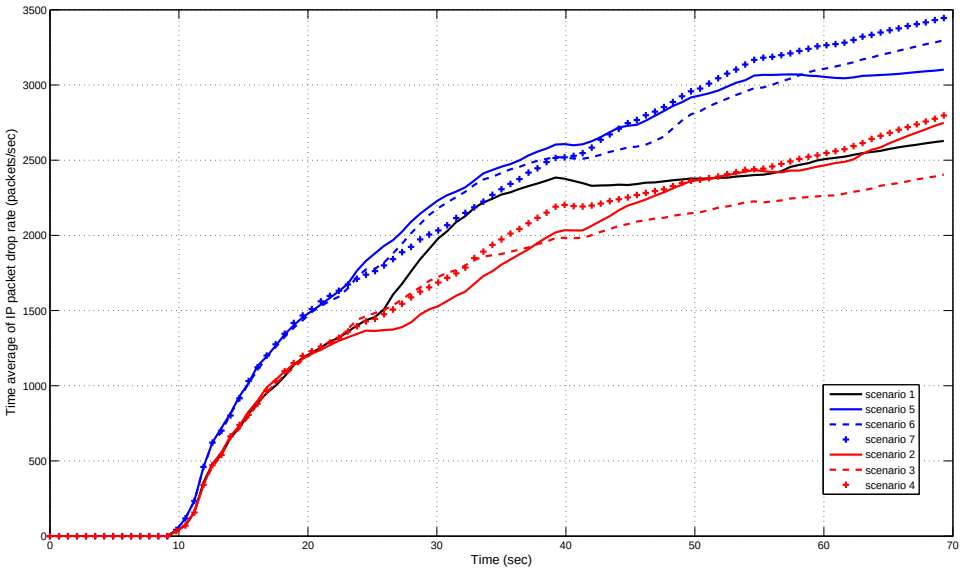

Figure 3.41: Time average of IP packet drop rate (Case 2)

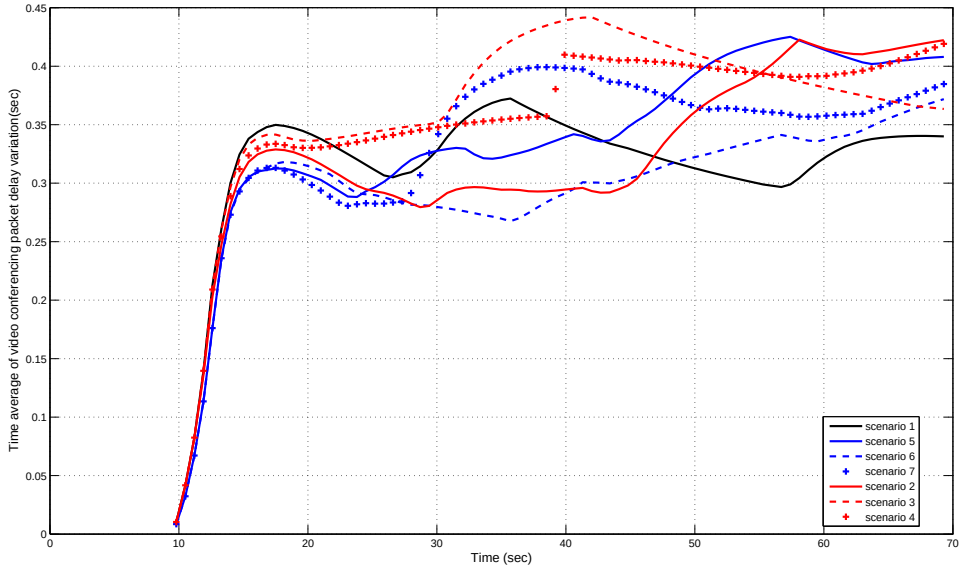

Figure 3.42: Time average of video conferencing packet delay variation (Case 2)

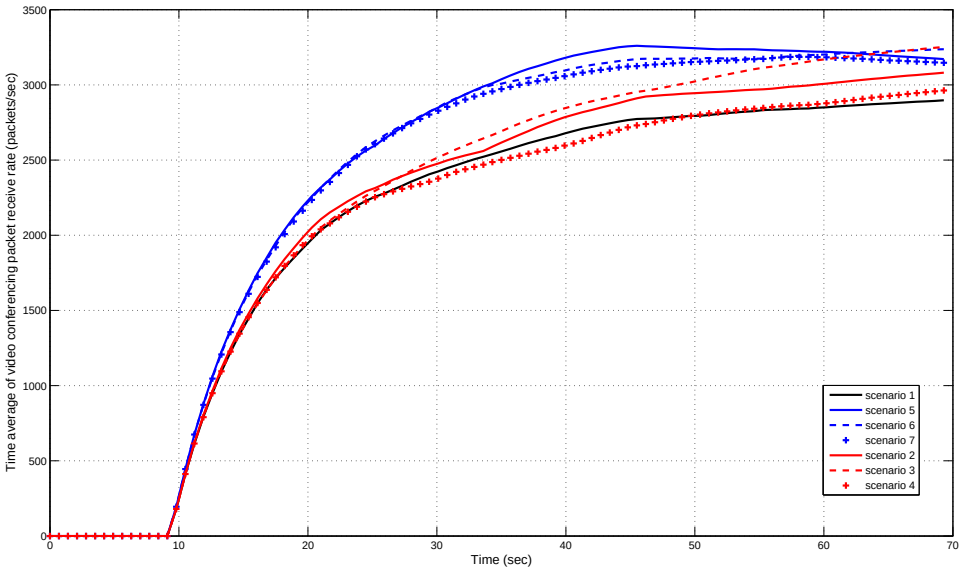

Figure 3.43: Time average of video conferencing packet receive rate (Case 2)

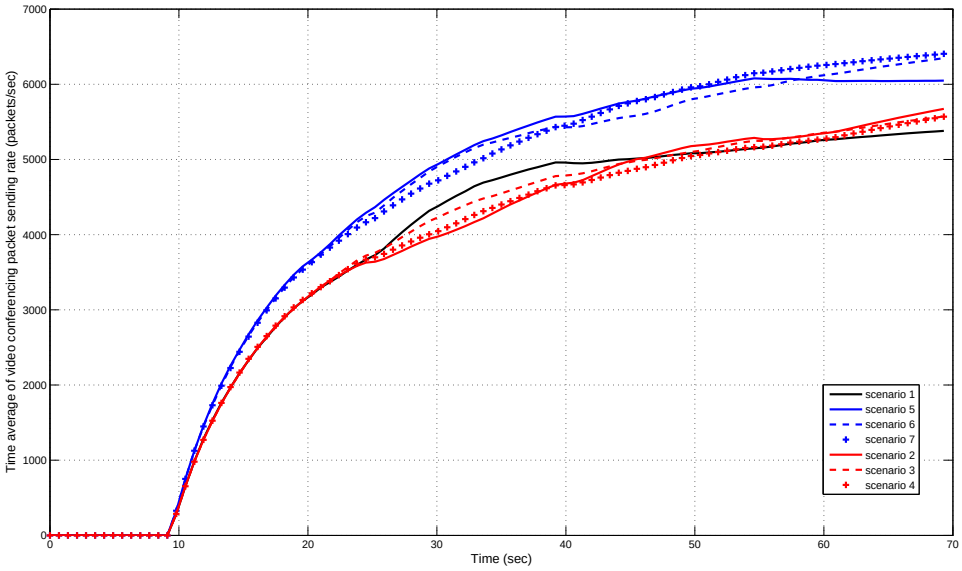

Figure 3.44: Time average of video conferencing packet sending rate (Case 2)

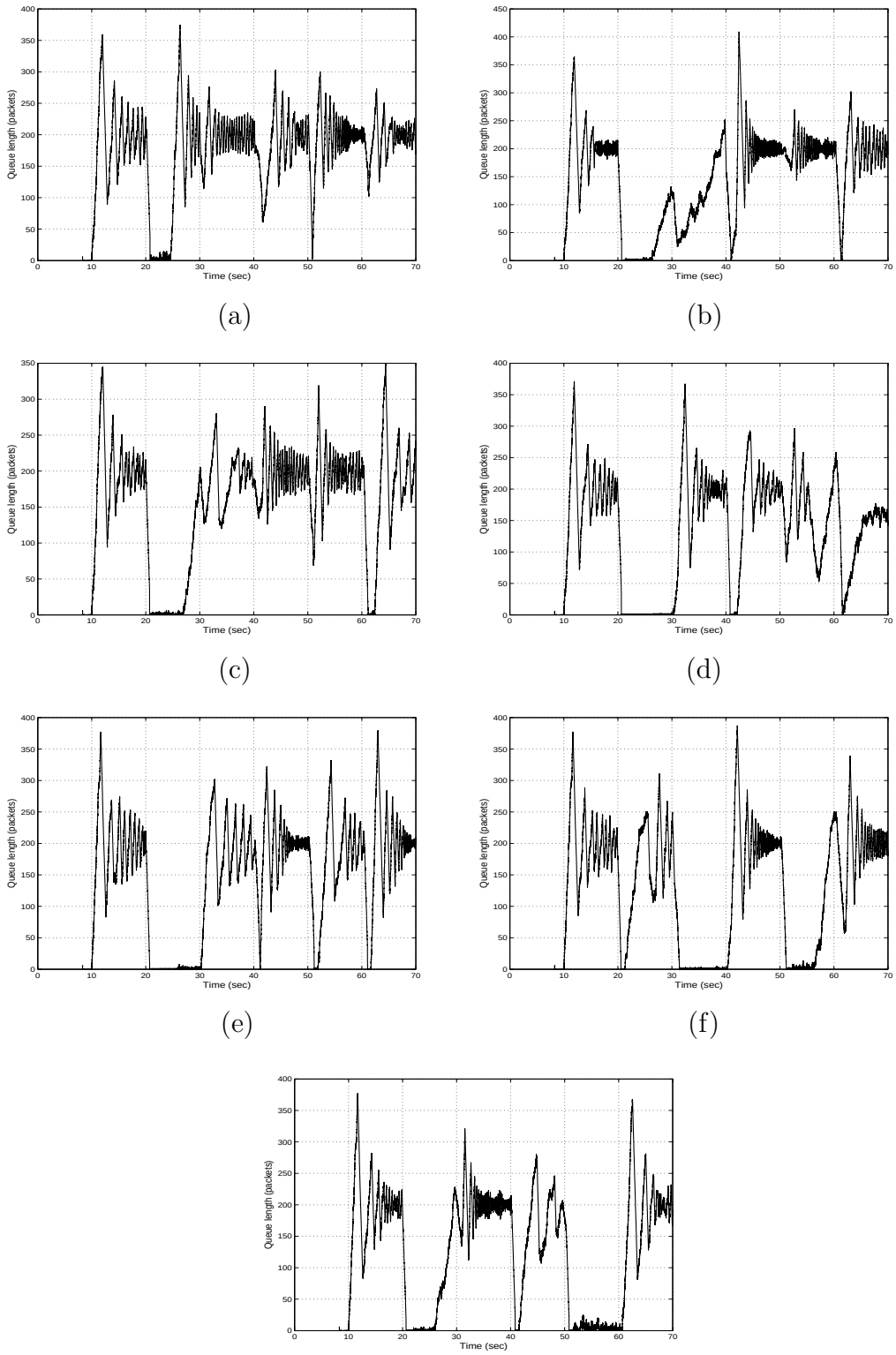

Figure 3.45: Queue length at the interface with IP address: 192.0.16.2 for Case 2. The graphics are for (a) Scenario 1, (b) Scenario 2, (c) Scenario 3, (d) Scenario 4, (e) Scenario 5, (f) Scenario 6, (g) Scenario 7.

Table 3.7: Video conferencing packets received (r) to sent (s) ratio (Case 2)

scenario no	Time average of traffic sending rate (pkts/sec) at 70^{th} second	Time average of traffic receiving rate (pkts/sec) at 70^{th} second	r/s
1	5380	2897	0.54
2	5673	3080	0.54
3	5572	3252	0.58
4	5572	2962	0.53
5	6048	3171	0.52
6	6348	3237	0.51
7	6405	3146	0.49

As seen in the figures and tables for Case 2, time averaged end-to-end delay values of Scenario 3 are less than that of Scenario 1 during simulation. End-to-end delay performances of other scenarios are not better than that of Scenario 1. Maximum ratio of receiving rate to sending rate is also obtained in Scenario 3.

After obtaining the results for Case 1 and Case 2, it is easy to see the differences between the scenarios. There are some common points for two cases:

1. Time-averaged value of IP packet drop rate is minimum in Scenario 3.
2. Time-averaged values of end-to-end delays in Scenario 3 are less than the values in Scenario 1. However, it is not possible to say that Scenario 3 is better than Scenario 1, in case of delay variations.
3. Scenarios 3 gives better throughput results than Scenario 1, while the throughput performances of Scenarios 2, 4, 5, 6 and 7 vary with changing traffic conditions.
4. As a result of the observations above, Scenario 3 should be preferred to others. The properties of this scenario are:

- In investigating SOM, weighted label values are used to estimate future congestion level
- $max_{p_{den}}$ value is changed in order to avoid congestion.

Therefore, it is apparent that the approach in Scenario 3 is successful in improving end-to-end delays and throughput. However, the approach is not as good as original RED, that is presented in Scenario 1, in minimizing delay variations. A new approach, Approach 2, is proposed for the aim of improving RED and providing a better performance than Approach 1.

3.2.3 Approach 2

Throughput and utilization are related characteristics of a link. Qelay and delay variation are two properties that are emphasized in determining QoS requirements. It is a favorite research area to find a trade off between throughput and delay values, that is minimizing delays/delay variations without causing much decrease in throughput. This is, in fact, like combining passive and AQM methods. There are studies proposed on this topic: [58] presents simulation results for the comparison of two queue management algorithms: Drop Tail (DT) and RED; [59] explains advantages of both algorithms and foresees that combination of these two methods might give better results. In Approach 2, we are focused on the idea of using advantages of DT and RED.

Approach 1 and Approach 2 are similar in some ways: both of them investigate the global congestion data, use SOM for future prediction and update some of the RED parameters. However, there are also significant differences between the two approaches: in Approach 2, statistical analysis of the collected data is performed through calculation of mean, variance, skewness and kurtosis. Results of these calculations are used to train a SOM. It is noticed that the regions of previous and current BMU in the SOM are useful indicators for future congestion status of the network. Assignment of RED parameters is based on a different idea, that is combining the advantages of active and

passive queue management techniques. The details about this approach will be explained later in this section. Maximum threshold (max_{th}) is a parameter which may be varied to make RED behave more like DT. Therefore, max_{th} is varied due to changing congestion status of the network. The performance of this approach is verified through a set of simulations.

Approach 2 may be summarized in eight steps which are explained below:

Step 1: Observing the queue lengths on the router output interfaces during a period of time.

We are interested in the global congestion status of the network. A simple way for determining this status is observing the total amount of queue lengths on all the interfaces and building a list of sums:

$$sum_k = q1_k + q2_k + \cdots + qf_k \tag{3.14}$$

where sum_k is the k^{th} sample among the list of observed sums, qm_k is the k^{th} sample of the queue length at output interface m, $m = 1, 2, ..., f$)

Step 2: Producing groups by taking Z consecutive values from the list of sums, then performing statistical analysis (by means of mean, variance, skewness, kurtosis values of each group).

A window of size Z is shifted by $P(P < Z)$ values each time a group is produced, therefore each group is made to have $Z - P$ common values with the previous group.

$$g_i = \left\{sum_{(i-1)P+1}, sum_{(i-1)P+2}, \cdots sum_{(i-1)P+Z}\right\} \tag{3.15}$$

where g_i is the i^{th} group of sums.

Mean of the i^{th} group is obtained by using the following equation, where x_{ij} is the j^{th} element of the i^{th} group (where $i = 1, 2, ...m$):

$$mean_i = \frac{1}{Z}\sum_{j=1}^{Z} x_{ij} \tag{3.16}$$

Table 3.8: Table for mean, variance, skewness and kurtosis values

i	$mean_i$	$variance_i$	$skewness_i$	$kurtosis_i$
1				
2				
$\vdots$				
m				

Variance of a set of Z observations is defined as the sum of the squared deviations of the observations from the mean divided by Z-1 [60]. Variance of the i^{th} group is obtained by using the following equation:

$$variance_i = \frac{1}{Z-1}\sum_{j=1}^{Z}(x_{ij} - mean_i)^2 \tag{3.17}$$

Skewness is an indication of the extent of deviation from symmetry [60]. Skewness of the i^{th} group is obtained by using the following equation:

$$skewness_i = \frac{\frac{1}{Z}\sum_{j=1}^{Z}(x_{ij} - mean_i)^3}{\left[\sqrt{\frac{1}{Z}\sum_{j=1}^{Z}(x_{ij} - mean_i)^2}\right]^3} \tag{3.18}$$

Kurtosis is a degree of peakedness of a distribution [60]. Kurtosis of the i^{th} group is obtained by using the following equation:

$$kurtosis_i = \frac{\frac{1}{Z}\sum_{j=1}^{Z}(x_{ij} - mean_i)^4}{\left[\sqrt{\frac{1}{Z}\sum_{j=1}^{Z}(x_{ij} - mean_i)^2}\right]^4} \tag{3.19}$$

Table 3.8 is obtained by using the mean, variance, skewness and kurtosis values of m groups:

Step 3: Normalizing the mean, variance, skewness and kurtosis values due to variance criteria.

Step 4: Appending 'label's to each group (to each row in Table 3.8) of mean, variance, skewness and kurtosis values, in order to label them with respect to the congestion level they represent. The higher the label, the greater the

congestion.

Step 5: Building the SOM structure and performing the training process.

Step 6: Obtaining regions of congestion on the trained map by the help of labels used in Step 4. Fig.3.67 shows an example for partitioning a map into regions. Here, the regions R1, R2, R3 are composed of the BMUs for data of uncongested, normally congested and highly congested network conditions, respectively.

Step 7: Studying the regions of BMU for previous/current data and describing the congestion behaviour of the network. Here, we need to define a new parameter: congestion alarm number. As its name indicates, the value of this parameter shows the congestion behaviour of the network. Its value varies from 7 to 1 as the congestion shows tendency to increasing, as shown in Table 3.9. It is better to explain the assignment of alarm numbers with an example:

Alarm numbers 1-3 indicate that congestion needs to be considered seriously and they are used when the region of current BMU is R3 (region of high congestion); if the region of previous BMU is R1, alarm number is set to 3 according to the idea that the congestion may be caused by a bursty traffic and it may be accomodated for a while; but if the region of previous BMU is also R3, alarm number is set to 1 in order to prevent sustained congestion.

The last step of the procedure is about RED parameter estimation and it will be explained in Step 8.

Step 8: Adjustment of RED parameters at router output queues. This step is about implementing a queue management procedure by using advantages of RED and Drop Tail. First of all, it will be useful to study the throughput and delay behaviours of a congested queue, under the management of DT and RED. Later both schemes will be used to manage an uncongested queue and the results will be compared.

Table 3.9: BMU regions and congestion alarm numbers

Region of previous BMU	Region of current BMU	Congestion alarm number (from 1(serious) to 7(not serious at all))
R1	R1	7
R1	R2	4
R1	R3	3
R2	R1	6
R2	R2	4
R2	R3	2
R3	R1	5
R3	R2	4
R3	R3	1

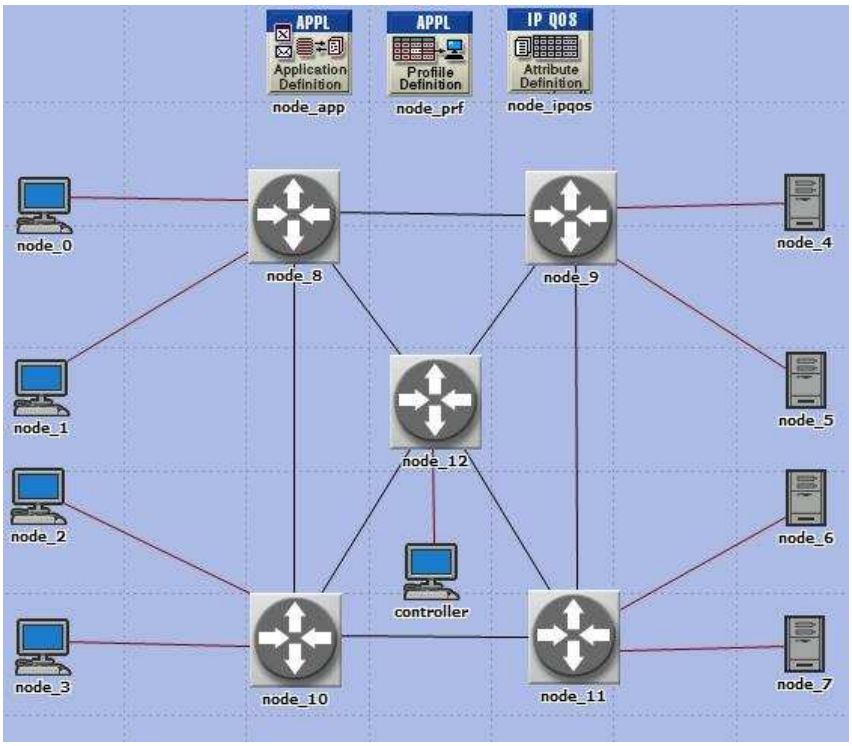

Figure 3.46: Network Model designed for Approach 2

Let's consider a highly congested queue, which is at the output interface of node_8 to the link connecting node_8 and node_9 as shown in Fig.3.46. This queue is investigated for two cases: under management of a passive queue management scheme (DT) and an AQM scheme (RED). For each case, four different simulations are performed and the average of results are obtained. Fig.3.47 is an example for obtaining curve of average values. There are five curves in this figure: data1, data2, data3 and data4 curves are obtained by different simulations; the curve plotted in black is the curve of average values. Fig.3.48 - Fig.3.54 show the curves of average values for both simulation cases (DT and RED management). Except Fig.3.48, all the plots are generated by finding the moving averages within a window of size 5. In Fig.3.48 - Fig.3.54, curves in black are for DT and curves in blue are for RED. The congestion level of the queue may be understood by investigating the queue length and the drop rate curves in Fig.3.48 and Fig.3.49, respectively. The queue length of DT is raised to the level of 300 packets and is not allowed to exceed it. In RED, queue length oscillates around the maximum threshold value and does not reach the buffer limit of 300 packets, as a result of early drops. Packet drop rates of both RED and DT are similar in the first 17 sec., after that the drop rates are decreased and the drop rate of DT becomes less than RED.

Traffic receive rate of an output interface is the rate of traffic that has reached the output queue of packets that will leave the router soon. In both RED and DT scenarios, output interface receives packets at similar rates as seen in Fig.3.50. Traffic sent by the interface is the throughput of the link between nodes 8 and 9, as seen in Fig.3.51. Throughput of DT becomes greater than RED after 17 sec.

The queuing delays are in Fig.3.52. RED has a lower queuing delay than DT.

The queue delay variations are seen in Fig.3.53. As seen in Fig.3.54, link utilizations of RED and DT are close to one another.

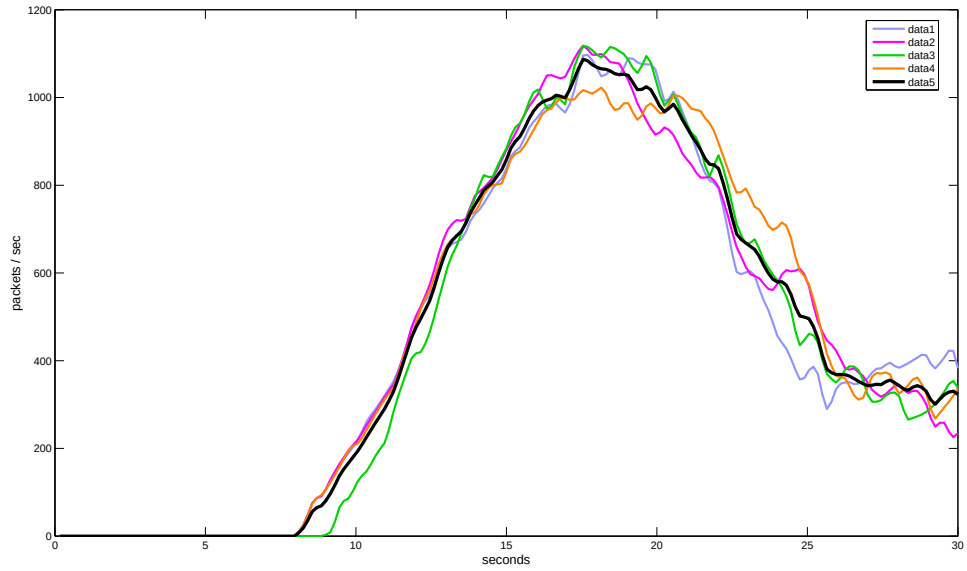

Figure 3.47: Moving average of traffic drop rates for four different simulations and the curve of average values (results for RED applied congested interface)

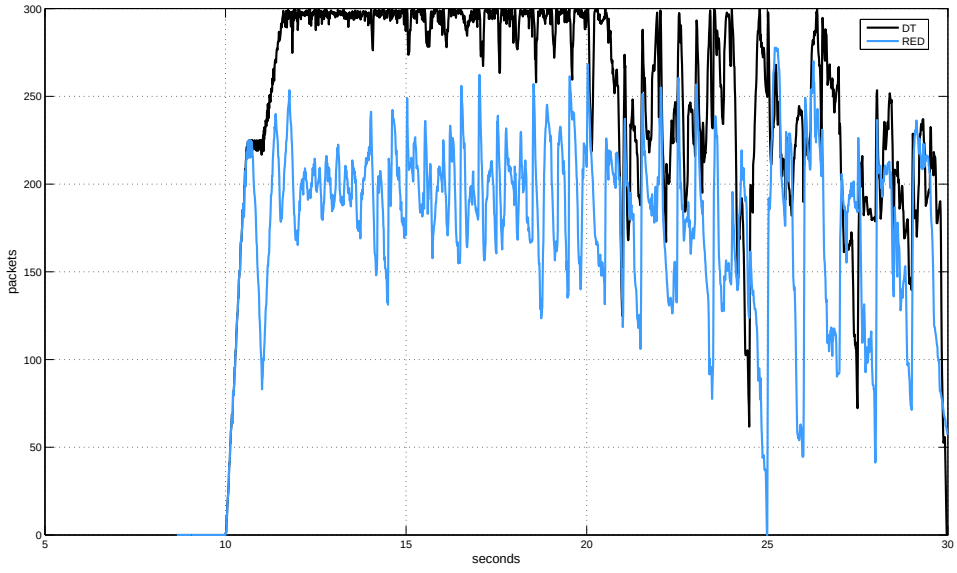

Figure 3.48: The queue length (congested interface)

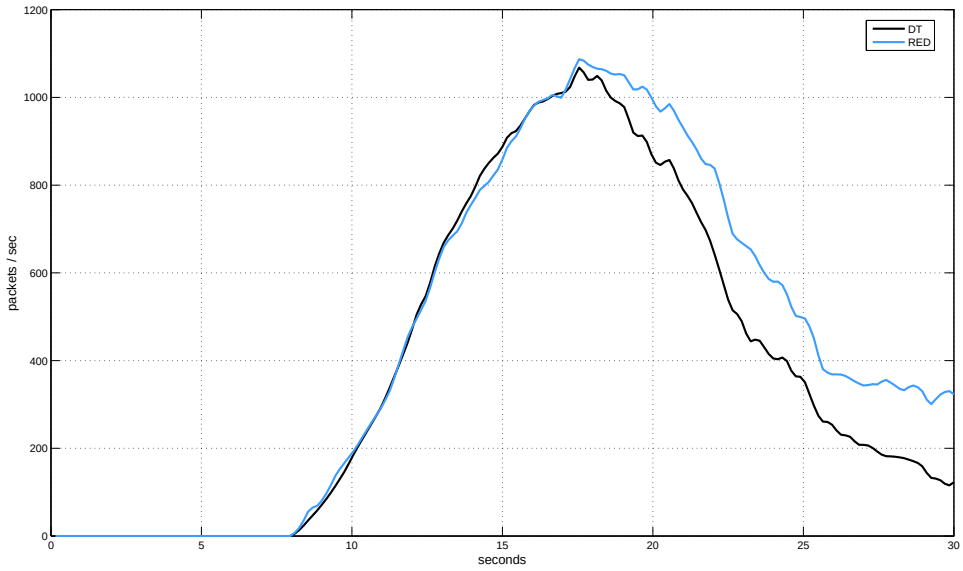

Figure 3.49: Moving average of traffic drop rate (congested interface)

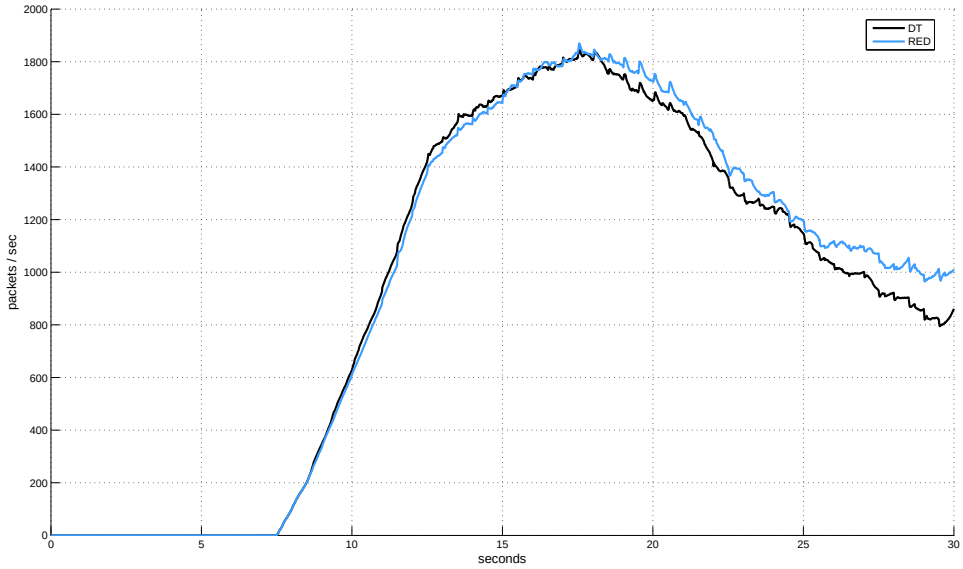

Figure 3.50: Moving average of traffic receive rate (congested interface)

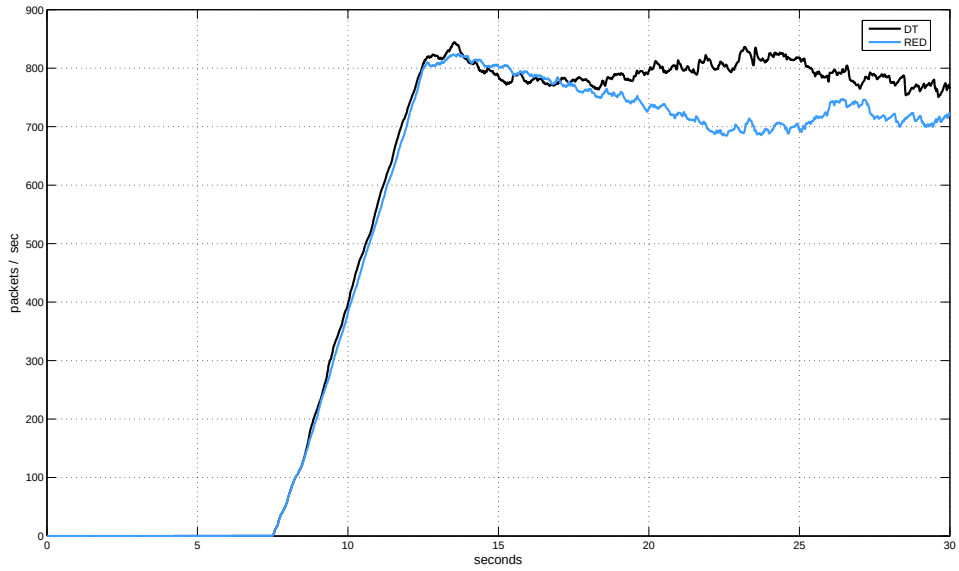

Figure 3.51: Moving average of throughput (congested link)

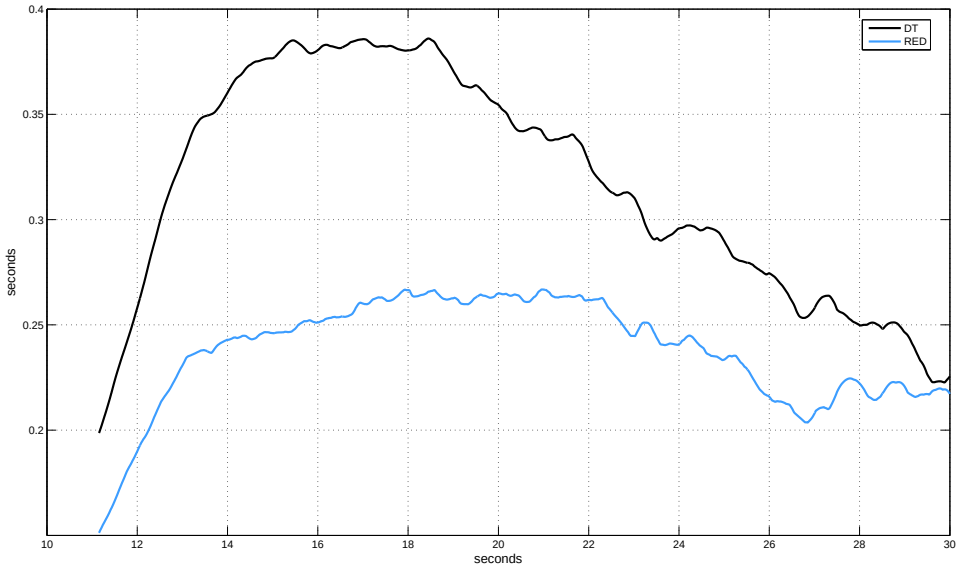

Figure 3.52: Moving average of queuing delay (congested interface)

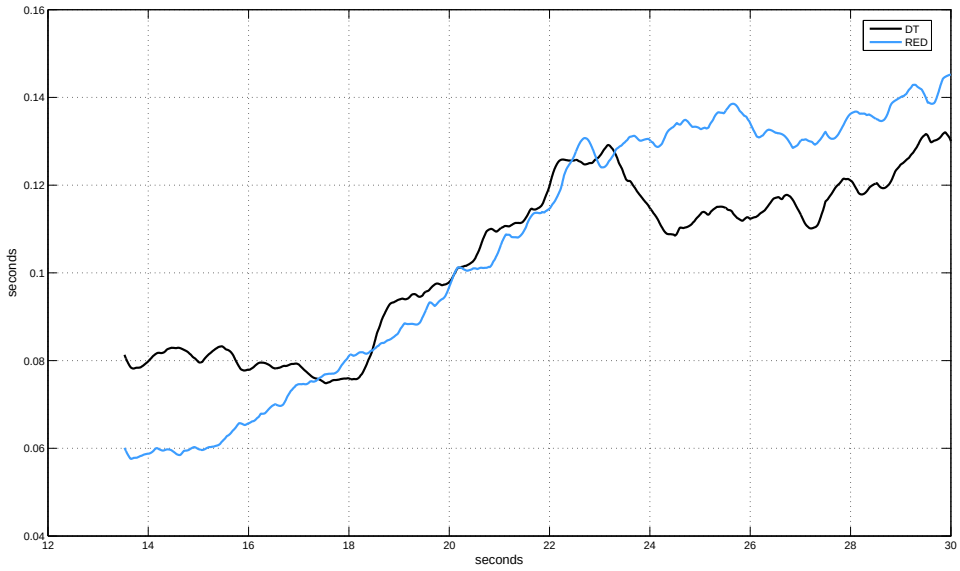

Figure 3.53: Moving average of queuing delay variation (congested interface)

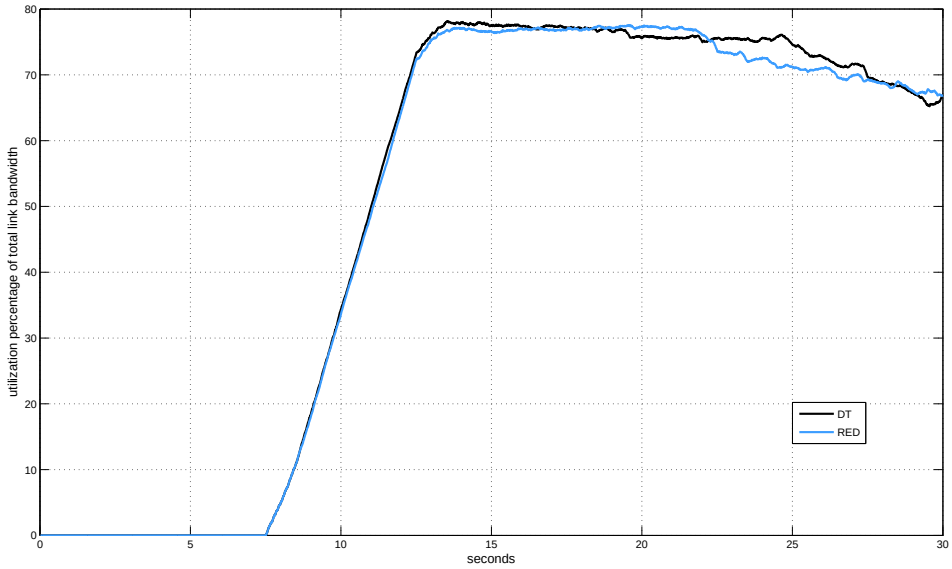

Figure 3.54: Moving average of link utilization (congested link)

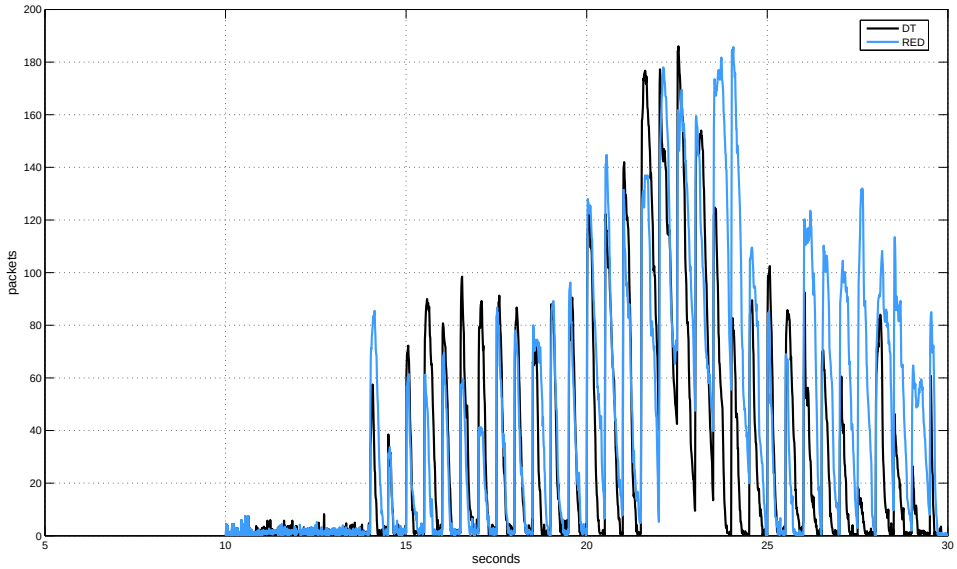

Figure 3.55: The queue length (uncongested interface)

Now, let's investigate an interface in uncongested status. This is at the output interface of the link connecting nodes 9 and 11, as shown in Fig.3.46.

When Fig.3.55 is investigated, it is seen that the queue lengths are not as high as the ones for the congested queue. None of the queues reach the buffer limit of 300 packets.

Except Fig.3.55, all the plots are generated by finding the moving averages within a window of size 5 values.

There is almost no packet drops as seen in Fig.3.56. There is no packet drops in DT, while there is only a few packet drops for a short time interval in RED.

In both RED and DT scenarios, output interface receives nearly the same amount of traffic as seen in Fig.3.57.

Traffic sent is the throughput of the link between nodes 9 and 11, as seen in Fig.3.58. Throughput results of both methods are similar.

The queuing delays are in Fig.3.59. Queuing delay of DT is smaller than RED.

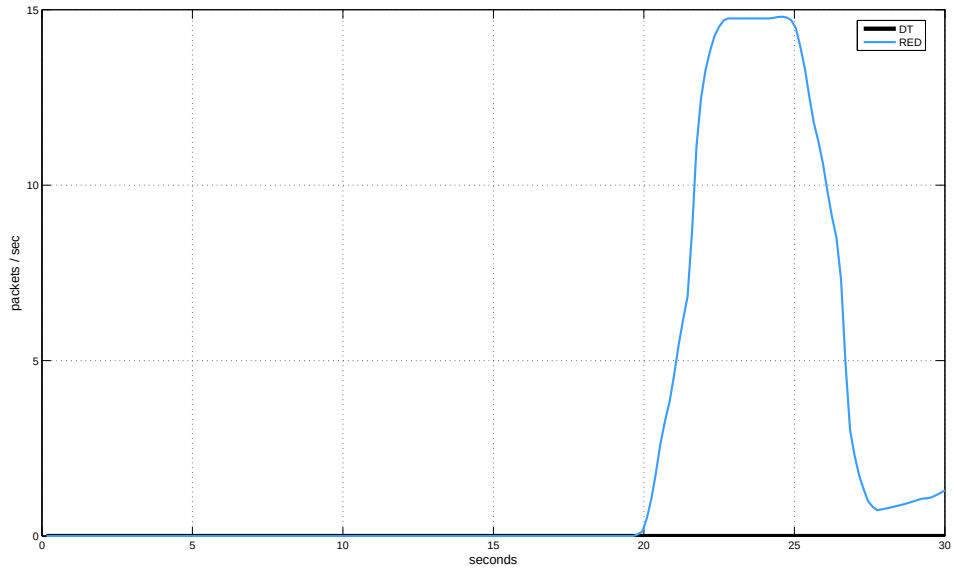

Figure 3.56: Moving average of traffic drop rate (uncongested interface)

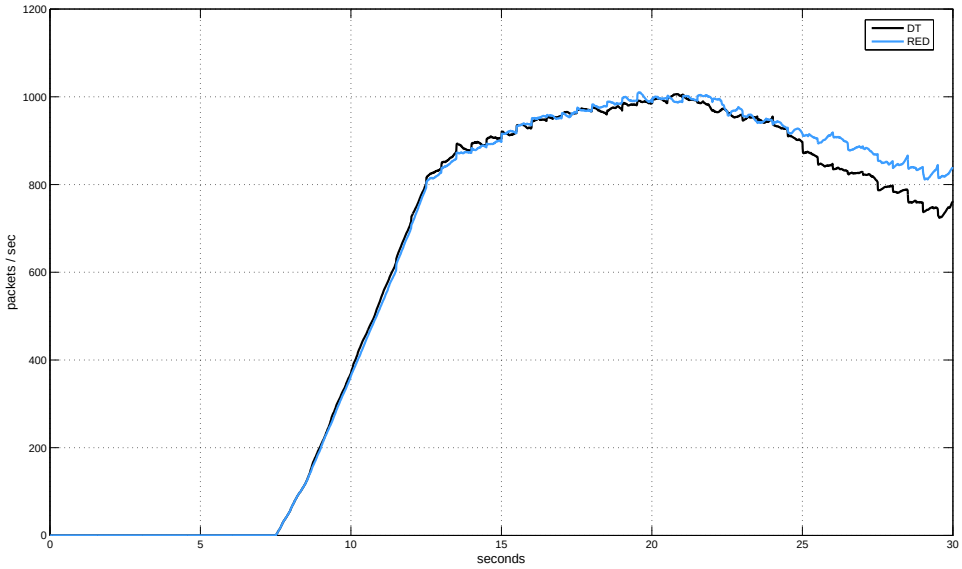

Figure 3.57: Moving average of traffic receive rate (uncongested interface)

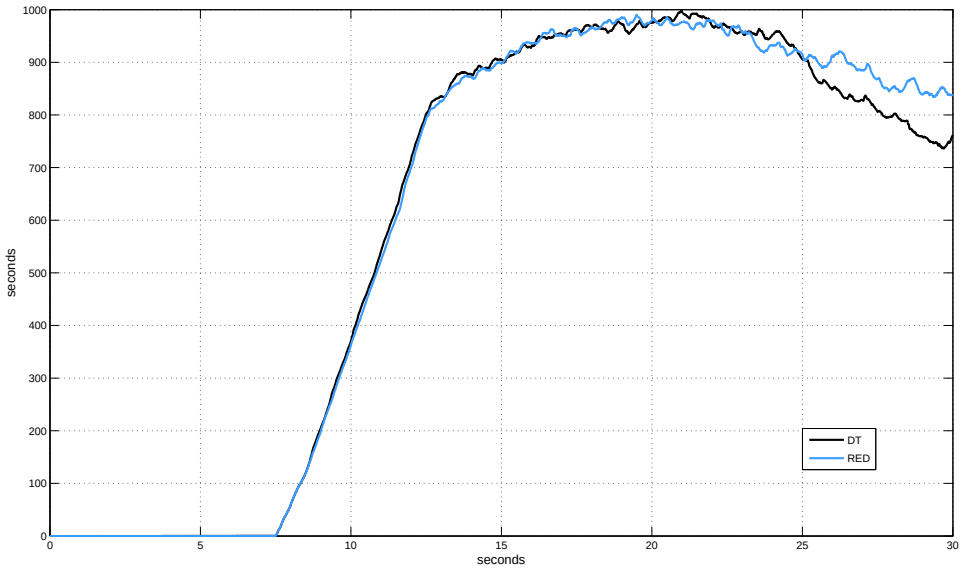

Figure 3.58: Moving average of throughput (uncongested link)

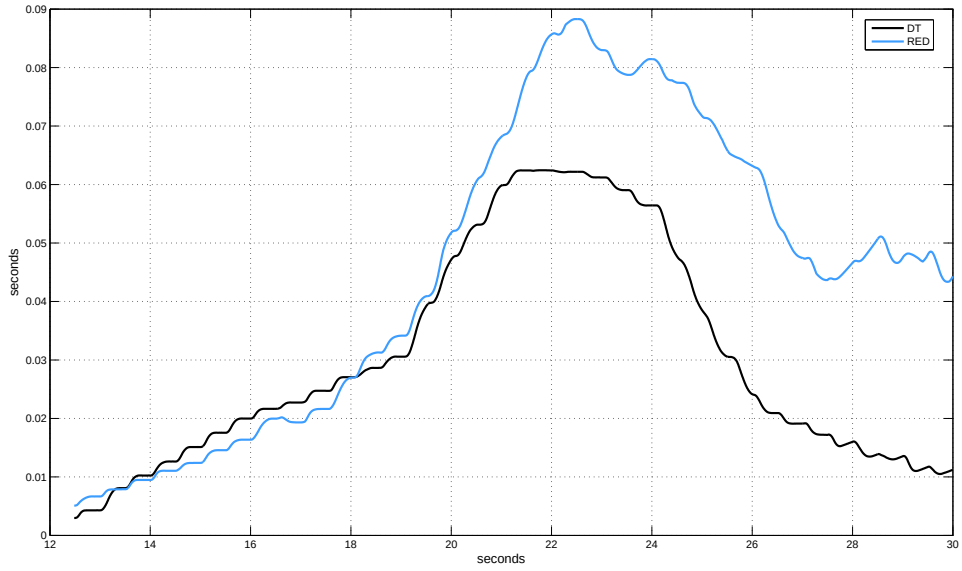

Figure 3.59: Moving average of queuing delay (uncongested interface)

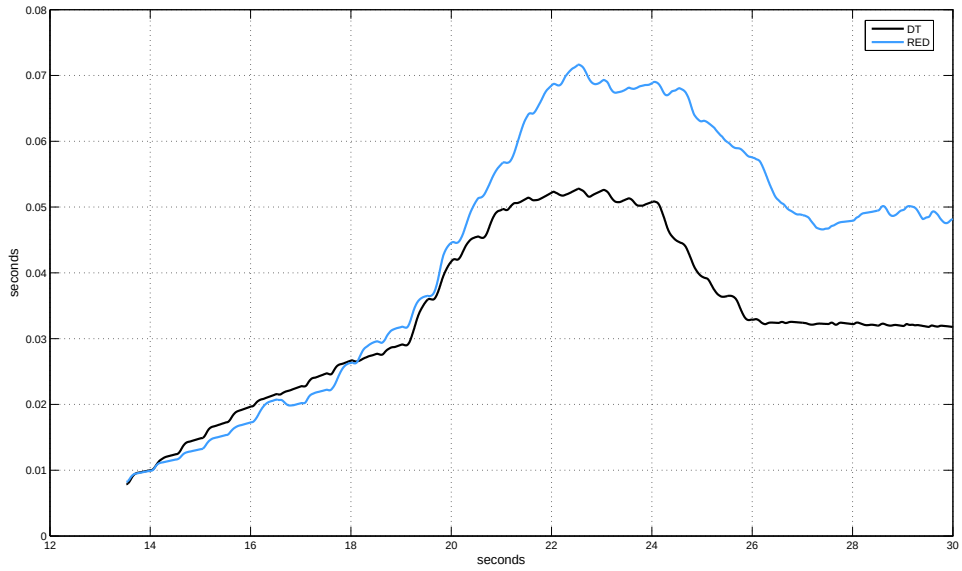

Figure 3.60: Moving average of queuing delay variation (uncongested interface)

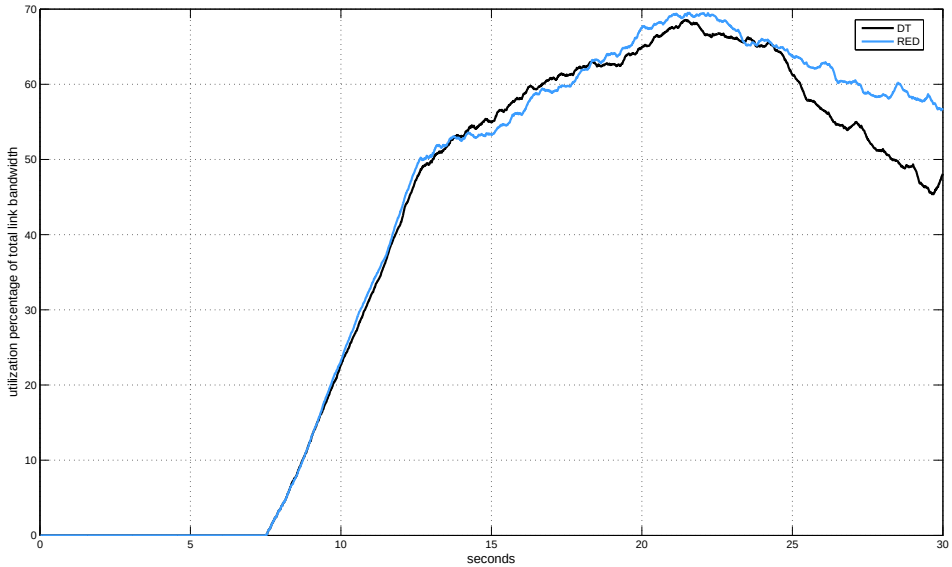

Figure 3.61: Moving average of link utilization (uncongested link)

The queue delay variations are given in Fig.3.60. Before 20 sec., variation curves are similar but after that DT gives smaller values than RED.

Link utilizations are seen in Fig.3.61. The curves are similar for both methods. The simulation results given above show that, a static queue management technique (either it is DT or RED) is likely to present disadvantages, i.e., unnecessary early packet drops or long delays in full queues, under variable congestion conditions. If it could be possible to switch between these two techniques as the conditions change, it seems to be possible to minimize delays. In our recent work [61], a parameter estimation method is presented for making RED adaptive to changing congestion conditions.

In TCP/IP netwoks, a packet drop (or marking) is a notification of congestion for TCP sources. A TCP source adjusts its window size due to the packet dropping (or marking) probability for its packet stream as stated in [13]. Due to our new approach, if the level of congestion is low, the queue management scheme will encourage packet transmission by preventing unnecessary early packet drops.

Approach 2 states that if the queue management technique used by interfaces are adaptive to changes in the global congestion level, it will be possible to minimize delays/delay variations. In this approach, instead of switching between DT and RED, we preferred to make adjustments on RED so that it can dynamically change its parameter values with respect to the congestion level of the network. This will make RED adaptive to global congestion notifications. As the global congestion status moves from high to low, packet dropping strategy of RED will become similar to DT. It is useful to prefer DT to RED in the routers of an uncongested/low congested network, for preventing unnecessary early drops, effectively using network resources, increasing throughput and link utilization. However, if the congestion level of the network is high or increasing, DT queues suffer from full queue and lock out problems, delays and delay variations are increased. RED uses early drop mechanism for keeping average queue length within limits and minimizing delays/delay vari-

ations. In this approach, we used the advantages of both schemes to improve RED. The following explanation is given to summarize the new scheme:

- RED parameters are dynamically changed by routers due to global congestion notifications
- For high/increasing congestion status, RED parameters are changed in order to encourage packet drops.
- For low/decreasing congestion status, RED parameters are changed to encourage packet queueing, drop rate is decreased to make RED similar to DT

Routers update their RED parameters due to changes in congestion condition. The information about the region of BMU is sent from the observation unit to routers, encapsulated in reply packets. The regions of previous $\wp$ current BMUs are investigated in each router and the congestion alarm number is updated. If congestion alarm number is high, RED parameters should be changed for preventing unnecessary early drops and the queue management scheme will be more like DT. If congestion alarm number is small, early drops are necessary for decreasing the number of queued packets and the level of congestion. The value of max_{th} will be linearly increased as the congestion alarm number increases.

In order to observe the performance of new queue management approach, simulations are performed. The network model, that is seen in Fig.3.46, is designed by using OPNET Modeler. The network has five IP routers (node_8, node_9, ..., node_12), four pairs of clients&servers (node_0 & node_6, node_1 & node_7, node_2 & node_4, node_3 & node_5) and a centralized observation unit (controller). Clients, servers and the observation unit are connected to the edge routers by 10BaseT links with a transmission rate of 10Mbps. The links between routers are 'PPP_E1' with a transmission rate of 2.048 Mbps. An ftp traffic is generated between clients and servers. Three simulation scenarios are generated and their properties are explained below.

Scenario 1:

The purpose of this scenario is obtaining data about the congestion behaviour of the network by means of the queue lengths on the router output interfaces. The simulation results will give a general information about the system behaviour when there is no control input destined to affect the AQM algorithms of routers. In this scenario, in addition to the ftp traffic generated between clients and servers, there is a traffic between routers and the observation unit: IP packets carrying the queue length values (that belong to the queues at various output interfaces) are sent to the centralized observation unit. Each of these special packets are originated in one of the routers; carries information about an output interface of this router (the address code of the output interface and the queue length at this interface) and is sent in the first 2ms of each 10ms. In the observation unit, these packets are received, the queue length values are collected and (together with the receipt time) written to a file. Then, reply packets are generated by the observation unit and sent back to the routers in the first 100ms of every 500ms. In Scenario 3, the function of reply packets will be explained in detail, for now it is enough to know that reply peckets are for readjusting some QoS parameters in the routers. In this scenario, DT (FIFO mechanism with disabled RED criteria and a buffer size of 300 packets) is used as the queue management scheme in the routers. At the end of the simulation of Scenario 1, the information received by the observation unit is used to perform the operations described by equations (3.14) - (3.19) in Step 1 and Step 2. In our network model, there are 25 router output interfaces. Groups of sums are produced by taking 100 consecutive sums each of which is calculated by adding the queue lengths of 25 interfaces. A window of size 100 (Z=100) values is shifted by 20 (P=20) values each time a group is produced, therefore each group is made to have 80 common values with the previous group. Mean, variance, skewness, kurtosis of each group are found. The curves for mean, variance, skewness and kurtosis values of the groups are shown in Fig.3.62 - 3.65, respectively.

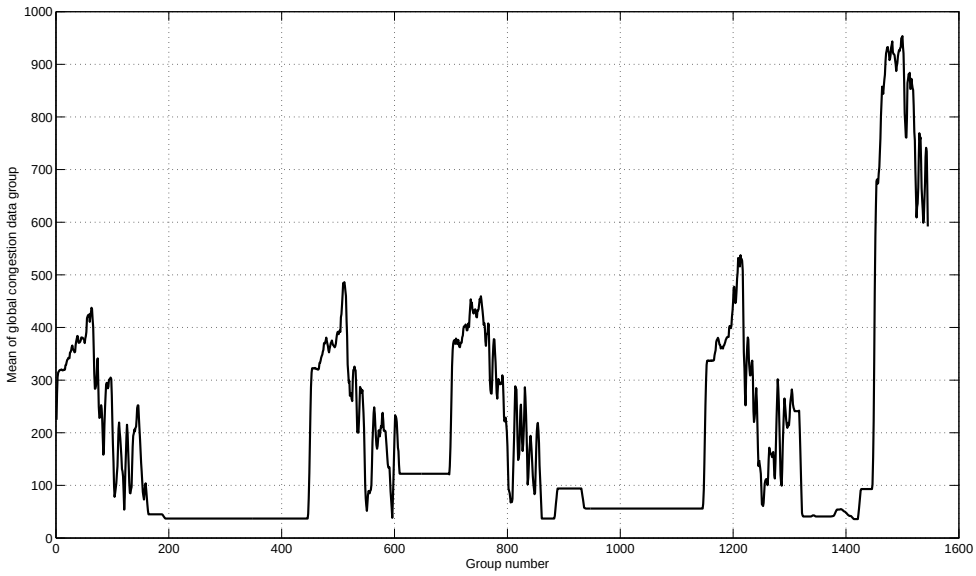

Figure 3.62: Mean of global congestion data group vs group number

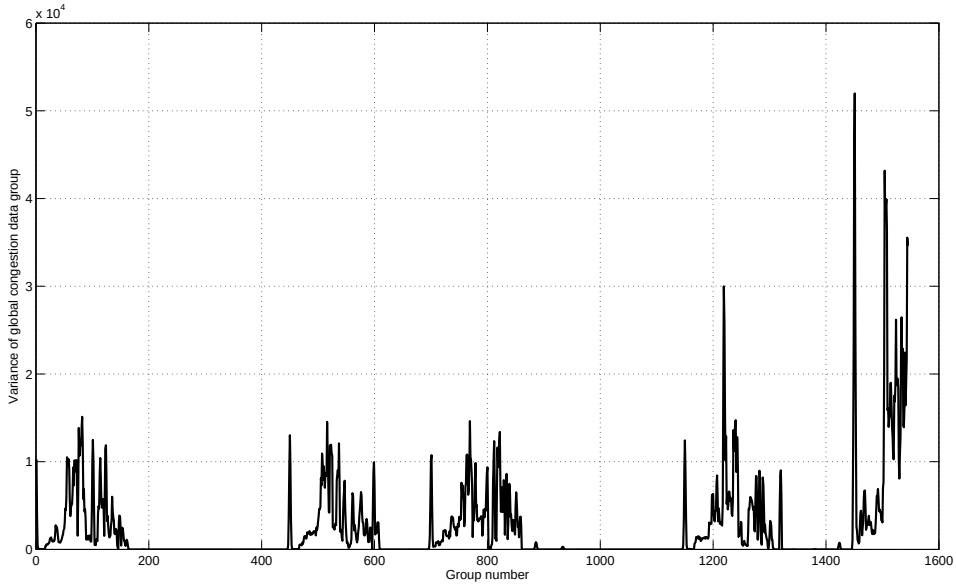

Figure 3.63: Variance of global congestion data group vs group number

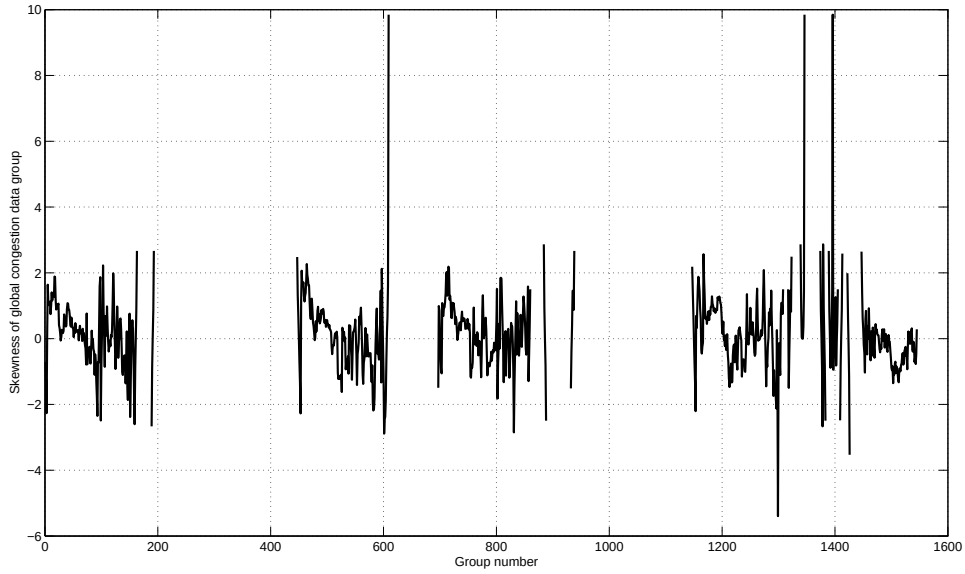

Figure 3.64: Skewness of global congestion data group vs group number

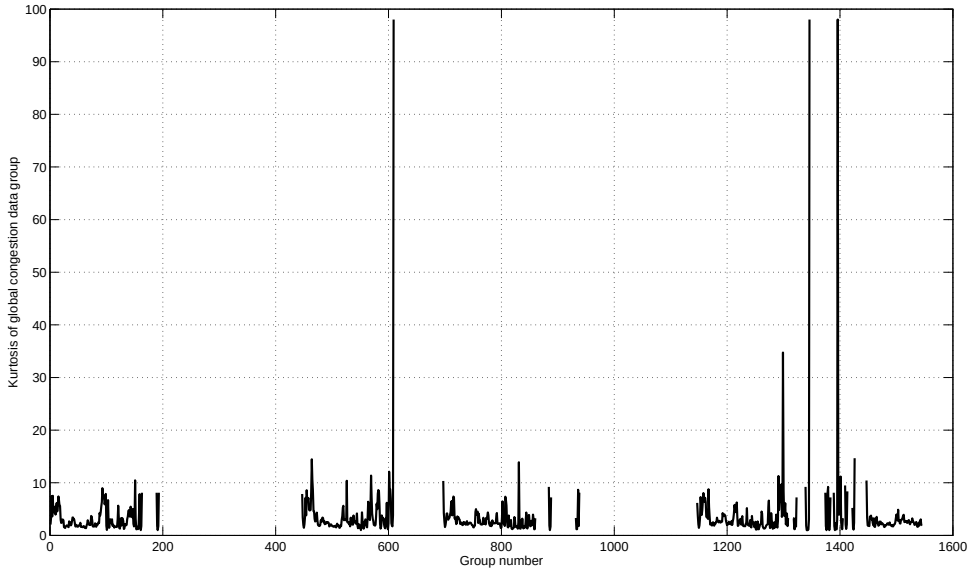

Figure 3.65: Kurtosis of global congestion data group vs group number

Table 3.10: Labels assigned to input vectors with different mean values

$mean_i$ interval	label
$100 > mean_i$	0
$200 > mean_i \geq 100$	1
$300 > mean_i \geq 200$	2
$400 > mean_i \geq 300$	3
$500 > mean_i \geq 400$	4
$600 > mean_i \geq 500$	5
$700 > mean_i \geq 600$	6
$800 > mean_i \geq 700$	7
$900 > mean_i \geq 800$	8
$mean_i \geq 900$	9

As in Step 3, mean, variance, skewness and kurtosis values are normalized due to variance criteria. 4 different normalizations with different mean and variance values took place. 'label's are assigned to input vectors as in Step 4, in order to label them with respect to the congestion level they represent. In Table 3.10, the intervals of $mean_i$ and the corresponding label values are given. By the end of Step 4, 1545 vectors (1x4) are produced to be used as inputs for the SOM and each has a label value representing the global level of congestion. The next phase is the definition of the SOM structure and the training:

- map size: 10x10 (a map of 100 neurons)
- lattice: hexagonal
- shape: sheet
- type of training: sequential (finetune training is preceded by rough training)
- initial training radius: 10 (rough training), 2 (finetune training)
- final training radius: 2 (rough training), 1 (finetune training)
- training length: 500 epochs (rough training and finetune training)

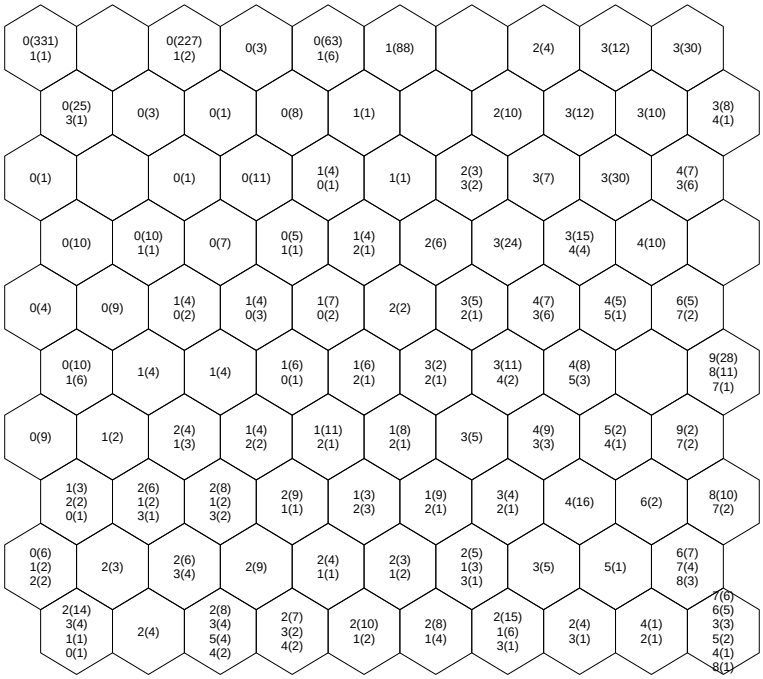

Figure 3.66: Hit and frequency values of the neurons

• initial learning rate: 0.5 (rough training), 0.05 (finetune training)

After training, we end up with a SOM and a codebook matrix. The codebook matrix is composed of codebook vectors of size 1x4 for the neurons 1 to 100. Each row of the codebook matrix is the codebook vector of a different neuron. A neuron is said to be hit when it is chosen as the BMU for an input vector. In Scenario 3, not only the hit values but also the frequencies of the hits are taken into consideration while studying the congestion behaviour. In Fig.3.66, these values are shown inside the neurons, most of the neurons carry the following information:

$$hit_1(f_1)$$
$$hit_2(f_2)$$
$$\vdots$$
$$hit_n(f_n)$$

which means that the neuron is hit for n different labels (hit_1, hit_2, ..., hit_n). The frequency values in paranthesis, $f_1, f_2, ..., f_n$, show how many times the neuron is hit for that label. A weighted averaging technique is used to represent each neuron by a single label value:

$$\frac{(hit_1).f_1 + (hit_2).f_2 + \ldots + (hit_n).f_n}{f_1 + f_2 + \ldots + f_n} \tag{3.20}$$

The result of the above equation is rounded up to obtain the average label value of each neuron. The resulting label values are seen in Fig.3.67. In Fig.3.66, there are neurons with no hit values. The average hit values for these unhit neurons are calculated by using the average hit values of their primary neighbors. As seen in Fig.3.67, neurons with the same average hit values are next to one another and they build regions.

Scenario 2:

As in Scenario 1, there is an information flow between the routers and the observation unit, however the queue management scheme (RED) does not change due to network congestion level. RED parameters (max_{th}, min_{th}, $max_{p_{den}}$) are kept constant whatever reply is received. The values of these parameters are as follows:

- $min_{th} = 100$ packets
- $max_{th} = 200$ packets
- $max_{p_{den}} = \frac{1}{max_p} = 10$
- exponential weight factor: ewf = 9

Scenario 3:

The resulting SOM structure of Scenario 1 is used in simulation of Scenario 3. At specific times, the vector of mean, variance, skewness, kurtosis values is applied as an input to the SOM obtained before. This vector is compared with each SOM codebook vector and the euclidean distances are calculated. The neuron whose codebook vector is the most similar to the input vector - with the least euclidean distance - is the BMU for this input.

For estimating future congestion status, we pay attention to the congestion regions (R1, R2, R3) - shown in Fig.3.67 and described in step 6 - which differ from one another by the level of congestion they represent:

- Neurons with average label values 0 or 1 are grouped to form region R1. R1 represents the network status which has no global congestion problem. If BMU is in R1, RED parameters should be changed for preventing unnecessary early drops; the amount of change is determined by the previous BMU region.

- Neurons with average label values greater than 3 are grouped to form region R3. R3 represents the status where the congestion problem must be seriously considered. If BMU is in R3, earlier drops are necessary for decreasing the number of queued packets and the level of congestion. The amount of change in RED parameters is determined by the previous BMU region.

- R2 appears between regions R1 and R3, not only by its location but also by the congestion status it represents.

Scenario 3 provides the estimation of RED parameters, as summarized in Steps 7 - 8. The queue management scheme used in this scenario is a RED variant whose parameters are given in Table 3.11. As seen in the table, the values of min_{th} and $max_{p_{den}}$ are kept constant while max_{th} values are changed with respect to the congestion alarm number. Equation (3.21) and Fig.3.68 show how max_{th} changes with respect to the congestion alarm number. max_{th} keeps its new value at least 1 second before another update takes place.

$$max_{th} = 295 - 15 * (7 - \text{congestion alarm number}) \tag{3.21}$$

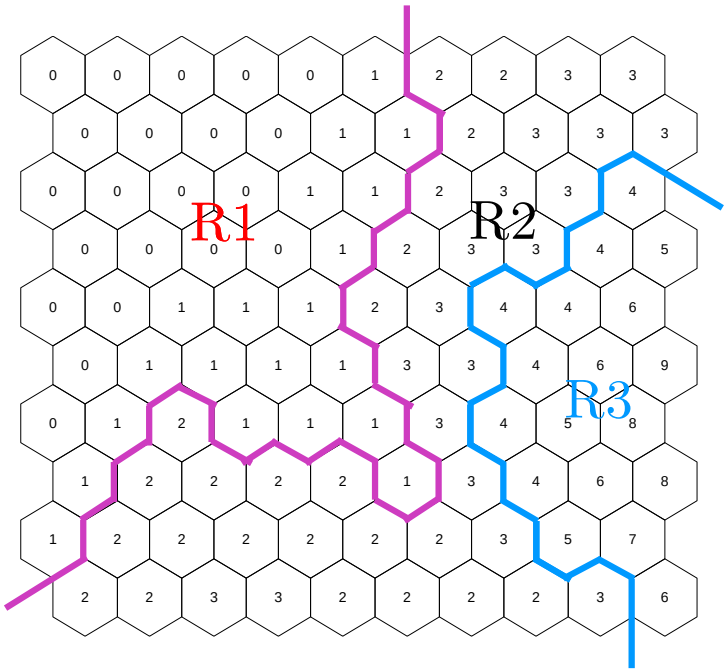

Figure 3.67: Average hit values of the neurons and the congestion regions

Table 3.11: BMU regions and RED parameters for Scenario 3

Region of previous BMU	Region of current BMU	Congestion alarm number	min_{th}	max_{th}	$max_{p_{den}}$
R1	R1	7	100	295	10
R1	R2	4	100	250	10
R1	R3	3	100	235	10
R2	R1	6	100	280	10
R2	R2	4	100	250	10
R2	R3	2	100	220	10
R3	R1	5	100	265	10
R3	R2	4	100	250	10
R3	R3	1	100	205	10

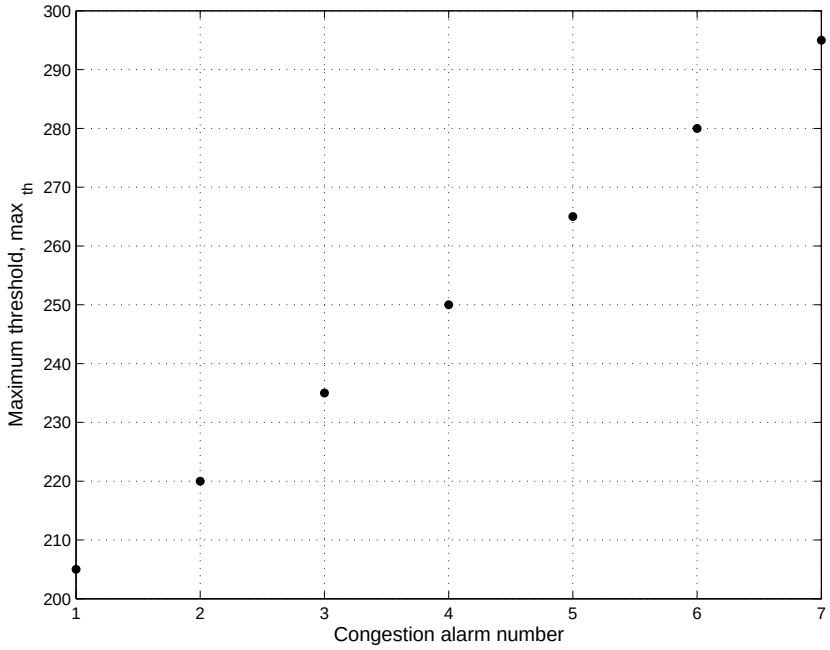

Figure 3.68: max_{th} vs congestion alarm number for Scenario 3

Scenarios 1 - 3 are simulated for 320 sec. Four different simulations are performed for each scenario. After running four individual simulations, the average of the results are obtained and comparative graphs are plotted. Fig.3.69 shows how the curve of average values is obtained for a global statistic: traffic receive rate of ftp traffic. Fig.3.69 has 5 curves: the curves of data1, data2, data3, data4 are obtained by different simulations. Curve of average values (plotted in black) is obtained by calculating the mean value for each time interval.

Simulation results are given in Fig.3.70 - Fig.3.79. Except Fig.3.70 and Fig.3.71, all the plots are generated by finding the moving averages within a window of size 5. The result for each statistic is given as a group of 3 graphs, each for a different scenario. Different colors are used for graphs: graph for Scenario 1 is plotted in black, graph for Scenario 2 is plotted in blue and graph for Scenario 3 is plotted in red. Each graph is composed of average values of simulation results obtained for four different simulations. Fig.3.70 and Fig.3.71 are the graphics for two global statistics: ftp traffic received and ftp traffic sent, respectively. The aim of plotting these figures is determining

the throughput from the ratio of received to sent. It is seen that throughput has similar values for all scenarios.

When all the graphs are investigated, the following results are obtained:

- In the interval [0 100] sec., which corresponds to the time where the global congestion status is low, DT (Scenario 1) provides minimum delay and delay variation.
- In the interval [300 315] sec., which corresponds to the time where the global congestion status is high, RED (Scenario 2) provides minimum delay and delay variation.
- Delay and delay variation values obtained for Approach 2 is
  - smaller than the values for RED, when the global congestion status is low,
  - smaller than the values for DT, when the global congestion status is high.
- From the perspective of throughput, Approach 2 performs as well as RED and DT.

As a result, Approach 2 was successful in taking advantages of RED and DT: delays and delay variations are kept at small values while throughputs are as high as those of RED and DT.

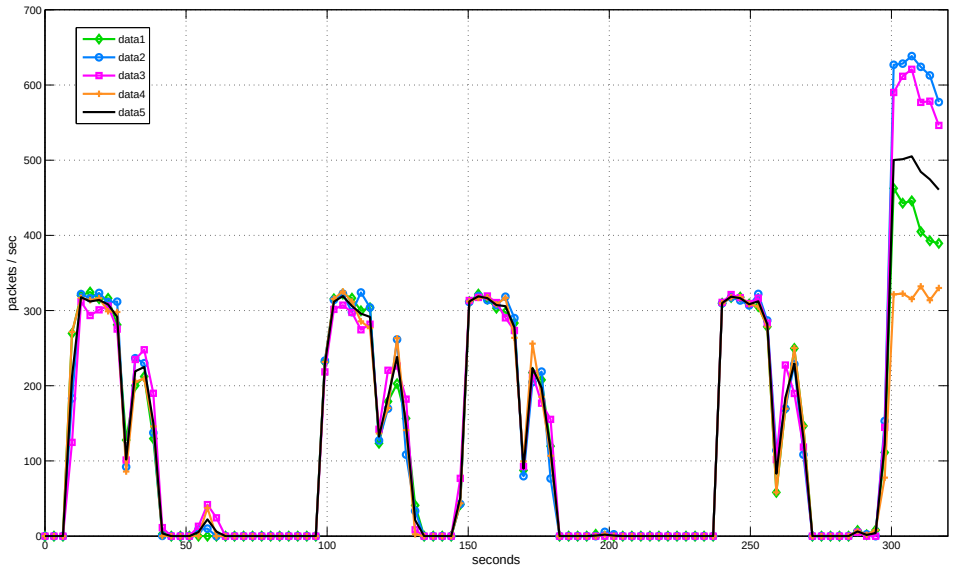

Figure 3.69: ftp traffic receive rate curves for four different simulations and the curve of average values (results for scenario1)

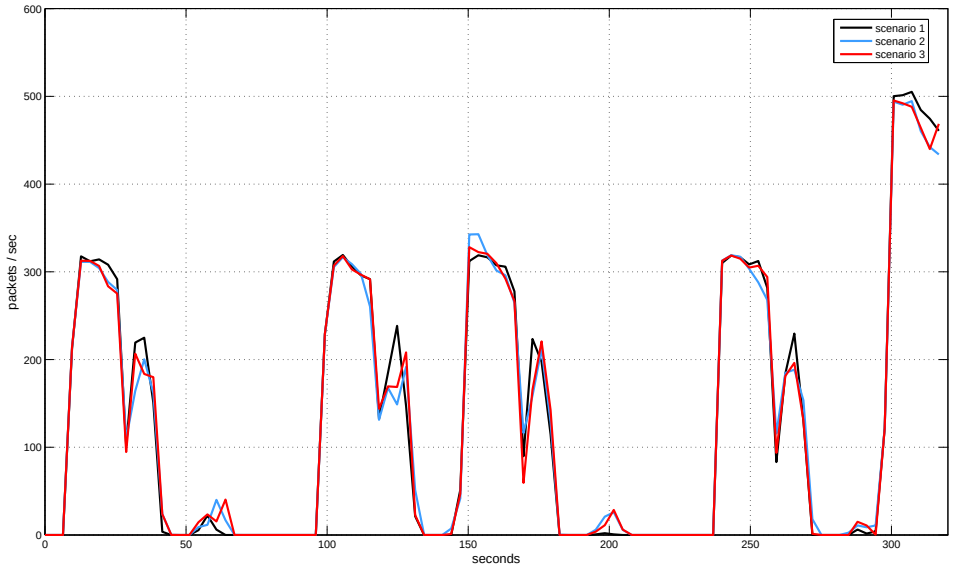

Figure 3.70: Global statistic: ftp traffic receive rate

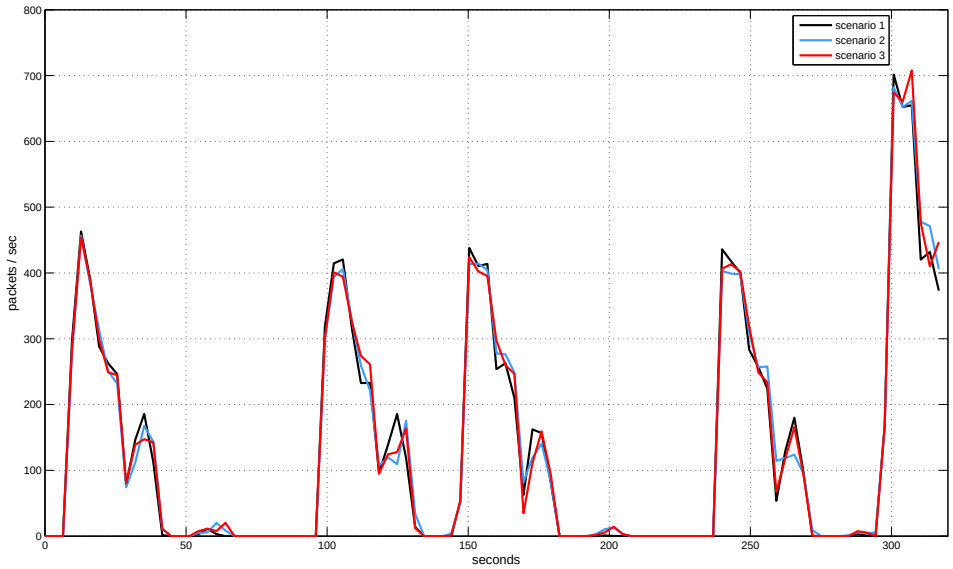

Figure 3.71: Global statistic: ftp traffic sending rate

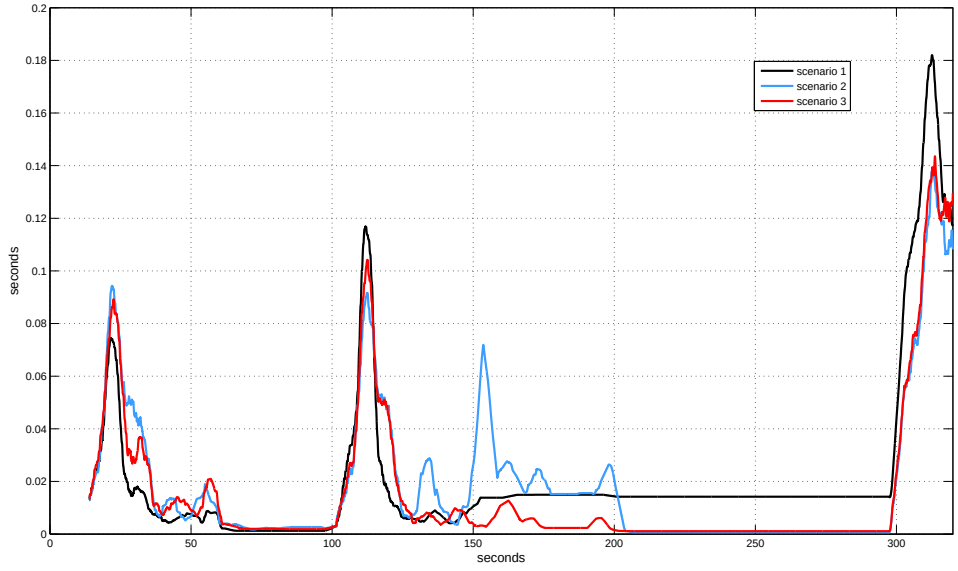

Figure 3.72: Moving average of end-to-end delay measured between node_0 and node_6

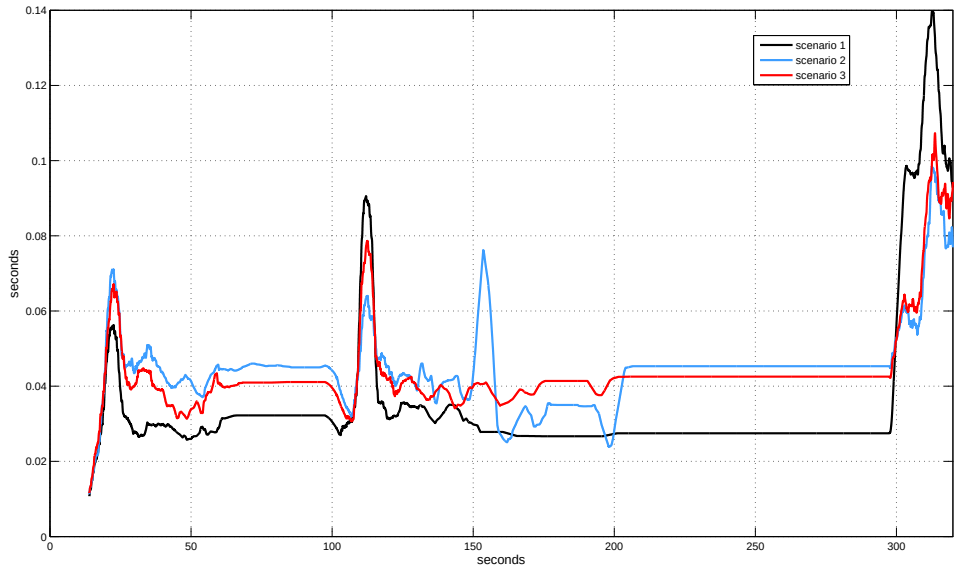

Figure 3.73: Moving average of end-to-end delay variation measured between node_0 and node_6

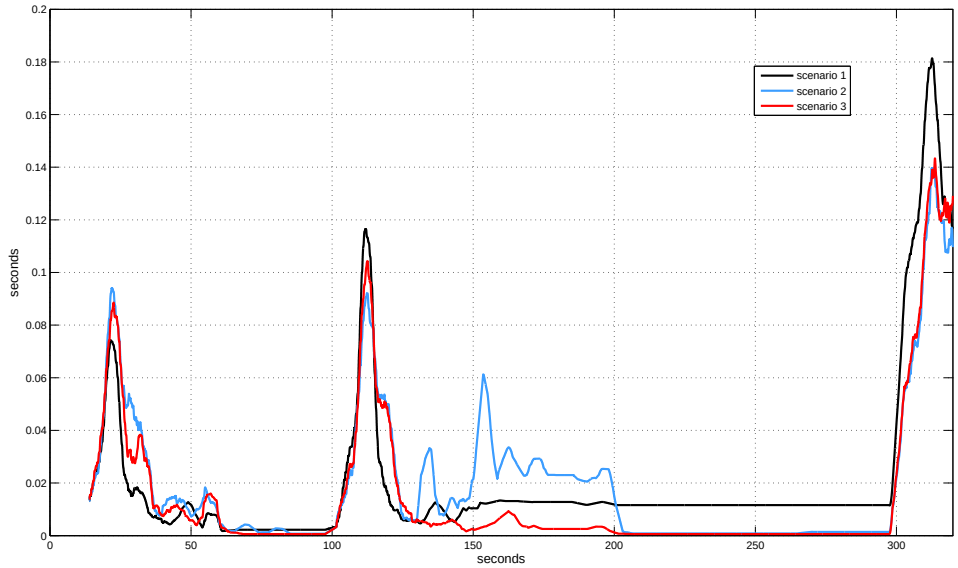

Figure 3.74: Moving average of end-to-end delay measured between node_1 and node_7

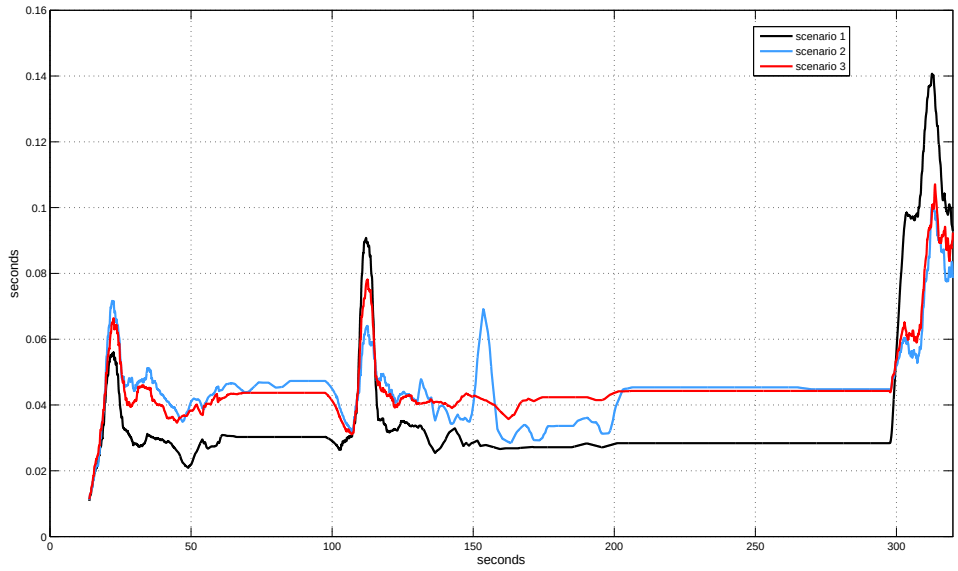

Figure 3.75: Moving average of end-to-end delay variation measured between node_1 and node_7

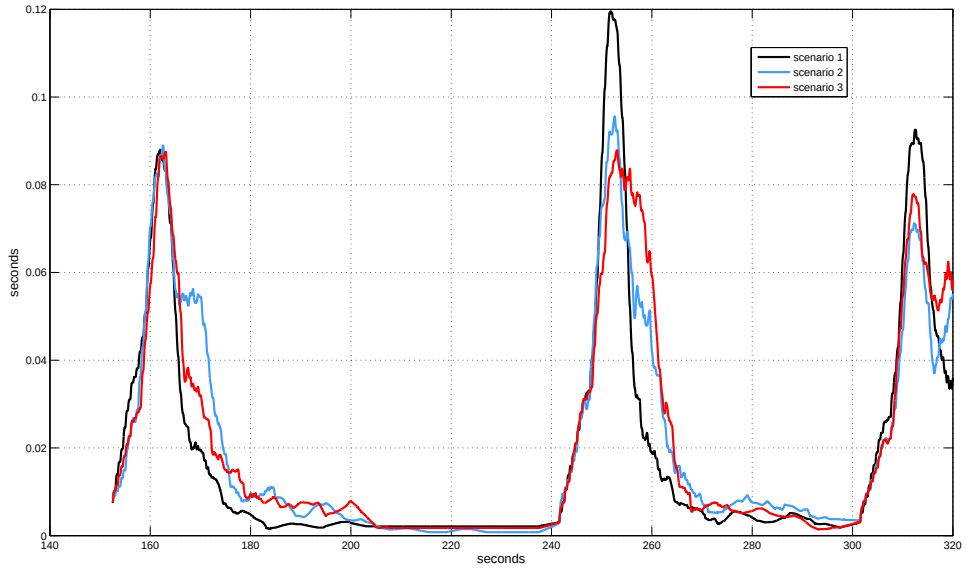

Figure 3.76: Moving average of end-to-end delay measured between node_2 and node_4

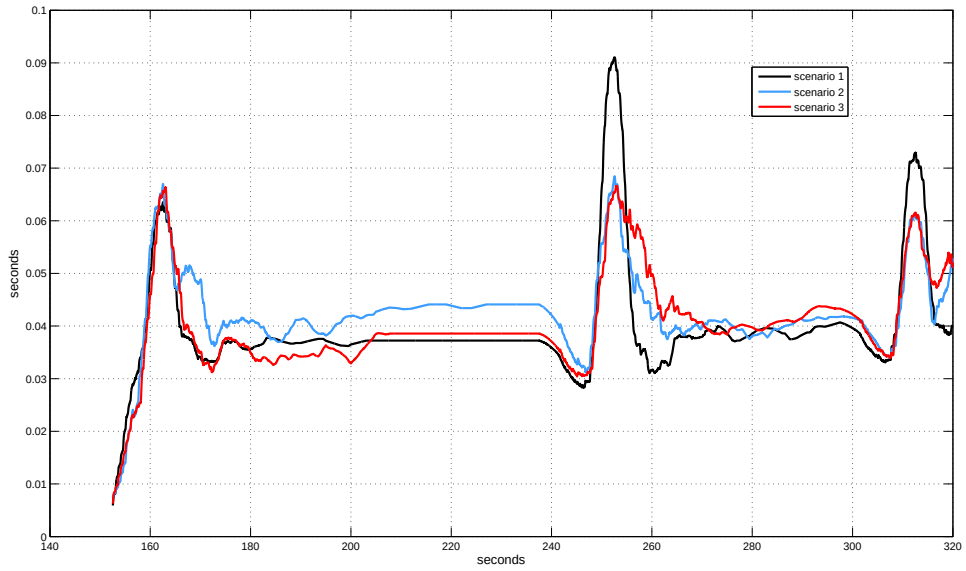

Figure 3.77: Moving average of end-to-end delay variation measured between node_2 and node_4

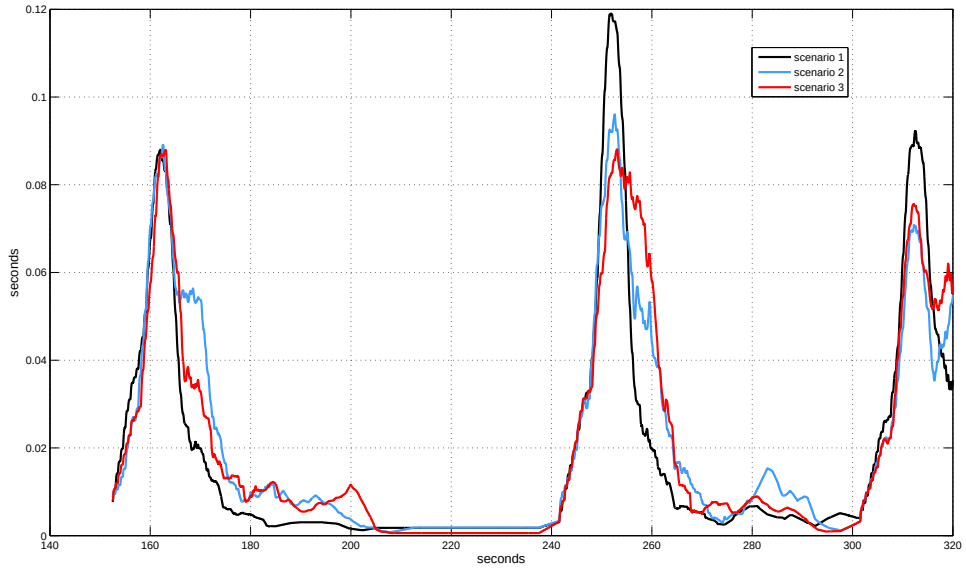

Figure 3.78: Moving average of end-to-end delay measured between node_3 and node_5

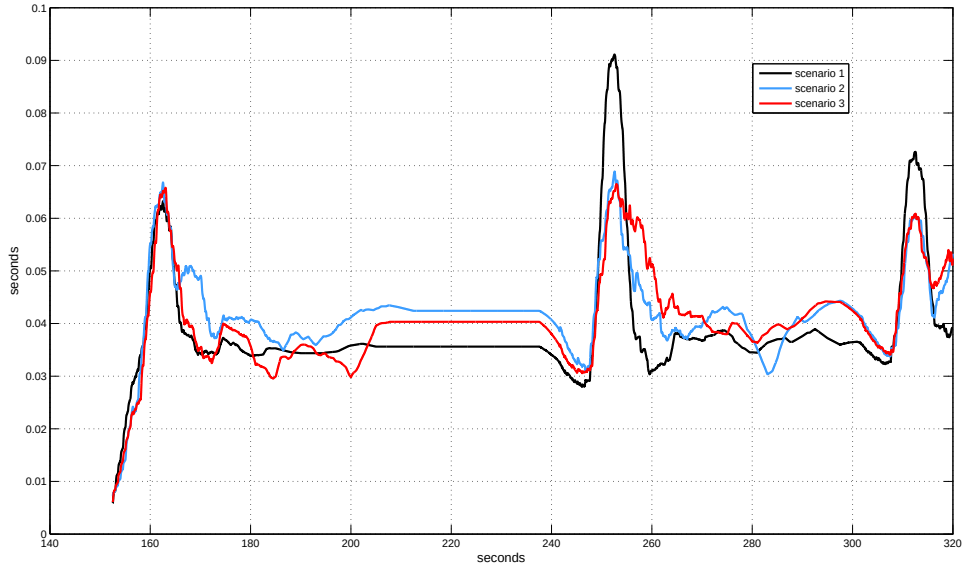

Figure 3.79: Moving average of end-to-end delay variation measured between node_3 and node_5

4 CONCLUSION

There are many published studies on avoiding congestion in dumbbell networks. Most of these studies are inspired by the idea of providing a trade-off between throughput and delay/delay variation in single-bottleneck networks. The study about RED [16] is one of the most famous studies and it has a proved ability in management of single bottleneck queues. When the performance of RED is tested in a multi-bottleneck network, as a part of this dissertation, some disadvantages are noticed. When the congestion level of the network is low, meaning there are no sustained packet drops in most of the interfaces, early drop mechanism and static threshold values will be handicaps for RED. In that case, its performance against delays/delay variations is not as good as that of DT. In order to improve its performance in multi-bottleneck networks, some changes have to be made. As a result of this idea, different approaches are proposed in this dissertation. The achievements of this dissertation may be summarized as follows:

- **A centralized observation method is presented to monitor global congestion behaviour of an IP network.**

  A centralized observation unit which collects information from all router interfaces, is presented. In addition to this, IP routers are specialized to inform the observation unit about the changes in their queue lengths.

- **The relation between local and global congestion behaviours are studied by the help of a SOM.**

  Queue length values of router interfaces, that are managed by RED (in Approach 1) or DT (in Approach 2), are collected by the observation unit. Offline training is performed by using the collected data and SOM is produced. After training, SOM structure is embedded in the centralized observation unit. The queue length information received by the observation unit is accepted as an input to the SOM. Global conges-

tion level is monitored and tendency of congestion is estimated by using the SOM.

- **The information provided by the trained SOM is used to develop new queue management approaches.**

  As a part of this dissertation, two new AQM approaches are proposed. With new approaches, different from the original RED scheme, values of some RED parameters are dynamically changed due to the observed level and tendency of global congestion, that are obtained by using SOMs. In Approach 1, updated parameters are min_{th} and max_p while in Approach 2, max_{th} is updated.

- **OPNET Modeler simulations are generated to study performance of new approaches.**

  OPNET Modeler is used not only to make innovations in IP routers and produce a host with specialized functionality, but also to generate an IP network and make simulations on it. New RED variants, that are introduced with respect to new approaches, update their RED parameters due to changes in global congestion level. Congestion avoidance is performed in IP routers and the RED variant of the new approach is used as the queue management scheme in router interfaces. By this way, RED scheme in individual routers are made adaptive to changes in the congestion throughout the network. The simulation results for Approach 1 show that changing the value of max_p is more effective in improving RED performance than changing min_{th}, due to variations in global congestion conditions. An improvement is observed in end-to-end time delays and throughput, however RED is still better than RED variant of Approach 1 in handling delay variations. Approach 2 is based on the idea of using advantages of AQM and PQM. Its performance is tested and compared with DT and RED performances through various simulations. Results show that Approach 2 is successful in eliminating disadvantages of RED and DT when analyzed from the perspective

of delay/delay variation. Moreover, Approach 2 performs this without causing a decrease in throughput.

The approaches proposed in this work worth attention for providing congestion avoidance in multi-bottleneck IP networks, because of the above achievements. However, assignment of RED parameters is still among favorite topics of queue management and it is open to improvements. New approaches could be produced in order to consider global and local congestion problems at the same time and develop new queue management strategies.

BIBLIOGRAPHY

[1] V. G. Cerf. On the evolution of Internet technologies. *Proceedings of the IEEE*, **92**(9):1360–1370, 2004.

[2] J. Postel. *RFC 793: Transmission Control Protocol*. Internet Engineering Task Force, 1981.

[3] J. F. Kurose and K. W. Ross. *Computer Networking: A Top Down Approach Featuring the Internet*. Addison-Wesley, U.S.A., 2003.

[4] R. Braden, D. Clark, and S. Shenker. *RFC 1633: Integrated services in the Internet architecture: an overview*. Internet Engineering Task Force, 1994.

[5] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. *RFC 2475: An Architecture for Differentiated Services*. Internet Engineering Task Force, 1998.

[6] D. Grossman. *RFC 3260: New terminology and clarifications for DiffServ*. Internet Engineering Task Force, 2002.

[7] A. Meddeb. Internet QoS: Pieces of the puzzle. *IEEE Communications Magazine*, **48**(1):86–94, 2010.

[8] R. Guerin and V. Peris. Quality-of-service in packet networks: basic mechanisms and directions. *Computer Networks*, **31**:169–189, 1999.

[9] H.-L. Lu and I. Faynberg. An architectural framework for support of quality of service in packet networks. *IEEE Communications Magazine*, **41**(6):98–105, 2003.

[10] S. Ryu, C. Rump, and C. Qiao. Advances in Internet congestion control. *IEEE Communications Surveys*, **5**(1):28–39, 2003.

[11] M. Hassan and R. Jain. *High Performance TCP/IP Networking: Concepts, Issues, and Solutions*. Pearson Prentice Hall, U.S.A., 2004.

[12] V. Jacobson. Congestion Avoidance and Control. *Proceedings of the ACM SIGCOMM*, 314-329, August 1988.

[13] W. Stevens. *RFC 2001: TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms*. Internet Engineering Task Force, 1997.

[14] 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, and L. Zhang. *RFC 2309: Recommendations on queue management and congestion avoidance in the Internet*. Internet Engineering Task Force, 1998.

[15] J. Aweya, M. Ouellette, and D.Y. Montuno. A control theoretic approach to active queue management. *Computer Networks*, **36**:203–235, 2001.

[16] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, **1**(4):397–413, 1993.

[17] C. V. Hollot, V. Misra, D. Towsley, and W.-B. Gong. A Control Theoretic Analysis of RED. *Proceedings of INFOCOM 2001*, **3**:1510–1519, 2001.

[18] W.-C. Feng, D. D. Kandlur, D. Saha, and K. G. Shin. A self-configuring RED gateway. *Proceedings of the IEEE INFOCOM 1999*, **3**:1320–1328, 1999.

[19] Y.-D. Xu, Z.-Y. Wang, and H. Wang. ARED: A novel adaptive cogestion controller. *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*, **2**:708–714, 2005.

[20] J.-S. Li and Y.-S. Su. Random early detection with flow number estimation and queue length feedback control. *Journal of System Architecture*, **52**:359–372, 2006.

[21] J. Ming, C. Qin, and T. Jingfan. TL-RED: A traffic load adaptive RED algorithm. *Proceedings of IET International Conference on Wireless, Mobile and Multimedia Networks*, 1-3, 2006.

[22] H.-J. Ho and W.-M. Lin. AURED - autonomous random early detection for TCP congestion control. *Proceedings of International Conference on Systems and Networks Communications*, 79-84, 2008.

[23] J. Aweya, M. Ouellette, D. Y. Montuno, and A. Chapman. A load adaptive mechanism for buffer management. *Computer Networks*, **36**:709–728, 2001.

[24] B. Zheng and M. Atiquzzaman. A framework to determine the optimal weight parameter of RED in next-generation Internet routers. *International Journal of Communication Systems*, **21**:987–1008, 2008.

[25] T. A. Trinh and S. Molnar. A comprehensive performance analysis of random early detection mechanism. *Telecommunication Systems*, **25**:9–31, 2004.

[26] D. Que, Z. Chen, and B. Chen. An improvement algorithm based on RED and its performance analysis. *Proceedings of International Conference on Signal Processing*, 2005-2008, 2008.

[27] B. Zheng and M. Atiquzzaman. DSRED: An active queue management scheme for next generation networks. *Proceedings of the IEEE Conference on Local Computer Networks*, 242-251, 2000.

[28] B. Zheng and M. Atiquzzaman. DSRED: Improving performance of active queue management over heterogeneous networks. *Proceedings of the IEEE International Conference on Communications*, **8**:2375–2379, 2001.

[29] T. J. Ott, T. V. Lakshman, and L. H. Wong. SRED: Stabilized Red. *Proceedings of the IEEE INFOCOM 99*, **3**:1346–1355, 1999.

[30] J. Aweya, M. Ouellette, and D. Y. Montuno. DRED: a random early detection algorithm for TCP/IP networks. *International Journal of Communication Systems*, **15**:287–307, 2002.

[31] W.-C. Feng, K. G. Shin, D. D. Kandlur, and D. Saha. The Blue active queue management algorithms. *IEEE/ACM Transactions on Networking*, **10**(4):513–528, 2002.

[32] G.-Y. Su and C. C. Ho. Random Early Detection improved by progressive adjustment method. *Proceedings of the IEEE* 6^{th} *National Conference on telecommunication technologies*, 250-253, 2008.

[33] J. Hong, C. Joo, and S. Bahk. Active queue management algorithm considering queue and load states. *Computer Communications*, **30**:886–892, 2007.

[34] A. Jain, A. Karandikar, and R. Verma. An adaptive prediction based approach for congestion estimation in active queue management (APACE). *Proceedings of IEEE Global Telecommunications Conference*, **7**:4153–4157, 2003.

[35] S. Athuraliya, S. H. Low, V. H. Li, and Q. Yin. REM: Active queue management. *IEEE Network Magazine*, **15**:48–53, 2001.

[36] S. S. Kunniyur and R. Srikant. An adaptive virtual queue (AVQ) algorithm for active queue management. *IEEE/ACM Transactions on Networking*, **12**:286–299, 2004.

[37] A. Chydzinski. Towards a stable AQM via dropping function shaping. *Proceedings of 9th International Conference on Networks*, 93-97, 2010.

[38] C. Zhu, O. W. W. Yang, J. Aweya, M. Ouellette, and D. Y. Montuno. A comparison of active queue management algorithms using the OPNET Modeler. *IEEE Communications Magazine*, **40**(6):158–167, 2002.

[39] C. V. Hollot, V. Misra, and W.-B. Gong D. Towsley. Analysis and design of controllers for AQM routers supporting TCP flows. *IEEE Transactions on Automatic control*, **47**:945–959, 2002.

[40] C. V. Hollot, V. Misra, and W.-B. Gong D. Towsley. On designing improved controllers for AQM routers supporting TCP flows. *IEEE INFOCOM 2001*, **3**:1726–1734, 2001.

[41] J. Aweya, M. Ouellette, D. Y. Montuno, and K. Felske. Design of rate-based controllers for active queue management in TCP/IP networks. *Computer Communications*, **31**:3344–3359, 2008.

[42] S. H. Low, F. Paganini, J. Wang, S. Adlakha, and J. C. Doyle. Dynamics of TCP/RED and a scalable control. *Proceedings of the* 21^{th} *Annual Conference of the IEEE Computer and Communications Societies*, **1**:239–248, 2002.

[43] J. Sun, K.-T. Ko, G. Chen, S. Chan, and M. Zukerman. PD-RED: To improve the performance of RED. *IEEE Communications Letters*, **7**:406–408, 2003.

[44] S. M. M. Alavi and M. J. Hayes. Robust active queue management design: a loop-shaping approach. *Computer Communications*, **32**:324–331, 2009.

[45] S. Ryu, B. Ryu, M. Jeong, and S. Park. PI-PD controller for adaptive and robust active queue management for Internet congestion control. *Simulation*, **81**(6):437–459, 2005.

[46] J. Sun and M. Zukerman. RAQ: A robust active queue management scheme based on rate and queue length. *Computer Communications*, **30**:1731–1741, 2007.

[47] C. Wang, B. Li, Y. T. Hou, K. Sohraby, and K. Long. A stable rate-based algorithm for active queue management. *Computer Communications*, **28**:1731–1740, 2005.

[48] P.-F. Quet and H. Özbay. On the design of AQM supporting TCP flows using robust control theory. *IEEE Transactions on Automatic Control*, **49**(6):1031–1036, 2004.

[49] Opnet Technologies Inc. Modeling Concepts Reference Manual Release 12.0.

[50] J. Postel. *RFC 792: Internet Control Message Protocol*. Internet Engineering Task Force, 1981.

[51] T. Kohonen. *Self-Organizing maps*. Springer, Germany, 1997.

[52] T. Kohonen. The Self-Organizing Map. *Proceedings of the IEEE*, **78**(9):1464–1480, 1990.

[53] S. Haykin. *Neural Networks: A comprehensive foundation*. Pearson Education, India, 1999.

[54] M. Masugi and T. Takuma. Multi-fractal analysis of IP network traffic for assessing time variations in scaling properties. *Physica D*, **225**:119–126, 2007.

[55] M. Masugi. QoS mapping of VoIP communication using self-organizing neural network. *IEEE workshop on IP operations and management*, 13-17, 2002.

[56] E. Lochin and B. Talavera. Managing network congestion with a Kohonen-based RED queue. *Proceedings of IEEE International Conference on Communications*, 5586-5590, 2008.

[57] L. Enhai, L. Yan, and P. Ruimin. An improved random early detection algorithm based on flow prediction. *Proceedings of International Conference on Intelligent Networks and Intelligent Systems*, 425-428, 2009.

[58] T. Bonald, M. May, and J.-C. Bolot. Analytic Evaluation of RED performance. *Proceedings of the IEEE INFOCOM*, 1415-1424, 2000.

[59] S. Patel, P. Gupta, and G. Singh. Performance Measure of Drop Tail and RED Algorithm. *Proceedings of International Conference on Electronic Computer Technology*, 35-38, 2010.

[60] C. A. Hawkins and J. E. Weber. *Statistical Analysis: Applications to Business and Economics*. Harper & Row Publishers, New York, 1980.

[61] Ö. Yelbaşı and E. Germen. A New Method for Estimating RED Parameters Based on Global Congestion Notification. *Proceedings of International Conference on Network Computing and Information Security, NCIS 2011 (accepted for publication)*, 2011.