

بسم الله الرحمن الرحيم



**SUDAN UNIVERSITY OF SCIENCE AND
TECHNOLOGY**



**COLLEGE OF GRADUATE STUDIES
ELECTRONICS ENGINEERING DEPARTMENT**

**An Algorithm for Adaptive Queue Management
Based on Time-to-live field (TTL)**

خوارزمية إدارة الصف إعتماًداً على TTL

A thesis submitted in Partial fulfillment of the requirements of the
degree of M.Sc. in Computer Engineering

Prepared By:

Hussam Mohammed Mukhtar

Supervisor:

Dr. Abuagla Babiker Mohammed

July 2015



Initiation

الآية

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِیْمِ

﴿يَرْفَعِ اللّٰهُ الَّذِیْنَ اٰمَنُوْا مِنْكُمْ وَالَّذِیْنَ اٰتَوْا الْعِلْمَ دَرَجٰتٍ﴾

[سورة المجادلة: الآية 11]

Dedication

Dedicated to

My parents, wife and son

And to

My sisters and brothers

Acknowledgement

I would like to thank my university (Sudan University of Science and Technology) and my (College of Graduate Studies Electronics Engineering Department). And my teachers for inspiring me this project.

Especial thanks to my supervisor Dr. Abuagla Babiker Mohammed; whose sponsorship and guidance are considered as a very precious asset to make this effort (An Algorithm for adaptive queue management Based on time-to-live field T.T.L) possible.

My greatest thanks to my wife who followed up this project since the beginning up to the end.

My thanks also extend to engineer Thowban who had personally provided the equipment required in this project. And thanks to engineer Osama who help me in the idea of this project.

Also my thanks to my parents who for their continuous support. At last, thanks to Mohammed Kamal for his help.

Abstract

Each Active Queue Management Algorithms aims to improve the performance of routers and thus improve the performance of the network in general. The goals of these Algorithms are avoiding congestion, reduce delays, and decrease (Bufferload) and preservation of the channel capacity. So all attempts of Active Queue Management Algorithms aimed to drop the packets appropriately and determine the length of the queue properly as much as possible to achieve the desired goal of improving network performance. this project has invented a new algorithm called the (An Algorithm for adaptive queue management based on time-to-live field (TTL) in this algorithm the queue are arranged according to the (TTL) value, from the smallest to the biggest, so that the smaller value served first. This algorithm does not depend on the parameters used in most Active Queue Management Algorithms. But it relies on the packets, which spend the longest distance and the (round-trip time RTT) is barely to be completed, will be served first in order to reduce retransmitting the packet many time. Therefore retransmission of the packets many time lead to congestion. It is found that this algorithm reduces congestion, delays and preserves the channel capacity better than (IP QoS Priority Queuing) Algorithms.

المستخلص

كل خوارزميات ادارة الصف تهدف الى تحسين اداء الموجهات، وبالتالي تحسين الشبكة بشكل عام، اذ أن مهمتها تجنب الازدحام وتقليل التاخير و تقليل تضخم الصفوف (Bufferbload) والمحافظة على سعة القناة. وعليه فإن كل محاولات خوارزميات ادارة الصف تهدف الى سقوط الرسائل بشكل مناسب وتحديد طول صف مناسب قدر المستطاع لتحقيق الهدف المنشود وهو تحسين أداء الشبكة. في هذا المشروع اقترعت خوارزمية جديدة سميت ب(خوارزمية ادارة الصف اعتمادا على time-to-live field TTL) في هذه الخوارزمية يتم ترتيب الصف وفق قيمة (TTL) من الأصغر الى الأكبر، بحيث القيمة الأصغر تخدم اولاً. هذه الخوارزمية لم تعتمد على المعاملات المستخدمة في أغلب خوارزميات ادارة الصف، بل اعتمدت على ان الرسالة التي قطعت أطول مسافة ويكاد زمن وصول الرسالة و عودة الأقرار بها (RTT) ان ينتهي، تخدم اولاً. وذلك لتقليل إعادة إرسال الرسالة مرة اخرى، وعليه فإن إعادة الارسال اكثر من مرة يزيد حجم الازدحام. ووجد ان هذه الخوارزمية تقلل الازدحام و التاخير و تحافظ على سعة القناة اكثر من خوارزمية (IP QoS Priority Queuing).

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
	Initiation	I
	Dedication	II
	Acknowledgement	III
	Abstract	IV
	Abstract in Arabic	V
	Table of contents	VI
	List of table	VII
	List of figures	IX
	List of abbreviations	X
1	Introduction	1
	1.1 Preface	1
	1.2 Problem statement	2
	1.3 Proposed solution	2
	1.4 Objectives	2
	1.5 Scope	2
	1.6 Methodology	3
	1.7 Research Outlines	3

2	Literature Review	5
	2.1 Background	5
	2.1.1 Understand Congestion	5
	2.1.2 Techniques to Avoid Congestion	6
	2.1.3 TCP Congestion Control	6
	2.1.4 Understanding Queues	8
	2.2 Related Work	9
	2.2.1 Effective queue management approaches leads to congestion reduction	9
	2.2.2 Queue management: the effect of Queue length (Buffer size) on congestion reduction	9
	2.2.3 Queue management: the effect of packet dropping in congestion reduction	11
	2.2.4 Controlled Delay Management (CoDel)	13
	2.2.5 TCP Congestion Control	16
	2.2.6 Queue management: the effect of TTL In congestion reduction	18
3	Methodology	23
	3.1 (priority Queuing based on TTL value) (PQT) algorithm	23
	3.1.1 Subroutine of PQT algorithm	24
	3.1.2 Time-To-Live Scheduler	25
	3.1.3 Flow Chart	26

	3.2 Network Simulation of PQT	28
	3.2.1 Overview	28
	3.2.2 Network Simulation block diagram	28
	3.2.3 The hardware of network	29
	3.2.4 The software of network	32
	3.2.5 Network Inventory Summary	36
4	Results and discussion	38
	4.1 Results of priority Queuing based on TTL value algorithm	38
	4.1.2 Evaluation of PQT through various traffic loads	40
	4.2 Router A <-> Router B: point-to-point Statistics	41
	4.2.1 Queuing delay in router A <-> router B (sec)	41
	4.2.2 Throughput (-- >) (packet / sec)	45
	4.3 Router A Statistics	47
	4.3.1 IP Processing Delay (sec)	47
	4.3.2 CPU Utilization in Router A	48
	4.3.3 Switch A <-> Router A – in point-to-point queuing delay (sec) -->	50
5	Conclusion and Recommendations	54
	5.1 Conclusion	54
	5.2 Recommendations	55
	REFERENCES	56
	Appendix A	59 – 60

List of Tables

TABLE NO.	TITLE	PAGE
3.1	Network Inventory Summary	36

List of Figures

FIGURE NO.	TITLE	PAGE
2.1	Four hosts connected by two switches	5
3.1	Subroutine of PQT algorithm	24
3.2	TTL scheduler	25
3.3	Flow Chart	27
3.4	Block diagram of network	28
3.5	Connected routers in logical network	30
3.6	Links used in network	31
3.7	Application configuration	33
3.8	QoS configuration	34
4.1	Simulation: Network infrastructure Design, Configuration and PQT implementation	39
4.2	Queuing delay from router A to router B	41
4.3	The zooming Queuing delay from router A to router B	42
4.4	Queuing delay from router B to router A	43

4.5	The zooming Queuing delay from router B to router A	44
4.6	The average Throughput (-- >) (packet / sec)	45
4.7	The zooming average in throughput (packets/sec)	46
4.8	The average in IP Processing Delay (sec)	47
4.9	CPU Utilization of Router A (overlaid)	48
4.10	CPU Utilization of Router A (stacked)	49
4.11	Switch A <-> Router A—in point-to-point-queuing delay (sec) (-->)	50
4.12	Switch A <-> Router A—in point-to-point-queuing delay (sec) (<--)	51
4.13	Switch A <-> Router A in point-to-point queuing delay (sec) (<--) (-- >)	52

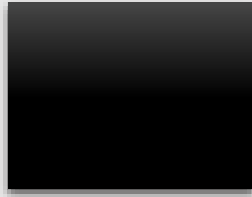
LIST OF SYMBOLS

- β - A constant timeout value weighting factor.
- α - A constant new weighted average factor.

LIST OF ABBREVIATIONS

ACK	:	Acknowledgement
AQM	:	Active Queue Management
ARED	:	Adaptive or Active Random Early Detection
BPDU	:	Bridge Protocol Data Units
CoDel	:	Controlled Delay Management
CPU	:	Central Process Unit
DRED	:	Dynamic Random Early Detection
FIFO	:	First In First Out
Gbps	:	Giga bit per second
HTTP	:	Hypertext Transfer Protocol
IP	:	Internet Protocol
LAN	:	Local Area Network
MTU	:	Maximum Transmission Unit
OSPF	:	Open Shortest Path First
PQT	:	Priority Queuing based on TTL value
QoS	:	Quality of Service
RED	:	Random Early Detection
RIP	:	Routing Information Protocol
RTT	:	Round Trip Time
Sec	:	Second
SRED	:	Stabilized Random Early Drop

TCP : Transmission Control Protocol
TOS : Type of Service
TTL : Time To Live
UDP : User Datagram Protocol
WFQ : weighted Fair Queuing



1

Chapter

Introduction

Introduction

1.1 Preface:

A number of active queue management algorithms for IP (Internet Protocol) router, have been proposed in the past few years ^[8]. The essential goal of most of them is to avoid congestion, reducing delay and keeping the link utilization high. To achieve the above mentioned objectives they play around several parameters, such as queue size, average queue size, average queue length and dropping probability.

Although a lot of work has been done with respect to the queue management (queue management algorithm), the time to live (TTL) value has an indication to the number of hops that the packet has taken.

Thus, in practice, the time to live acts as a (hop limit) rather than an estimate of delay. Each router only decrements the value by 1. One of the most important and complex ideas in TCP (Transmission Control Protocol) is embedded in the way it handles timeout and retransmission. Retransmission is resending of packets which have been either damaged or lost. Retransmission is one of the basic mechanisms used by TCP protocols. To handle packet loss, transport protocols use positive acknowledgement with retransmission.

If congestion occurs, the ratio of packet dropping will be increased accordingly, the retransmission will be more resulting in lower throughput. Moreover, regarding the dropping and overall bandwidth consumption, dropping the packet with less TTL may lead to bandwidth waste (since those packet with less TTL value have travelled very long than those with large

value of TTL). Thus there is an urgent need to consider the TTL value when doing queuing management.

1.2 Problem statement:

Inefficient queue management has a direct effect on retransmission due to packet dropping. When congestion occurs in a router, no way to skip packet dropping, but the question to be raised is to Which packet that must be dropped, and what are the suitable criteria for dropping a specific packet which can lead to retransmission reduction.

1.3 Proposed solution:

Propose a new algorithm for retransmission reduction using TTL. In another way it means to re-order the router queue using time-to-live (TTL) as one of the metrics for prioritizing the packet dropping. The algorithm should be able to avoid congestion and reduce the delay caused by retransmission and acknowledgments.

1.4 Objectives:

The main objectives of this research are to:

- ❖ Propose and simulate new queue management algorithm using TTL as one of the essential metrics.
- ❖ Comparing the proposed algorithm with the related work
- ❖ Avoiding the congestion and reducing delay

The outcome of the above mentioned objectives is to reduce the bufferload, packet dropping and Preserves the channel capacity

1.5 Scope:

The scope of this research is to cover the area of backbone router, queue core router, retransmission, and internet network area should be covered.

1.6 Methodology:

The research had been done in multi-stages to complete the design of (priority Queuing based on TTL value) PQT algorithm and run it.

Stage one: design the controlling retransmission using TTL. (PQT algorithm) by flowing parameters:

Determine the size queue according to the literature review

Stage two: re-order the packets in router queue using time-to-live (TTL).

Stage three: implement and test of the PQT algorithm in simulation system.

Stage four: comparing the proposed algorithm with standard queuing algorithms via the Op-net simulation and plot the comparison results.

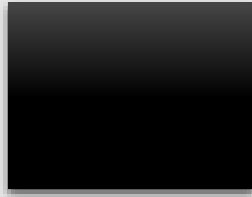
1.7 Research Outlines:

Chapter2: Explores the literature review which consists of two major parts theoretical background which includes oversizing router buffers, packet loss, packet drop, transport protocols, time-to-live (TTL) positive acknowledgement with retransmission and timeout... etc. more over the second part elaborates the relevant related work.

Chapter3: Shows the network design including Algorithm (priority Queuing based on TTL value) (PQT) and explain in details all components and their parameters and how does it work.

Chapter4: Explain the implementation; furthermore, it also includes the simulation in addition to the results and discussion. Finally, it also compares the results of the proposed algorithm with other scenario of similar work.

Chapter5: Comprises the conclusion and the recommendations for the future researchers.



2

Chapter

Literature Review

Literature Review

2.1 Background:

This chapter discusses concept of understand congestion, techniques to avoid Congestion, transmission control protocol (TCP) congestion control, understanding queues, queuing delay and related Work.

2.1.1 Understand Congestion:

To understand how easily congestion can occur, consider four hosts connected by two switches as Figure (2-1) illustrates.



Figure (2-1) Four hosts connected by two switches.

Assume each connection in the figure operates at 1 Giga bit per seconds (Gbps), and consider what happens if both computers attached to switch 1 attempt to send data to a computer attached to switch 2. Switch 1 receives data at an aggregate rate of 2 Gbps, but can only forward 1 Gbps to switch 2. The situation is known as congestion. Even if a switch temporarily stores packets in memory, congestion results in increased delay. If congestion persists, the switch will run out of memory and begin discarding packets. Although retransmission can be used to recover lost packets, retransmission sends more packets into the network.

Thus, if the situation persists, an entire network can become unusable; the condition is known as congestion collapse. In the Internet, congestion usually occurs in routers. Transport protocols attempt to avoid congestion collapse by monitoring the network and reacting quickly once congestion starts.

2.1.2 Techniques to Avoid Congestion:

There are two basic approaches:

- 1- Arrange for intermediate systems (i.e., routers) to inform a sender when congestion occurs
- 2- Use increased delay or packet loss as an estimate of congestion

The former scheme is implemented either by having routers send a special message to the source of packets when congestion occurs or by having routers set a bit in the header of each packet that experiences delay caused by congestion. When the second approach is used, the computer that receives the packet includes information in the acknowledgement (ACK) to inform the original sender. Using delay and loss to estimate congestion is reasonable in the Internet because:

Modern network hardware works well; most delay and loss results from congestion, not hardware failure. The appropriate response to congestion consists of reducing the rate at which packets are being transmitted. Sliding window protocols can achieve the effect of reducing the rate by temporarily reducing the window size.

2.1.3 TCP Congestion Control:

One of the most interesting aspects of TCP is a mechanism for congestion control. Recall that in the Internet, delay or packet loss is more likely to be caused by congestion than a hardware failure, and that retransmission can

exacerbate the problem of congestion by injecting additional copies of a packet. To avoid congestion collapse, TCP uses changes in delay as a measure of congestion, and responds to congestion by reducing the rate at which it retransmits data. Although we think of reducing the rate of transmission, TCP does not compute a data rate. Instead, TCP bases transmission on buffer size. That is, the receiver advertises a window size, and the sender can transmit data to fill the receiver's window before an ACK is received. To control the data rate, TCP imposes a restriction on the window size by temporarily reducing the window size, the sending TCP effectively reduces the data rate. The important concept is:

Conceptually, a transport protocol should reduce the rate of transmission when congestion occurs. Because it uses a variable-size window, TCP can achieve a reduction in data rate by temporarily reducing the window size. In the extreme case where loss occurs, TCP temporarily reduces the window to one-half of its current value. TCP uses a special congestion control mechanism when starting a new connection or when a message is lost. Instead of transmitting enough data to fill the receiver's buffer (i.e., the receiver's window size), TCP begins by sending a single message containing data. If an acknowledgement arrives without additional loss, TCP doubles the amount of data being sent and sends two additional messages. If both acknowledgements arrive, TCP sends four messages, and so on. The exponential increase continues until TCP is sending half of the receiver's advertised window. When one-half of the original window size is reached, TCP slows the rate of increase, and increases the window size linearly as long as congestion does not occur. The approach is known as slow start. TCP's congestion control mechanisms respond well to increases in traffic.

By backing off quickly, TCP is able to alleviate congestion. In essence, TCP avoids adding retransmissions when the Internet becomes congested. More important, if all TCPs follow the standard, the congestion control scheme means that all senders back off when congestion occurs and congestion collapse is avoided. ^[6]

2.1.4 Understanding Queues:

Developing effective active queue management has been hampered by misconceptions about the cause and meaning of queues. Network buffers exist to absorb the packet bursts that occur naturally in statistically multiplexed networks. Queues occur in the buffers as a result of short-term mismatches in traffic arrival and departure rates that arise from upstream resource contention, transport conversation startup transients, and/or changes in the number of conversations sharing a link. Unfortunately, other network behavior can cause buffers to fill, with effects that are not nearly as benign. With the wrong conceptual model for queues, Active Queue Managements (AQM)s have limited operational range, require a lot of configuration tweaking, and frequently impair rather than improve performance. ^[2]

Queuing Delay: the store-and-forward paradigm used in packet switching means that a device such as a router collects the bits of a packet, places them in memory, chooses a next hop, and then waits until the packet can be sent before beginning transmission. Such delays are known as queuing delays. In the simplest case, a packet is placed in a first in first out output (FIFO) queue, and the packet only needs to wait until packets that arrived earlier are sent; more complex systems implement a selection algorithm that gives priority to some packets. Queuing delays are variable the size of a queue depends entirely on the amount of traffic that has arrived recently.

Queuing delays account for most delays in the Internet. When queuing delays become large, we say that the network is congested.^[6]

2.2 Related Work:

2.2.1 Effective queue management approaches leads to congestion reduction:

The requirement to management the queue by increasing or adaptive the length of the queue on routers is very important to reduce the congestion occur in routers there for reduce the congestion for all network.

2.2.2 Queue management: the effect of Queue length (Buffer size) on congestion reduction:

All Internet routers contain buffers to hold packets during times of congestion. Today, the size of the buffers is determined by the dynamics of TCP's congestion control algorithm. In particular, the goal is to make sure that when a link is congested, it is busy 100% of the time; which is equivalent to making sure its buffer never goes empty. That is mean determined the size of buffer is very important today, because it has more effect to reduced congestion.

C. Villamizar and C. Song Report the ((Round-Trip Time) $RTT \times C$) rule equation ----- (1), in which the authors' measure link utilization of a 40 Mb/s network with 1, 4 and 8 long-lived TCP flows for different buffer sizes. They found that for FIFO dropping discipline and very large maximum advertised TCP congestion windows it is necessary to have buffers of $(RTT \times C)$ to guarantee full link utilization. They concluded that because of dynamics of TCP's congestion control algorithms a router needs an amount of buffering equal to the average round trip time of a flow that passes

through the router, multiplied by the capacity of the router's network interfaces.

This is the well-known rule-of-thumb ($B = RTT \times C$) rule. Where B is length of the queue, RTT is the average round-trip time of a flow passing across the link and return Ack, and C is the data rate of the link (channel capacity).^[1]

The rule-of-thumb ($B = RTT \times C$) is lacks efficiency because small number of TCP flows and the length of queue directly proportional with channel capacity that causes long queues and increase the delay of queues.

G. Appenzeller et al prove that the rule-of-thumb ($B = RTT \times C$) is now outdated and incorrect for backbone routers today. This is because of the large number of flows (TCP connections) multiplexed together on a single backbone link. Then the rule-of-thumb is correct only if the number of flow is few. They believe that significantly smaller buffers could be used in backbone routers (e.g. by removing 99% of the buffers) without a loss in network utilization.

The goal of their paper is to determine the size of the buffer so as to maximize throughput of a bottleneck link. The basic idea is that when a router has packets buffered, its outgoing link is always busy. If the outgoing link is a bottleneck, then we want to keep it busy as much of the time as possible, and so we just need to make sure the buffer never under flows and goes empty. They showed that a link with (n) flows requires no more than $B = (RTT \times C) / \sqrt{n}$ ----- equation (2), for long-lived or short-lived TCP flows. Here (n) is the number of flows at link.^[2]

The rule of sizing router buffers $B = (RTT \times C) / \sqrt{n}$ is convincing for my objectives. In this research because they prove that we can use smaller buffer for large number of flows with full link utilization. Subsequently this will

cause a little delay for any packet. In this research the same equation is used but, the calculated (n) is being differently tacked.

O. A. Bashir et al developed a new algorithm called the automatic calculation of the length of the queues (auto), the main work for this algorithm is to modify the length of the queue B automatically, according to the equation $B = (RTT \times C) / \sqrt{n}$.

Where that C is the channel capacity and RTT is the time to send a packet and return Ack, and n is the number of connections probably estimated and not retain any information relating to connections.

The idea of new algorithm is to estimate the length of the accurate queue and estimates the number of connections in the router probably. Estimates the number of connection probably taken from SRED (Stabilized Random Early Drop) algorithm. [3]

The auto algorithm used the $B = (RTT \times C) / \sqrt{n}$ and depend on parameters as follow: C= fixed, RRT=100ms (from codel: Controlled Delay Management) and n= number of connections probably estimated. In my research I take the idea of (n) and apply it in another way. They found that the performance of the auto algorithm is better with respect to congestion that causes Buffer bloat and reduced the delay in routers in addition to the preservation of the channel capacity.

2.2.3 Queue management: the effect of packet dropping in congestion reduction:

Dropping is one of the common methods using to reduce congestion, and there are many manners and techniques can perform the dropping. And these some of them.

T. J. Ott et al described a mechanism called “SRED” (Stabilized Random Early Drop). Like RED (Random Early Detection) SRED pre-emptively discards packets with a load-dependent probability when a buffer in a router in the Internet or an Intranet seems congested.

SRED has an additional feature that over a wide range of load levels helps it stabilize its buffer occupation at a level independent of the number of active connections.

The main idea is to compare, whenever a packet arrives at some buffer, the arriving packet with a randomly chosen packet that recently preceded it into the buffer. When the two packets are “of the same flow” we declare a “hit”. The sequence of hits is used in two ways, and with two different objectives in mind:

- To estimate the number of active flows
- To find candidates for “misbehaving flow”

A simple way of comparing an arriving packet with a recent other packet is to compare it with a packet still in the buffer. This makes it impossible to compare packets more than one buffer drain time apart. To give the system longer memory, we augment the information in the buffer with a “Zombie List”. We can think of this as a list of M recently seen flows, with the following extra information for each flow in the list: a “Count” and a “time stamp”. Note that this zombie list or flow cache is small and maintaining this list is not the same as maintaining per-flow state. We call the flows in the zombie list “zombies”.

The zombie list starts out empty. As packets arrive, as long as the list is not full, for every arriving packet the packet flow identifier (source address,

destination address, etc.) is added to the list, the Count of that zombie is set to zero, and its timestamp is set to the arrival time of the packet. Once the zombie list is full it works as follows: Whenever a packet arrives, it is compared with a randomly chosen zombie in the zombie list.

(1: Hit) If the arriving packet's flow matches the zombie we declare a "hit". In that case, the Count of the zombie is increased by one, and the timestamp is reset to the arrival time of the packet in the buffer. (2: No Hit) If the two are not of the same flow, we declare a (no hit). In that case, with probability p the flow identifier of the packet is overwritten over the zombie chosen for comparison. The Count of the zombie is set to 0, and the timestamp is set to the arrival time at the buffer. With probability $1-p$ there is no change to the zombie list. ^[4]

The idea of SRED algorithm is create zombie list, the arriving packet with a recent other packet is zombie list to compare it with a packet still in the buffer. Once the zombie list is full it works as follows: Whenever a Packet arrives, it is compared with a randomly chosen zombie in the zombie list. In the research produced different list from TTL.

2.2.4 Controlled Delay Management (CoDel):

K. Nichols and V. Jacobson innovate a new algorithm called CoDel (Controlled Delay Management) has three major innovations that distinguish it from prior AQMs. First, CoDel's algorithm is not based on queue size, queue-size averages, queue-size thresholds, rate measurements, link utilization, and drop rate or queue occupancy time. They used the local minimum queue as a more accurate and robust measure of standing queue. Then we observed that it is sufficient to keep a single-state variable of how long the minimum has been above or below the target value for standing

queue delay rather than keeping a window of values to compute the minimum. Finally, rather than measuring queue size in bytes or packets, they used the packet-sojourn time through the queue. Use of the actual delay experienced by each packet is independent of link rate, gives superior performance to use of buffer size, and is directly related to the user-visible performance. Using the minimum value has some important implications. The minimum packet sojourn can be decreased only when a packet is dequeued, which means all the work of CoDel can take place when packets are dequeued for transmission and that no locks are needed in the implementation. The minimum is the only statistic with this property. The only addition to packet arrival is that a timestamp of packet arrival time is created.

If the buffer is full when a packet arrives, then the packet can be dropped as usual. CoDel assumes a standing queue of target is acceptable and that it is unacceptable to drop packets when there are less than one MTU's (maximum transmission unit's) worth of bytes in the buffer. CoDel identifies the persistent delay by tracking the (local) minimum queue delay packets experience. To ensure the minimum value does not become stale, it has to have been experienced within the most recent interval. When the queue delay has exceeded target for at least interval, a packet is dropped and a control law sets the next drop time. The next drop time is decreased in inverse proportion to the square root of the number of drops since the dropping state was entered, using the well-known relationship of drop rate to throughput to get a linear change in throughput.

When the queue delay goes below target, the controller stops dropping. No drops are carried out if the buffer contains fewer than an MTU's worth of

bytes. Additional logic prevents reentering the dropping state too soon after exiting it and resumes the dropping state at a recent control level, if one exists. Target and interval are constants with straightforward interpretations: acceptable standing queue delay and a time on the order of a worst-case RTT of connections through the bottleneck. We experimented to determine values for target and interval that give a consistently high utilization with a controlled delay across a range of bandwidths, RTTs, and traffic loads. Below a target of 5ms, utilization suffers for some conditions and traffic loads; above 5ms there is very little or no improvement in utilization. Interval is loosely related to RTT since it is chosen to give endpoints time to react without being so long that response times suffer.

A setting of 100ms works well across a range of RTTs from 10ms to 1 second (excellent performance is achieved in the range from 10ms to 300ms). CoDel's efficient implementation and lack of configuration are unique features that make it suitable for managing modern packet buffers. The three innovations using minimum rather than average as the queue measure, simplified single-state variable tracking of minimum, and use of queue-sojourn time lead directly to these unique features^[5].

The CoDel algorithm is new Innovations in controlling queue delay has major difference from active queue management (AQM) (e.g.: RED, SRED, ARED, DRED, BLUE, FRED) are congestion control algorithm. Codel is delay control algorithm is depend on the equation of size queue ($B = \text{nominal RRT} \times C$) ----- equation (3). Where nominal RRT = 100ms. Used in the research. Codel has constants parameters: Target (Target queue delay) = 5 ms Interval (Sliding minimum time window width) (RRT) = 100 ms Max packet (Maximum packet size in bytes) (MTU) = 512

2.2.5 TCP Congestion Control:

D. E. Comer explains that to handle packet loss, transport protocols use positive acknowledgement with retransmission. Whenever a frame arrives intact, the receiving protocol software sends a small acknowledgement (ACK) message that reports successful reception. The sender takes responsibility for ensuring that each packet is transferred successfully. Whenever it sends a packet, the sending-side protocol software starts a timer. If an acknowledgement arrives before the timer expires, the software cancels the timer; if the timer expires before an acknowledgement arrives, and the software sends another copy of the packet and starts the timer again. The action of sending a second copy is known as retransmitting, and the copy is commonly called a retransmission. Of course, retransmission cannot succeed if a hardware failure has permanently disconnected the network or if the receiving computer has crashed. Therefore, protocols that retransmit messages usually bound the maximum number of retransmissions. When the bound has been reached, the protocol stops retransmitting and declares that communication is impossible.

Note that if packets are delayed, retransmission can introduce duplicate packets. Thus, transport protocols that incorporate retransmission are usually designed to handle the problem of duplicate packets.

As expected TCP uses retransmission to compensate for packet loss. Because TCP provides data flow in both directions; both sides of a communication participate in retransmission. When TCP receives data, it sends an acknowledgement back to the sender. Whenever it sends data, TCP starts a timer, and retransmits the data if the timer expires. Before TCP was invented, transport protocols used a fixed value for retransmission. Delay the

protocol designer or network manager chose a value that was large for the expected delay. Designers working on TCP realized that a fixed timeout would not operate well for the Internet. Thus, they chose to make TCP's retransmission adaptive.

That is, TCP monitors current delay on each connection, and adapts (i.e., changes) the retransmission timer to accommodate changing conditions.

How can TCP monitor Internet delays? In fact, TCP cannot know the exact delays for all parts of the Internet at all times. Instead, TCP estimates round-trip delay for each active connection by measuring the time needed to receive a response. Whenever it sends a message to which it expects a response, TCP records the time at which the message was sent. When a response arrives, TCP subtracts the time the message was sent from the current time to produce a new estimate of the round-trip delay for that connection. As it sends data packets and receives acknowledgements, TCP generates a sequence of round-trip estimates and uses a statistical function to produce a weighted average. In addition to a weighted average, TCP keeps an estimate of the variance, and uses a linear combination of the estimated mean and variance when computing the timeout which retransmission is needed.

Experience has shown that TCP adaptive retransmission works well. Using the variance helps TCP react quickly when delay increases following a burst of packets. Using a weighted average helps TCP reset the retransmission timer if the delay returns to a lower value after a temporary burst. When the delay remains constant, TCP adjusts the retransmission timeout to a value that is slightly longer than the mean round-trip delay. When delays start to

vary, TCP adjusts the retransmission timeout to a value greater than the mean to accommodate peaks. ^[6]

2.2.6 Queue management: the effect of TTL in congestion reduction:

Time to live: An 8-bit integer initialized by the original sender and decremented by each router that processes the datagram.

If the value reaches zero, the datagram's discarded and an error message is sent back to the source. ^[6]

D. E. Comer clarify that in principle, field time to live specifies how long, in seconds, the datagram is allowed to remain in the internet system. The idea is both simple and important: whenever a computer injects a datagram into the internet, it sets a maximum time that the datagram should survive. Routers and hosts that process datagram's must decrement the time to live field as time passes and remove the datagram from the internet when its time expires. Estimating exact times is difficult because routers do not usually know the transit time for physical networks.

A few rules simplify processing and make it easy to handle datagram without synchronized clocks. First, each router along the path from source to destination is required to decrement the time to live field by one when it processes the datagram header. Furthermore, to handle cases of overloaded routers that introduce long delays, each router records the local time when the datagram arrives, and decrements the time to live by the number of seconds the datagram remained inside the router waiting for service.

Whenever a time to live field reaches zero, the router discards the datagram And sends an error message back to the source. The idea of keeping a timer for datagram's is interesting because it guarantees that datagram cannot travel around an internet forever, even if routing tables become corrupt and

routers route datagram's in a circle. Although once important, the notion of a router delaying a datagram for many seconds is now outdated - current routers and networks are designed to forward each datagram within a reasonable time. If the delay becomes excessive, the router simply discards the datagram.

Thus, in practice, the time to live acts as a "hop limit" rather than an estimate of delay. Each router only decrements the value by 1. One of the most important and complex ideas in TCP is embedded in the way it handles timeout and retransmission. Like other reliable protocols, TCP expects the destination to send acknowledgements whenever it successfully receives new octets from the data stream. Every time it sends a segment, TCP starts a timer and waits for an acknowledgement. If the timer expires before data in the segment has been acknowledged, TCP assumes that the segment was lost or corrupted and retransmits it.

To understand why the TCP retransmission algorithm differs from the algorithm used in many network protocols, we need to remember that TCP is intended for use in an internet environment. In an internet, a segment traveling between a pair of machines may traverse a single, low-delay network (e.g., a high-speed LAN: Local Area Network), or it may travel across multiple intermediate networks through multiple routers. Thus, it is impossible to know a priori how quickly acknowledgements will return to the source. Furthermore, the delay at each router depends on traffic, so the total time required for a segment to travel to the destination and an acknowledgement to return to the source varies dramatically from one instant to another. TCP software must accommodate both the vast differences in the time required to reach various destinations and the changes

in time required to reach a given destination as traffic load varies. TCP accommodates varying internet delays by using an adaptive retransmission algorithm. In essence, TCP monitors the performance of each connection and deduces reasonable values for timeouts. As the performance of a connection changes, TCP revises its timeout value (i.e., it adapts to the change).

To collect the data needed for an adaptive algorithm, TCP records the time at which each segment is sent and the time at which an acknowledgement arrives for the data in that segment. From the two times, TCP computes an elapsed time known as a sample round trip time or round trip sample.

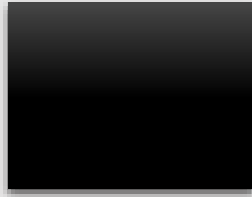
Whenever it obtains a new round trip sample, TCP adjusts its notion of the average round trip time for the connection. Usually, TCP software stores the estimated round trip time, RZT, as a weighted average and uses new round trip samples to change the average slowly. For example, when computing a new weighted average, one early averaging technique used a constant weighting factor, α , where $0 \leq \alpha < 1$, to weight the old average against the latest round trip sample:

$$RTT = (\alpha \times \text{Old RTT}) + ((1 - \alpha) \times \text{New Round Trip Sample}) \text{ --- equation (4)}$$

Choosing a value for α close to 1 makes the weighted average immune to changes that last a short time (e.g., a single segment that encounters long delay). Choosing a value for α close to 0 makes the weighted average respond to changes in delay very quickly. When it sends a packet, TCP computes a timeout value as a function of the current round trip estimate. Early implementations of TCP used a constant weighting factor, β ($\beta > 1$), and made the timeout greater than the current round trip estimate: $\text{Timeout} = \beta * \text{RTT}$ ---- equation (5) Choosing a value for β can be difficult.

On one hand, to detect packet loss quickly, the timeout value should be close to the current round trip time (i.e., β should be close to 1). Detecting packet loss quickly improves throughput because TCP will not wait an unnecessarily long time before retransmitting. On the other hand, if $\beta = 1$, TCP is overly eager - any small delay will cause an unnecessary retransmission, which wastes network bandwidth.

The original specification recommended setting $\beta=2$; more recent work described below has produced better techniques for adjusting timeout. ^[7]



3

Chapter

Methodology

Methodology

3.1 (priority Queuing based on TTL value) (PQT) algorithm:

The idea behind this algorithm is re-ordering the queue of router based on time-to-live (TTL) value from lowest to biggest. Moreover the TTL can also be used to determine the dropped packets when the buffer is full.

The important question is why do we choose TTL as an important metric for prioritizing the packet scheduling and dropping?

The answer to this question can be summarized according to the following reasons:

The First reason: the current TTL value for packet illustrated the time spent in network. Because the current TTL value for packet show the number of hops that packet cross.

The Second reason: the lowest TTL value of packet demonstrated that the packet cross long distance. The cause of that is the packet cross many router in network, and finally, **The Third reason:** There is another reason but indirect to use TTL value in this algorithm is the round trip time (RTT). Before finish the RTT the packet must service.

RTT is determined by the sender and it is 200ms by default for the packet, there for if the TTL value is lowest that is mean the RTT is close to finish. If the RTT finished then retransmission is done again to the packet and that cause congestion in queue of the router and increase the delay of the queue.

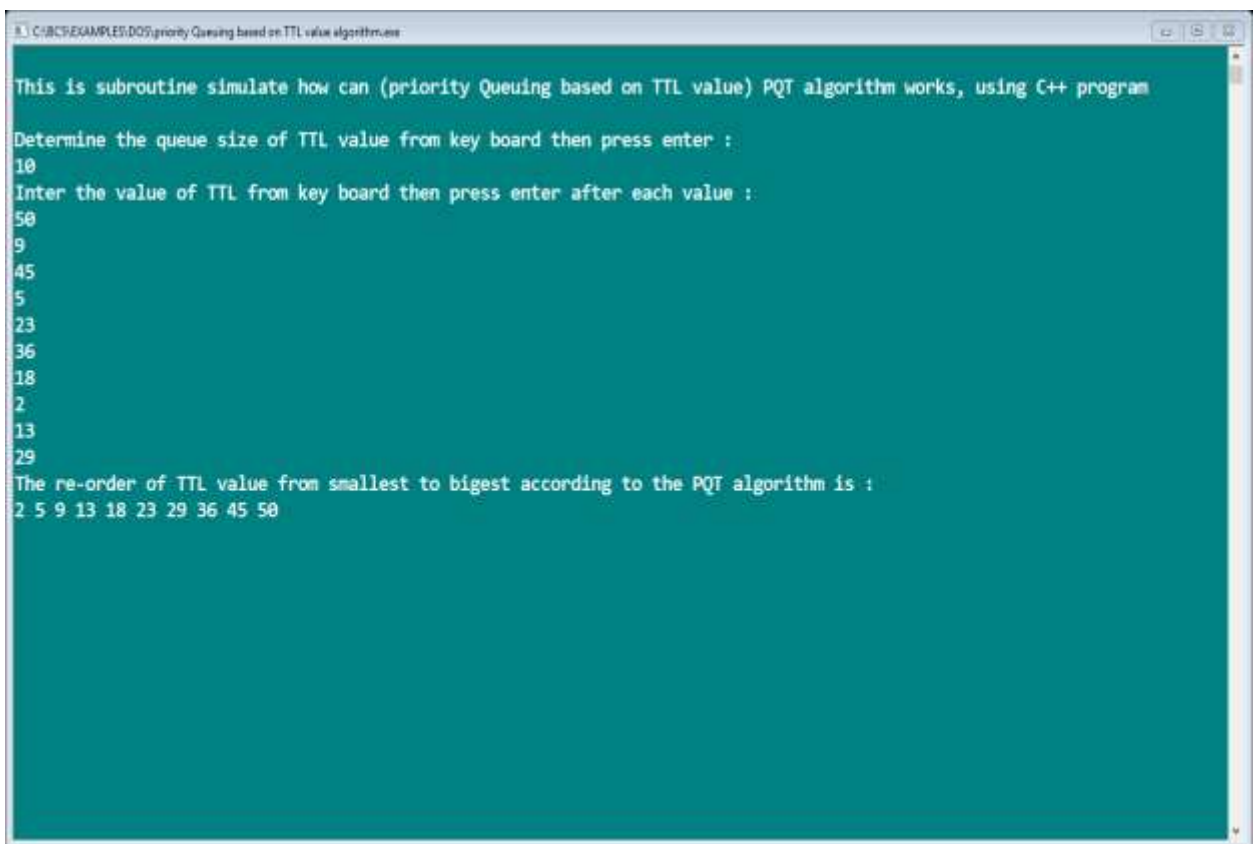
For dropping: if congestion occur and the queue be full, the (PQT) algorithm drop the max TTL value from the queue.

For all these reasons above the TTL value chosen as essential metric for PQT algorithm, there for the packet with lowest value service first from the router.

And one of the method used to avoid the congestion, reducing delay, reducing retransmission is using (PQT) algorithm.

3.1.1 Subroutine of PQT algorithm:

Figure (3-1) below illustrate subroutine simulate how can (priority Queuing based on TTL value) PQT algorithm works, programed by C++ Language. The code of this program illustrated in appendix A.



```
C:\BCF\EXAMPLES\DOS\priority Queuing based on TTL value algorithm.exe

This is subroutine simulate how can (priority Queuing based on TTL value) PQT algorithm works, using C++ program

Determine the queue size of TTL value from key board then press enter :
10
Enter the value of TTL from key board then press enter after each value :
50
9
45
5
23
36
18
2
13
29
The re-order of TTL value from smallest to biggest according to the PQT algorithm is :
2 5 9 13 18 23 29 36 45 50
```

Figure (3-1): subroutine of PQT algorithm

As can be seen from the figure above there is ten different TTL value entered from the keyboard, then the program re-ordered the TTL value from smallest to biggest according to the PQT algorithm.

3.1.2 Time-To-Live Scheduler:

There are many flows came to router, in TTL scheduler the flows have several packets, and every packet has time-to-live value.

The flows enter as input queue to TTL scheduler, the TTL scheduler re-order the packets according to the TTL value from smallest to bigger in output queue. And if the queue is full, packets with maximum TTL values will have greater dropping probability rather than those with lower TTL values (since they are travelling longer distance than the others and has been surviving with more intermediate systems (routers)).

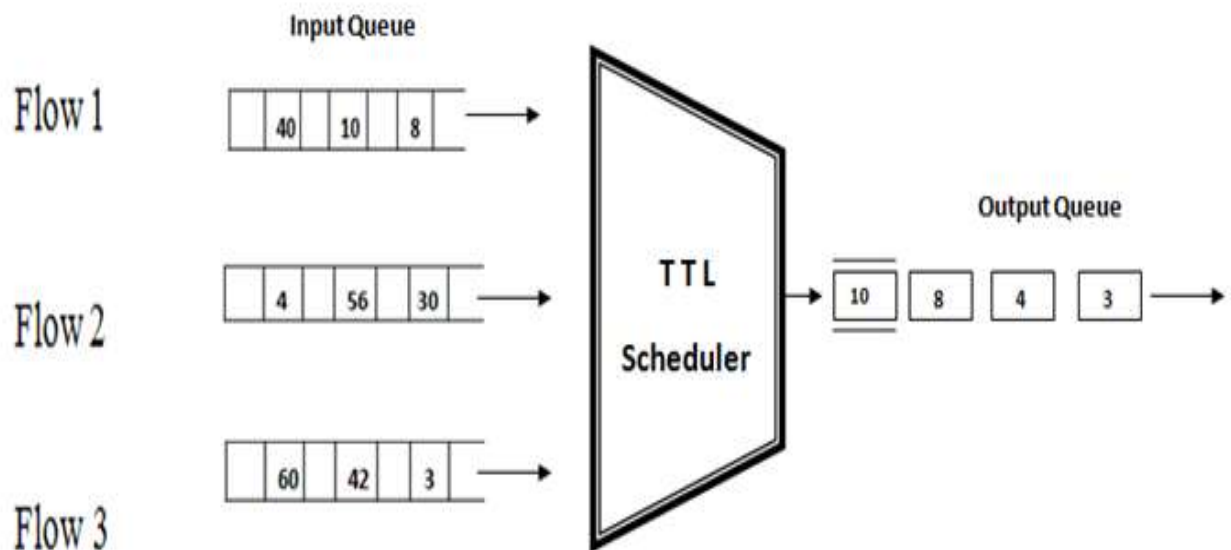


Figure (3-2): TTL scheduler

3.1.3 Flow Chart:

Figure (3-2) shows the flow chart of the proposed algorithm it consists of several steps such as:

First step: the packets received from many flows to the queue of the router with different time-to-live (TTL) value, some packets send from far sender and other send from near sender.

Second step: read and check the time-to-live (TTL) value from IP header by the proposed algorithm (priority Queuing based on TTL value algorithm) (PQT).

Third step: the PQT algorithm re-order the packets according to the smallest time-to-live value.

Case one: if the queue full, then drop the packet with max TTL value.

Case tow: if the queue isn't full, then insert the packet in the queue while considering the TTL value in queue ordering.

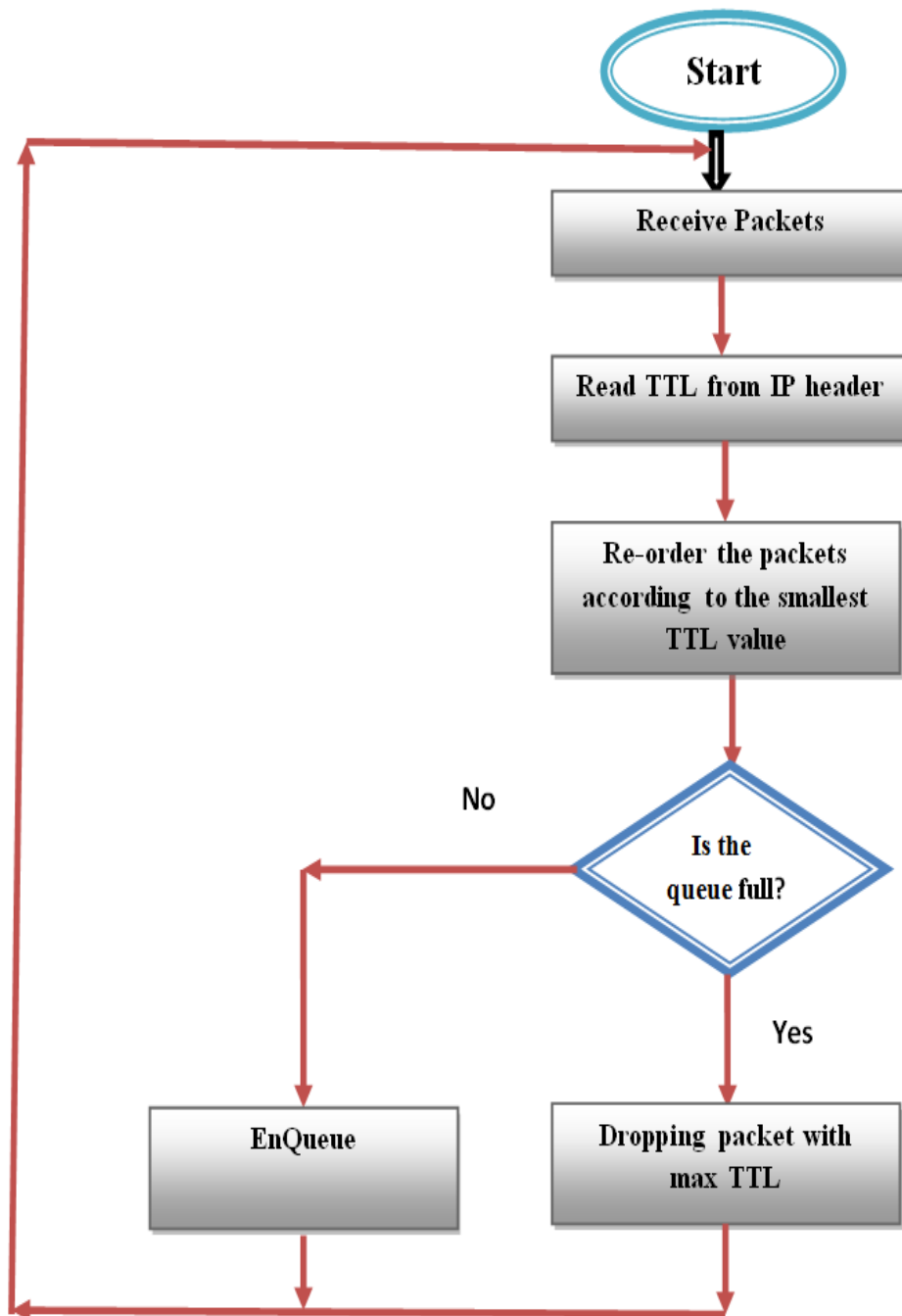


Figure (3-3): Flow Chart

3.2 Network Simulation of PQT:

This part consists of the following: the overview, network simulation block diagram, the hardware of network, the software of network and the network inventory Summary.

3.2.1 Overview:

This section covers and explains the design of all Network Simulation, including Algorithm (priority Queuing based on TTL value) (PQT) in details and their components and parameters and how does it work, using opnet simulation.

3.2.2 Network Simulation block diagram:

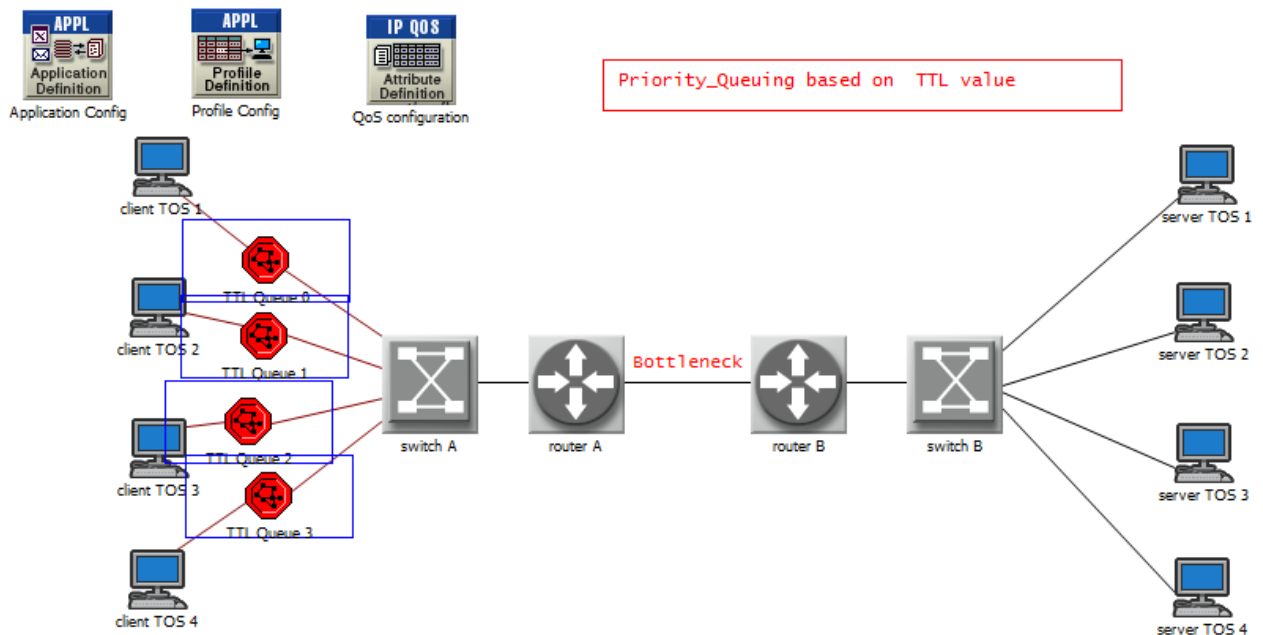


Figure (3-4): block diagram of network

The network consists of two parts: hardware and software.

The 4 clients connected with 4 logical network, every logical network have multi routers, ranged from (10 to 40 routers so as to obtain different TTL values for the purpose of simulation). Every logical network connected with switch A. And switch A connected with router A.

Router A connected with router B. router B connected with switch B. switch B connected with 4 servers. Moreover the performance analysis of the algorithm will be done by injecting different traffic loads to this network from various application types (heavy/light)

3.2.3 The hardware of network:

First the Devices are (8 Ethernet workstations: as 4 clients (Type Of Service) TOS-sender, and 4 servers TOS-receiver), (102 routers), (2 Switches) and (4 logical network).

Client: the (Ethernet wkstn adv) node model represents a workstation with client-server applications running over TCP/IP and (User Datagram Protocol) UDP/IP. The workstation supports one underlying Ethernet connection at 10 Mbps, 100 Mbps, or 1000 Mbps.

Switch: the (ethernet8_switch_base) node model represents a switch supporting up to 8 Ethernet interfaces. The switch implements the Spanning Tree algorithm in order to ensure a loop free network topology. Switches communicate with each other by sending Bridge Protocol Data Units (BPDU's). Packets are received and processed by the switch based on the current configuration of the spanning tree.

Router: the (ethernet2_slip8_gtwy_base) node model represents an IP-based gateway supporting up to two Ethernet interfaces and up to 8 serial line interfaces at a selectable data rate. IP packets arriving on any interface are

routed to the appropriate output interface based on their destination IP address. The Routing Information Protocol (RIP) or the Open Shortest Path First (OSPF) protocol may be used to automatically and dynamically create the gateway's routing tables and select routes in an adaptive manner.

Logical network: logical network contains routers, the type of routers used are (the ethernet2_slip8_gtwy_base) connected together in series using Ethernet-1000 Base Advance link. First router named (node_0) connected to the client and last router named (node_39) connected to switch A. And so on up to logical network 4.

The figure (3-5) illustrated an example of one of logical network.

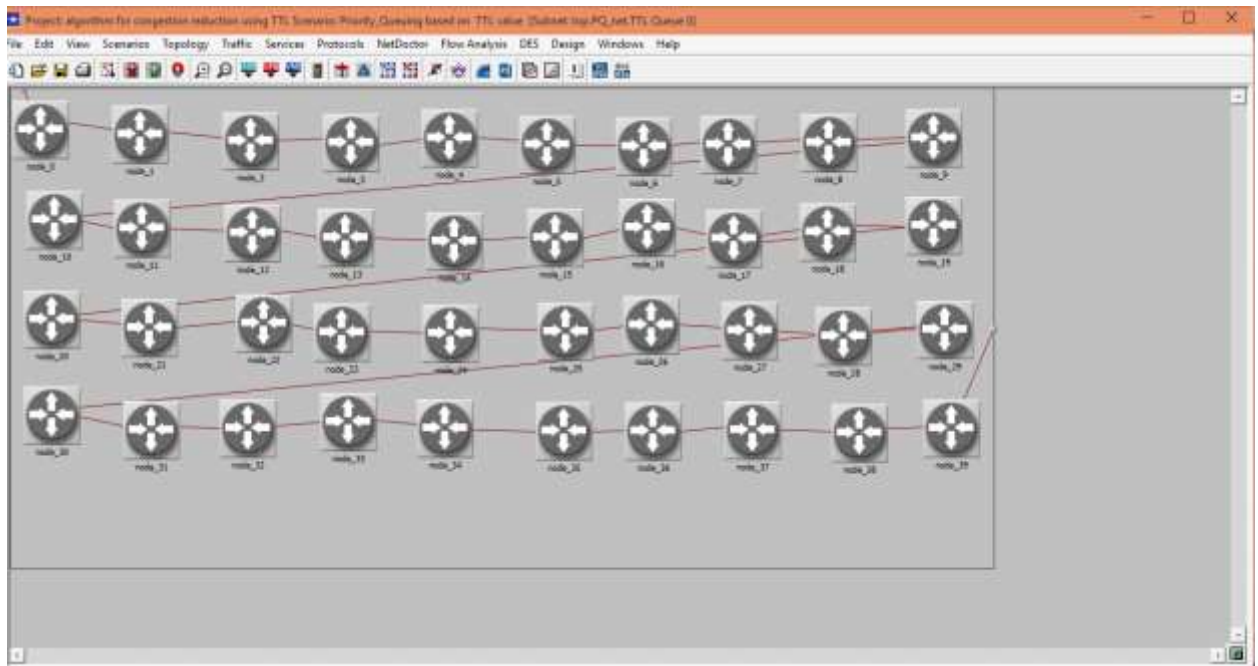


Figure (3-5): connected routers in logical network

Physical Links: Second the links there are (110) Ethernet link, type: 1000 Base Advance) shown in blue color in figure below and (1 Serial) in red color.

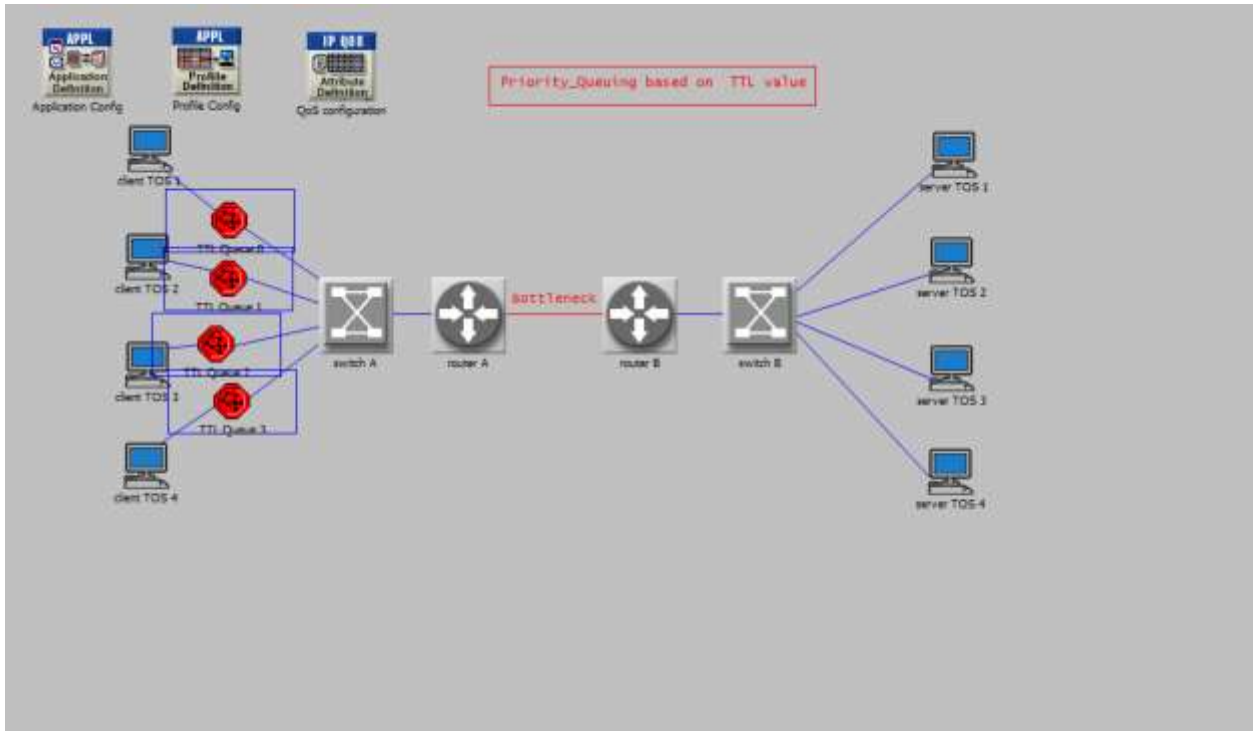


Figure (3-6): links used in network

3.2.4 The software of network:

Consists of (3 Configuration Utilities):

Profile configuration: the Profile Configure node can be used to create user profiles. These user profiles can then be specified on different nodes in the network to generate application layer traffic. The application defined in the Application Configure objects are used by this object to configure profiles. Therefore, you must create applications using the Application Configure object before using this object. You can specify the traffic patterns followed by the applications as well as the configured profiles on this object.

Application configuration: the Application Configuration node can be used for the following specifications:

- 1. ACE Tier Information:** Specifies the different tier names used in the network model. The tier name and the corresponding ports at which the tier listens to incoming traffic is cross-referenced by different nodes in the network.
- 2. Application Specification:** Specifies applications using available application types. You can specify a name and the corresponding description in the process of creating new applications. For example, "Web Browsing Heavy Hypertext Transfer Protocol (HTTP 1.1)" indicates a web application performing heavy browsing using HTTP 1.1. The specified application name will be used while creating user profiles on the Profile Configuring object.
- 3. Voice Encoder Schemes:** Specifies encoder Parameters for each of the encoder schemes used for generating voice traffic in the network.

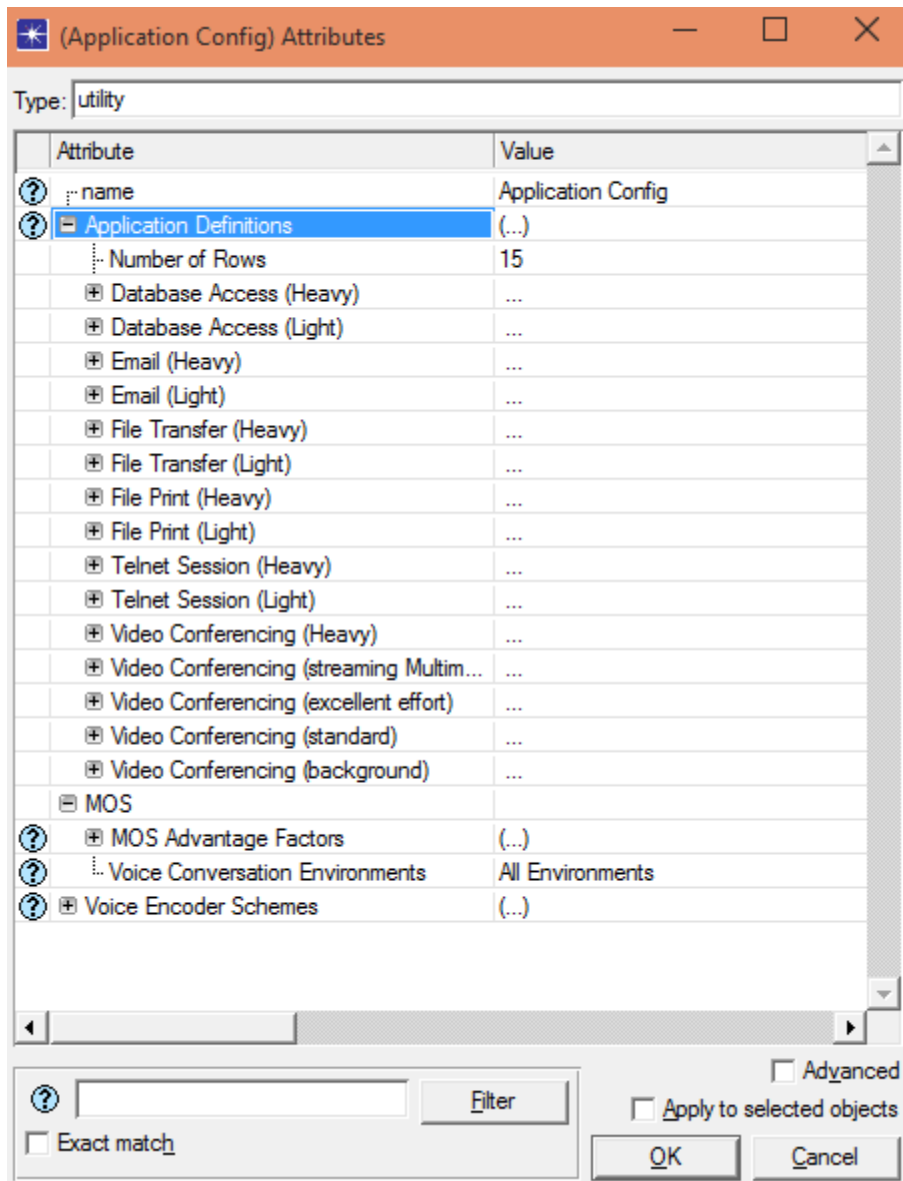


Figure (3-7): application configuration

Quality of service configuration (QoS): defines attribute configuration details for protocols supported at the IP layer. These specifications can be referenced by the individual nodes using symbolic names (character strings.)

1. Queuing Profiles: Defines different queuing profiles such as FIFO, WFQ (weighted Fair Queuing), Priority Queuing, Custom Queuing, MWRR, MDRR and DWRR.

2. CAR Profiles: Defines different CAR profiles that can be used in the network.

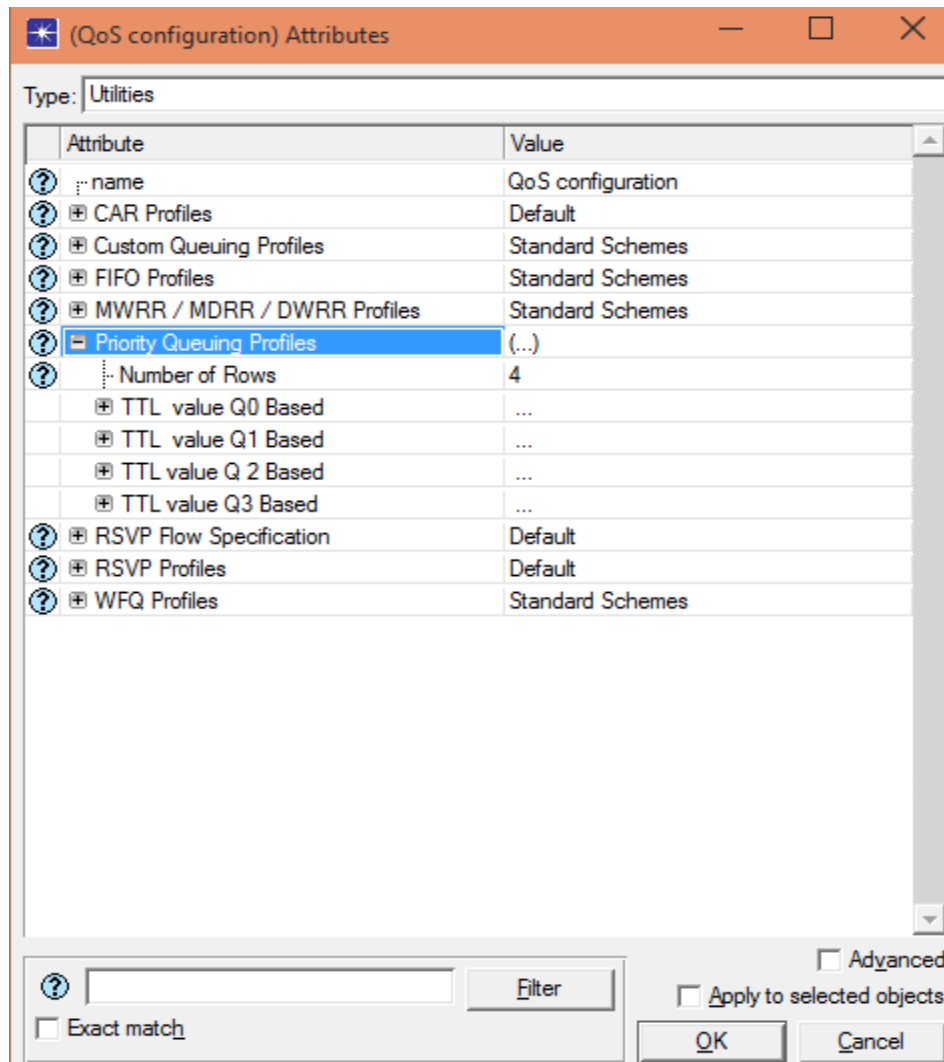


Figure (3-8): QoS configuration

The network is composed of four pairs of video clients. Each pair uses a distinct TOS (Type of Service) for data transfer. The link between the two routers is a bottleneck. Routers support multiple queues for each type of service. Queue (4) receives TOS (4) traffic, queue (3) receives TOS (3) traffic, queue (2) receives TOS (2) and queue (1) receives TOS (1). Queues are serviced using "Priority Queuing of TTL value" mechanism.

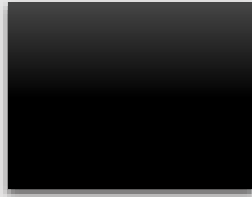
Priority queuing can be enabled on each interface in "advanced" routers. Queuing profile and queuing processing mechanism are set in a sub-attribute called (Interface Information) in the (IP QoS Parameters) compound attribute. Queuing profile defines the number of queues and the classification scheme. Global queuing profiles are defined in the QoS configuration object. This object is found in (utilities) palette. Local Priority queuing profiles (not used in this network) can be configured under (Priority Queue Profiles) in the (IP QoS Parameters) compound attribute on the router.

3.2.5 Network Inventory Summary:

The table below illustrate all devices, physical links and Configuration Utilities are used in network simulation.

Table 3-1: Network Inventory Summary

Element	Type	Count
Devices	Total	112
	Routers	102
	Switches	2
	Workstations	8
	Logical network	4
Physical Links	Total	111
	Serial	1
	Ethernet	110
Other	Configuration Utilities	3



4

Chapter

**Results and
Discussion**

Results and discussion

4.1 Results of priority Queuing based on TTL value algorithm:

This paragraph gives a brief description for the results of the (priority Queuing based on TTL value algorithm) (PQT) using the Opent simulation. On the other hand PQT considers the TTL value as one of the important factors in the queue management decisions. Furthermore, this results has been compared with the results of (IP QoS Priority Queuing algorithm.)

The comparison is based on the following network infrastructure shown in figure (4-1). In this scenario, stream of the packets have been sent form client to server through switch A, router A, switch B, and router B. Since Router A represent a bottleneck, thus in this implementation the above two mentioned queue management algorithm have been compared according to the generated traffic, and accordingly their results has been plotted in the coming graphs (please refer to section 4.2.1 for more information).

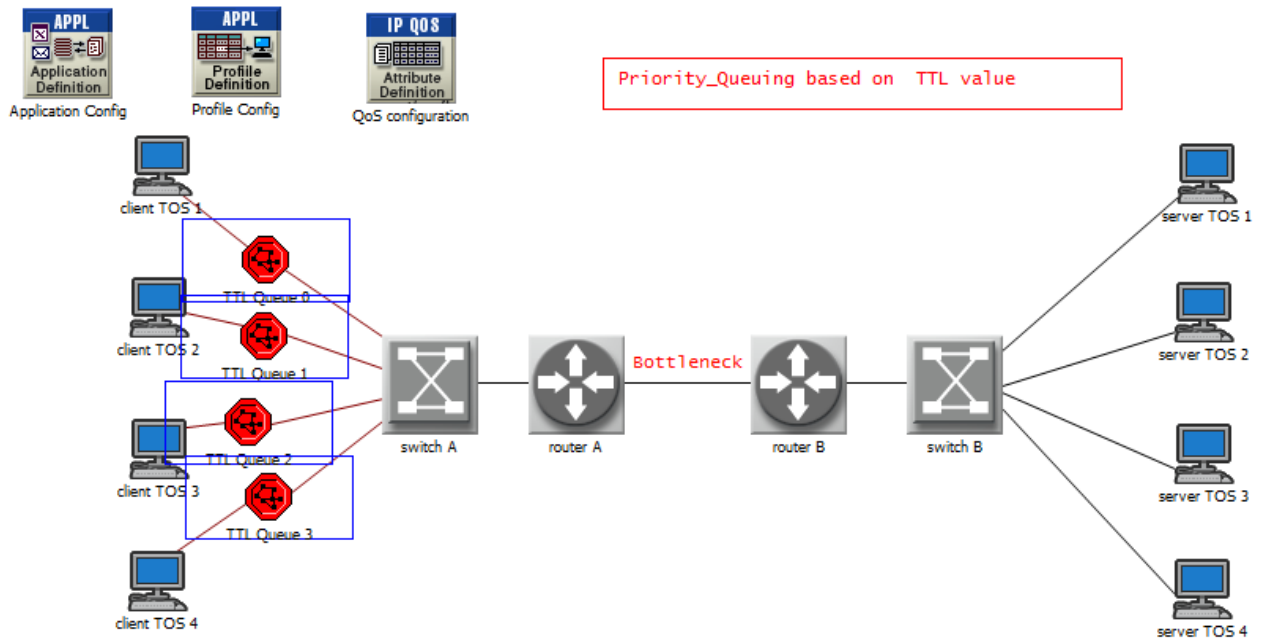


Figure (4-1): Simulation: Network infrastructure Design, Configuration and PQT implementation

4.1.2 Evaluation of PQT through various traffic loads

To examine the performance of PQT compared with IP QoS Priority Queuing, several types of applications are used in simulation (around fifteen different application types have been used), some of them are heavy while the others are considered light. The following examples give a sample of applications to obtain the expected results.

- Database Access (Heavy)
- Database Access (Light)
- Email (Heavy)
- Email (Light)
- File Transfer (Heavy)
- File Transfer (Light)
- File Print (Heavy)
- File Print (Light)
- Telnet Session (Heavy)
- Telnet Session (Light)
- Video Conferencing (Heavy)
- Video Conferencing (streaming Multimedia)
- Video Conferencing (excellent effort)
- Video Conferencing (standard)
- Video Conferencing (background)

4.2 Router A <-> Router B: point-to-point Statistics:

4.2.1 Queuing delay in router A <-> router B:

The Figure (4-2) illustrates Queuing delay between router A and Router B in (second) of proposed (priority Queuing based on TTL value) algorithm represented in red curve, and with compare (IP QoS Priority Queuing) algorithm represented in blue curve. When the traffics flow from router A to router B represent by this symbol (-->). And all application run that mentioned in section (4.1.2). Simulation duration is 600 second. The column represent the delay in second. The row represent the time duration of simulation run.

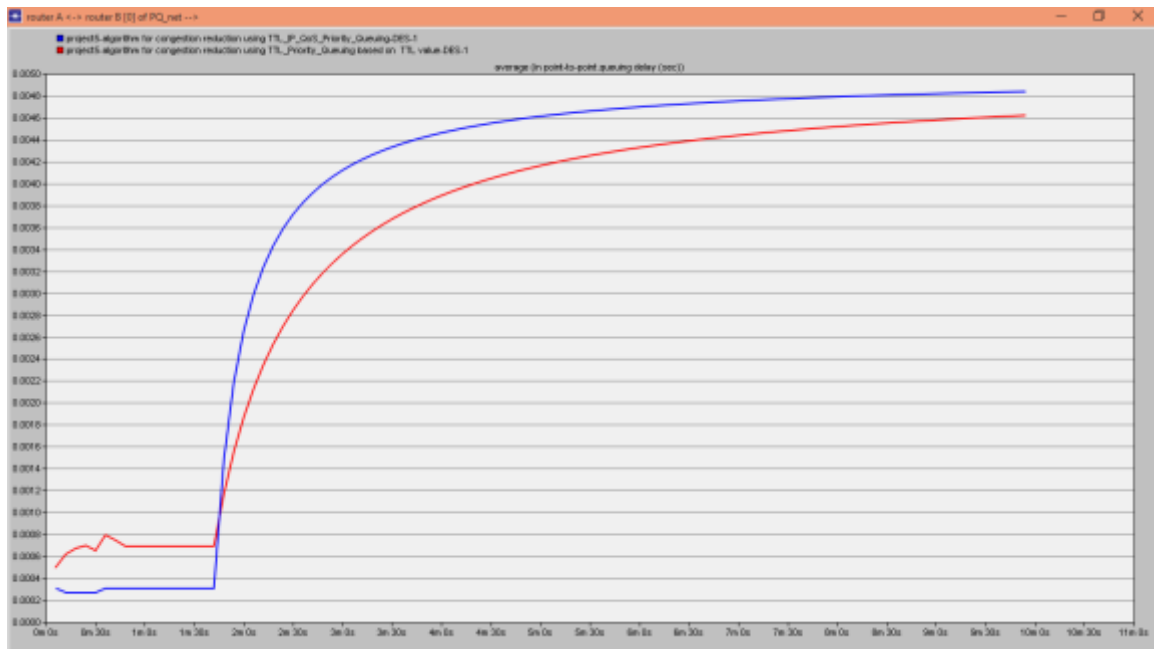


Figure (4-2) illustrates Queuing delay from router A to router B

As can be seen from the figure above the proposed algorithm start at (0.0003s) then after (1m 30s) the delay increased up to (0.0046s), for compared algorithm start at (0.0003s), also after (1m 30s) the delay increased up to (0.0048). And this is clearly from the figure above the delay of proposed algorithm has been decreased about (5%) from compared algorithm.

Figure (4-3) illustrates queuing delay between router A and Router B in (second) separate into two graph one for (priority Queuing based on TTL value) and the other one is (IP QoS Priority Queuing). Max queuing delay value for both algorithm explained in the figure (4-2). In this graph illustrated the zooming of the delay that shown in figure (4-2).

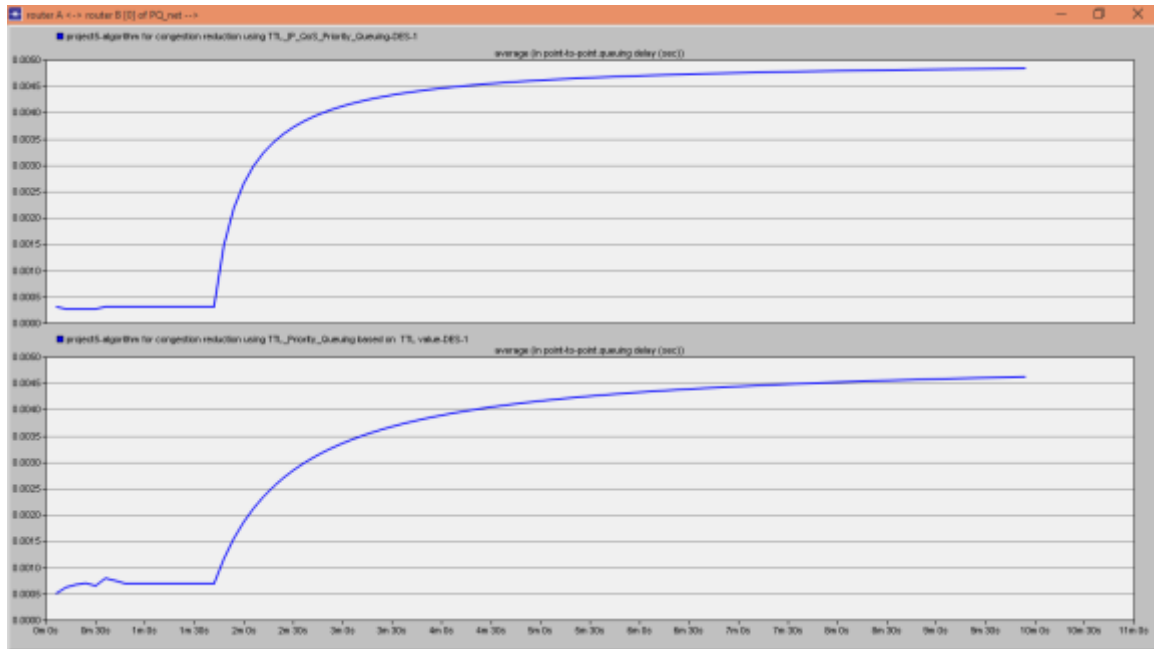


Figure (4-3) the zooming illustrates Queuing delay from router A to router B

As can be seen from figure (4-3), it is clearly seen that the queuing delay is decreased for this design of big network and with heavy applications.

Figure (4-4) illustrates Queuing delay between router B and Router A in (second) for proposed algorithm in this research (priority Queuing based on TTL value) represented by (red curve) and the other method (IP QoS Priority Queuing) represented by (blue curve). When the traffics stream from router B to router A represent by this symbol (<-). Simulation duration is 600 second.

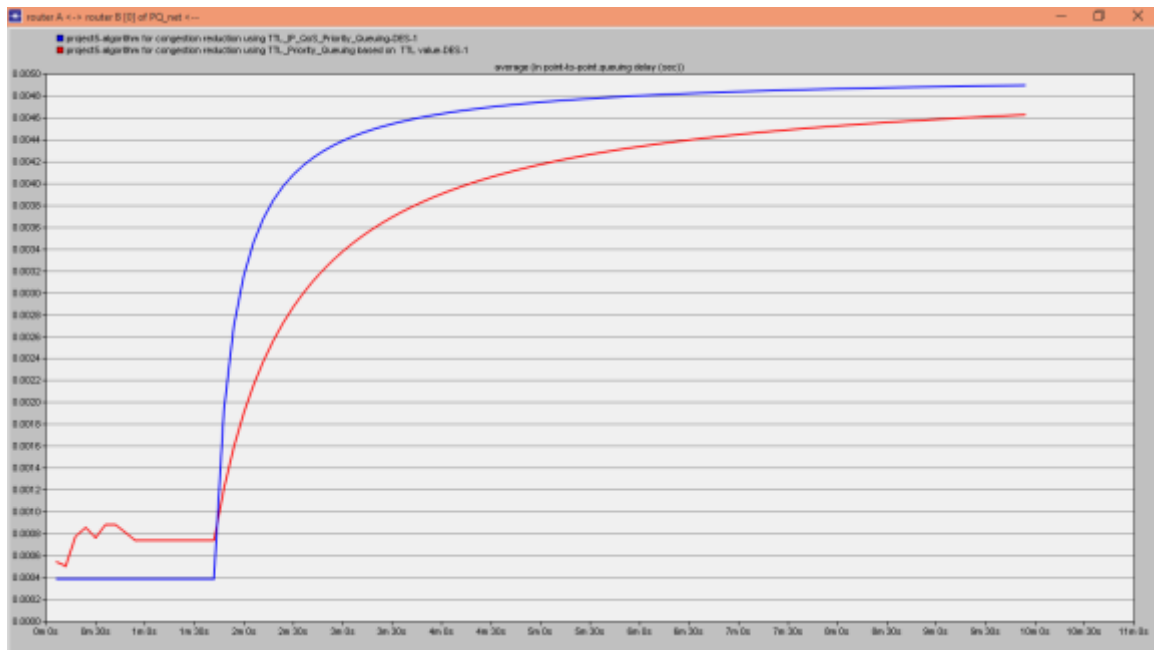


Figure (4-4) illustrates Queuing delay from router B to router A

As can be seen in above figure the delay of comparing algorithm starts after (1m 30s), and then begin arising up to (0.0049). At the same point the proposed algorithm starts the delay, then increasing up to (0.0046). It is obviously seen that the delay of proposed algorithm is less than the compared algorithm about (7%) that mean in figure (4-4) the delay is better than the delay in figure (4-3).

Figure (4-5) illustrates Queuing delay between router B and Router A in (second) separated into two graph one for (priority Queuing based on TTL value) and the other one is (IP QoS Priority Queuing). Max value for both algorithm explained in figure (4-4). In this graph illustrated the zooming of the queuing delay is shown in figure (4-4).

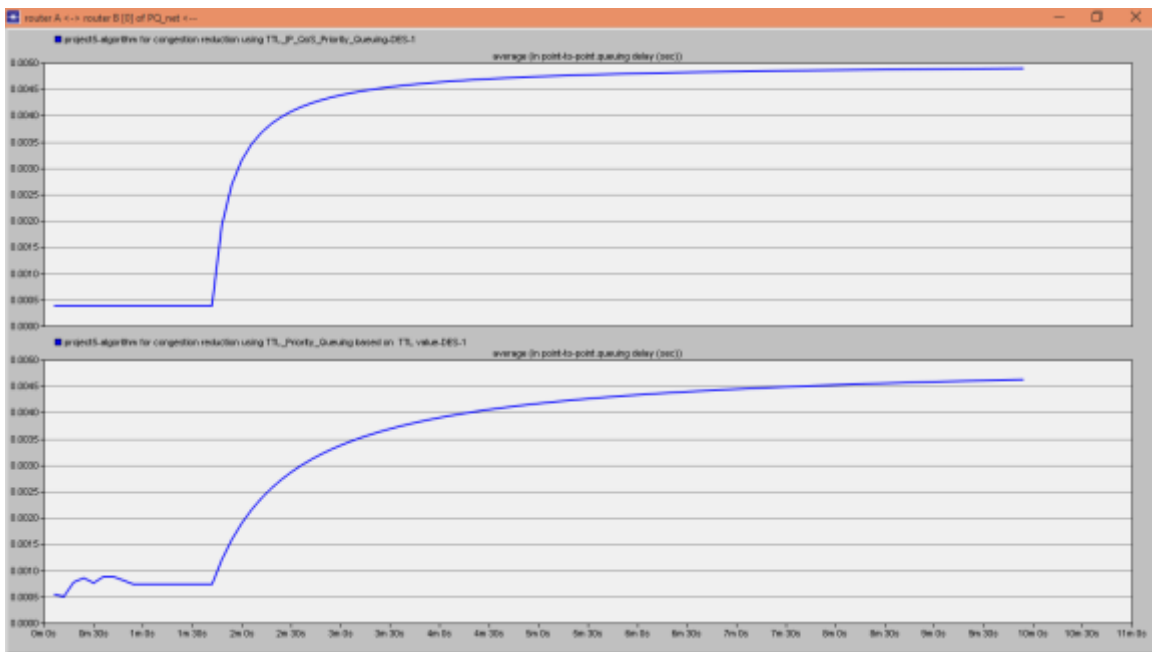


Figure (4-5) illustrates the zooming Queuing delay from router B to router A

As can be seen from figure (4-5), it is clearly seen that the queuing delay of proposed algorithm is decreased about (7%) from compared algorithm.

4.2.2 Throughput (-- >) (packet / sec):

Figure (4-6) illustrates average Throughput (packets/sec) for both algorithms (priority Queuing based on TTL value) and (IP QoS Priority Queuing). The traffic stream from router A to router B. Simulation duration is 600 second, The Column shows the packers (packets) and the row shows the time (sec).

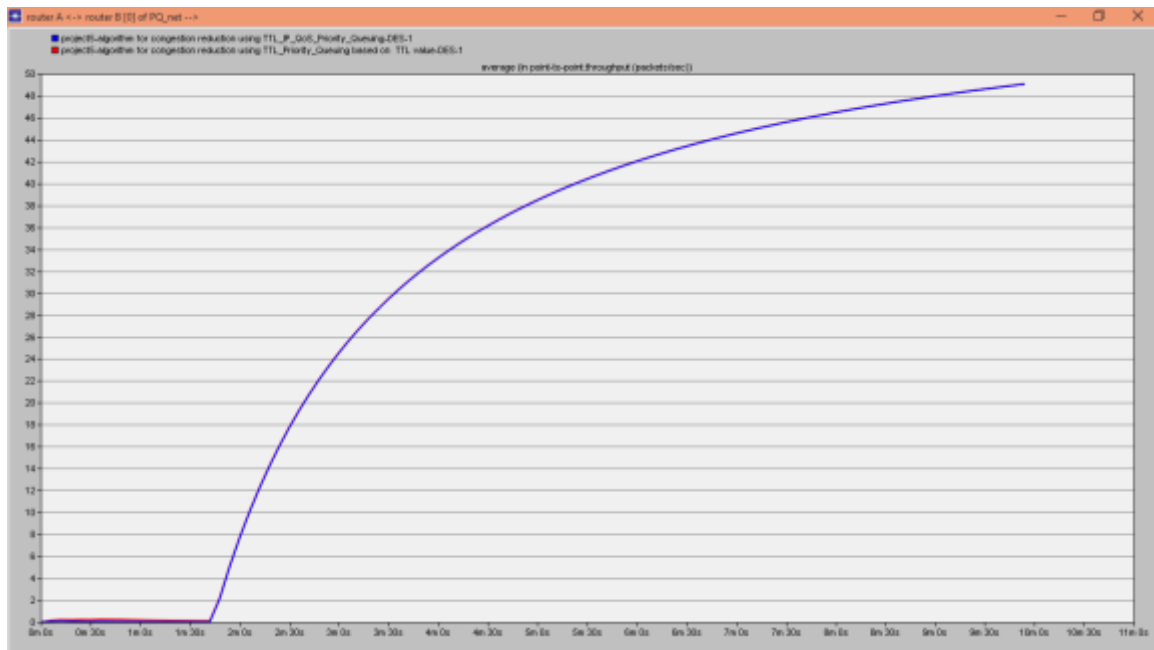


Figure (4-6) illustrates the average Throughput (-- >) (packet / sec)

As can be seen from the above figure shows that the values of the throughput for both algorithms are same and they have the value (50) packets. The throughput starting after (1m.30s) of time and began increased up to (50) packets. It apparently show that the throughput is equal for two algorithms. That is mean, this result is reasonable for propose algorithm.

Figure (4-7) illustrates the average in throughput in (packets/sec) for both algorithms separated in two graph. The upper graph demonstrates (IP QoS Priority Queuing) algorithms and lower graph shows (priority Queuing based on TTL value) algorithms. The two graphs explained in figure (4-6). They show the zooming of the average throughput.

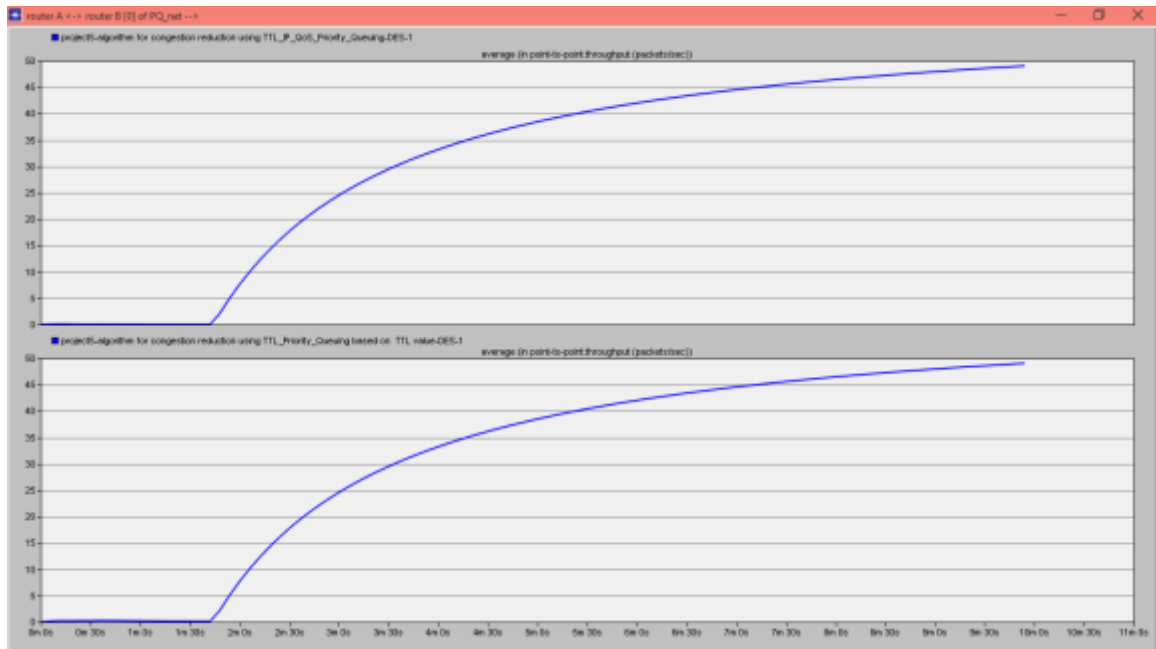


Figure (4-7) illustrates the zooming average in throughput (packets/sec)

As can be seen from the figure both algorithms start after (1m 30s), the upper curve that represents compare algorithm and lower curve representing proposed algorithm. By looking to the two curves it can see that are similar, as shown in figure (4-6).

4.3 Router A Statistics:

4.3.1 IP Processing Delay (sec):

Figure (4-8) illustrates the average in IP Processing Delay (sec) in router A for both algorithms (priority Queuing based on TTL value) represented by (red curve) and (IP QoS Priority Queuing) represented by (blue curve). The Column shows the time in (second) and the row shows the time duration of simulation. Simulation duration is 600 second.

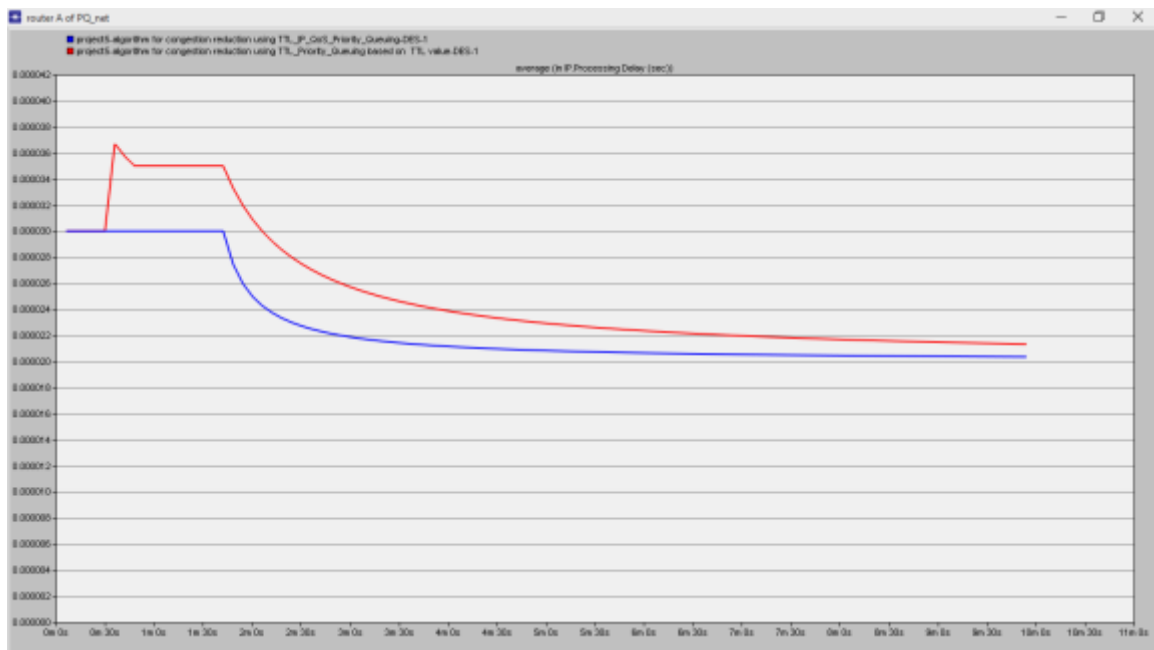


Figure (4-8) illustrates the average in IP Processing Delay (sec)

As can be seen from the figure above the average IP Processing Delay (sec) in (priority Queuing based on TTL value) algorithm and (IP QoS Priority Queuing) algorithm begin together at same time (0.000030s), then with TTL rising up to (0.000036s) then the two algorithms decrease down between (0.000020s) and (0.000022s). It is clearly seen that the algorithm with TTL increased from compare algorithm in processing delay with percentage (9%)

4.3.2 CPU Utilization in Router A:

Figure (4-9) illustrates average Central Process Unit (CPU) utilization in router A for both algorithms separated in two graphs. Simulation duration 600 second, the blue curve represents the (IP QoS Priority Queuing) algorithm and the Red curve represents the proposed algorithm.

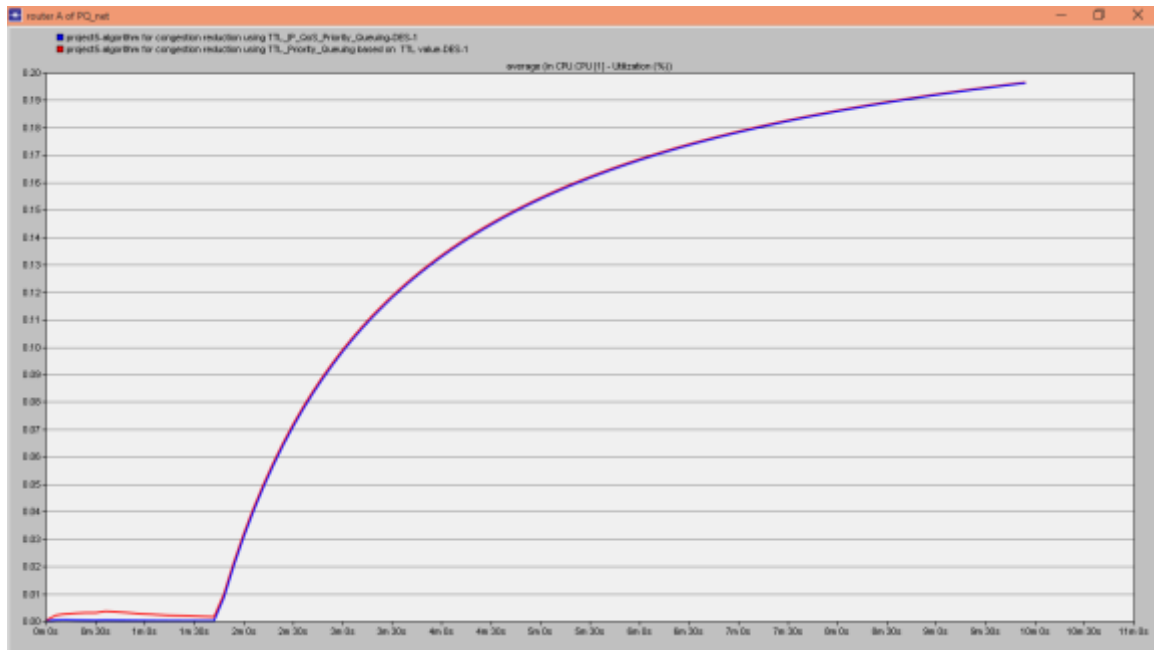


Figure (4-9) CPU Utilization of Router A (overlaid)

As can be seen from the figure above the compared algorithm is beginning utilization after (1m 30s) then rising up to over (0.20) with time duration (10m 30s). The proposed algorithms begins at the same time and continue rising up to same value (0.20), observed that CPU utilization of proposed is began higher than compared then be equal after (1m 30s) in utilization in CPU.

Figure (4-10) illustrates average CPU utilization in router A for both algorithms separated in two graphs. Simulation duration 600 second. But the curve is separated in two: first one is for (IP QoS Priority Queuing) while the second is for proposed algorithm to show the curve in zooming.

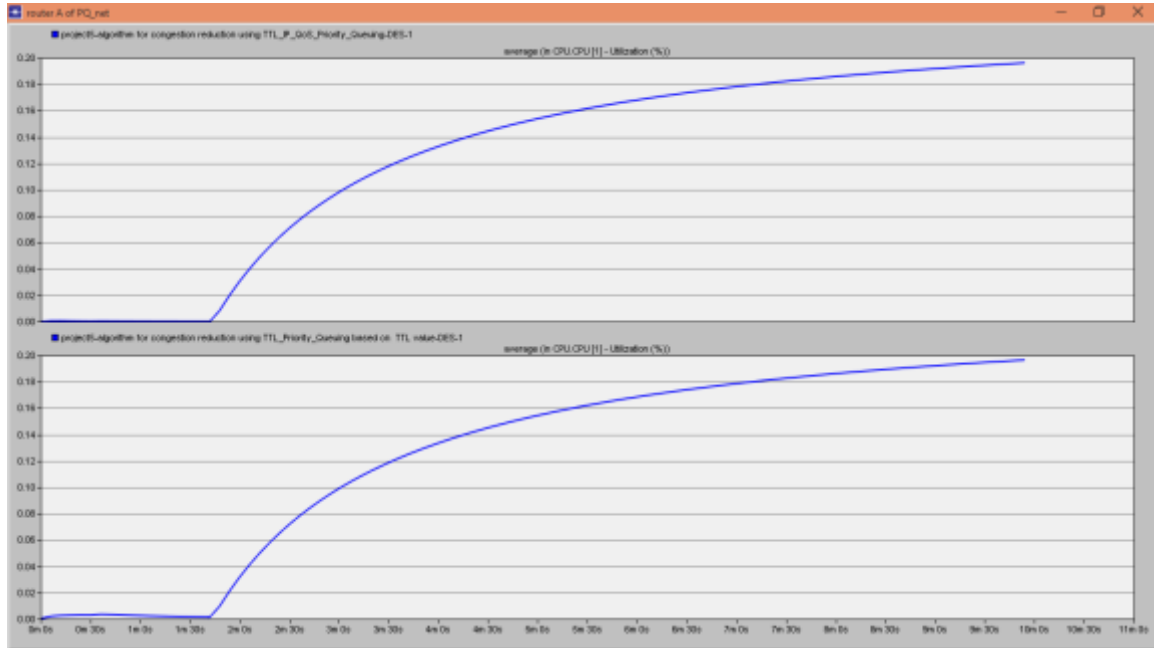


Figure (4-10) CPU Utilization of Router A (stacked)

4.3.3 Switch A <-> Router A – in point-to-point - queuing delay (sec) -->:

Figure (4-11) illustrates average in point-to-point queuing delay in switch A and router A, for both algorithm. Simulation duration 10 minutes.

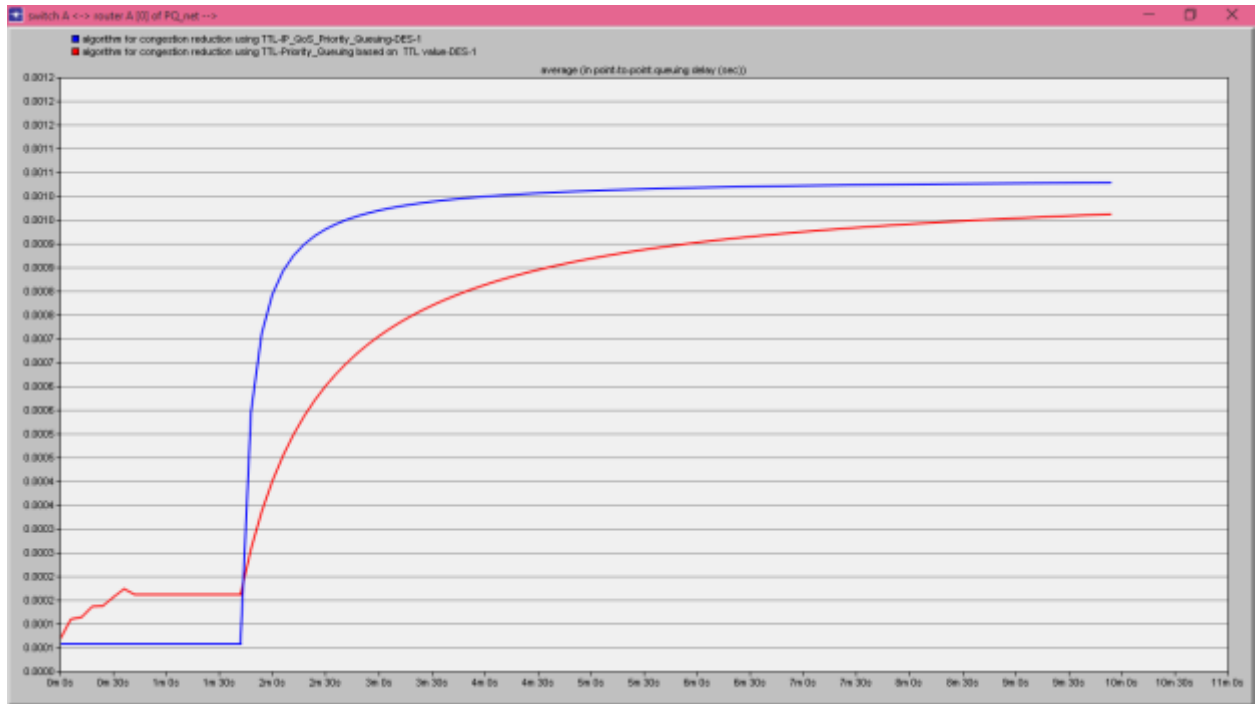


Figure (4-11) Switch A <-> Router A – in point-to-point - queuing delay (sec) -->

As shown in the figure (4-11) the packets passing from switch A to router B, represent by this symbol (-->).

The red curve represent proposed algorithm it begin at (0.0001) s and go up to (0.0002) s until (1m 30s) then the delay increase up to (0.0010) s at (10m 30s). For another curve with blue color represent (IP QoS Priority Queuing) it begin at (0.0001) until (1m 30s), then the delay increase up to near (0.0011) at (10m 30s). Observe that queuing delay in proposed algorithm start high than compared algorithm, but after (1m 30s) time, the queuing delay starting on decrease.

That mean queuing delay in proposed algorithm is better than other algorithm.

Figure (4-12) illustrates average in queuing delay (sec) (<-->) point-to-point queuing delay in switch A and router A, for both algorithm. Simulation duration 10 minutes.

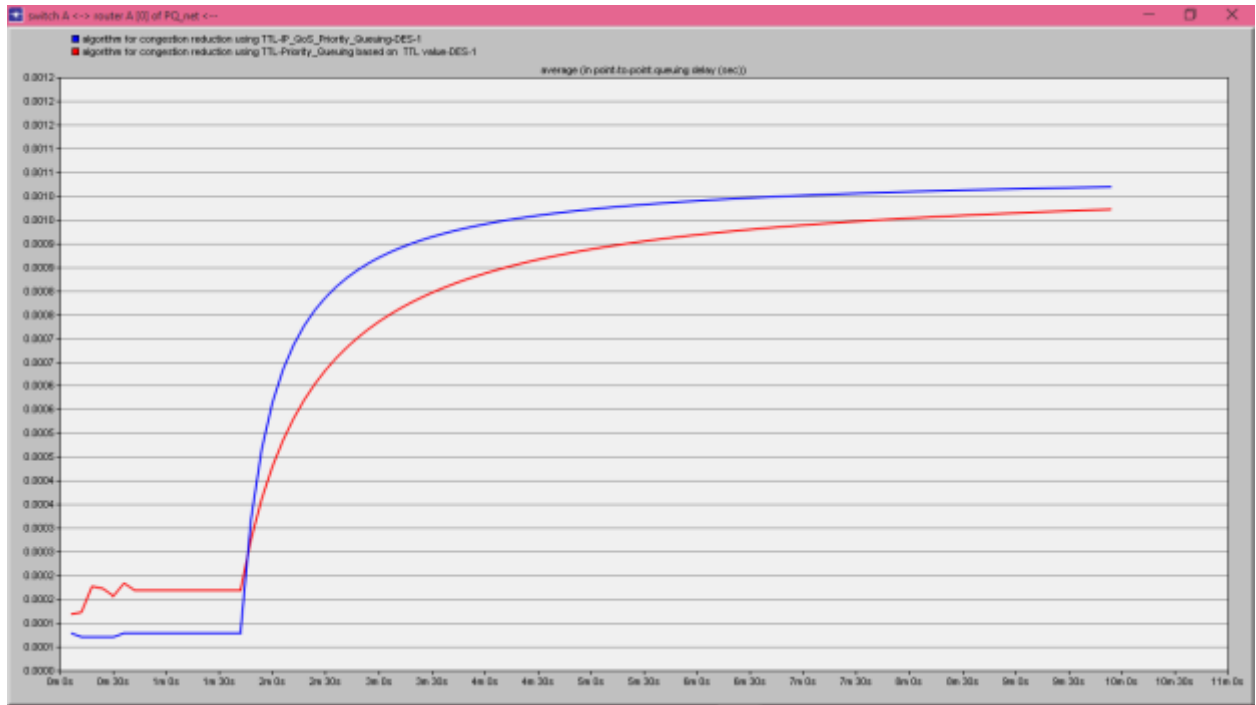


Figure (4-12) Switch A <-> Router A – in point-to-point - queuing delay (sec) (<-->)

As shown in the figure (4-12) the packets passing from router A to switch A, represent by this symbol (<-->). The queuing delay for both algorithm in this figure is less than the queuing delay in figure (4-11) especially, compared algorithm.

Figure (4-13) illustrates average in point-to-point queuing delay in switch A and router A, for both algorithm the packets passing from and to switch A, router A. Simulation duration is 10 minutes. Represent by these symbol (<--), (-->).

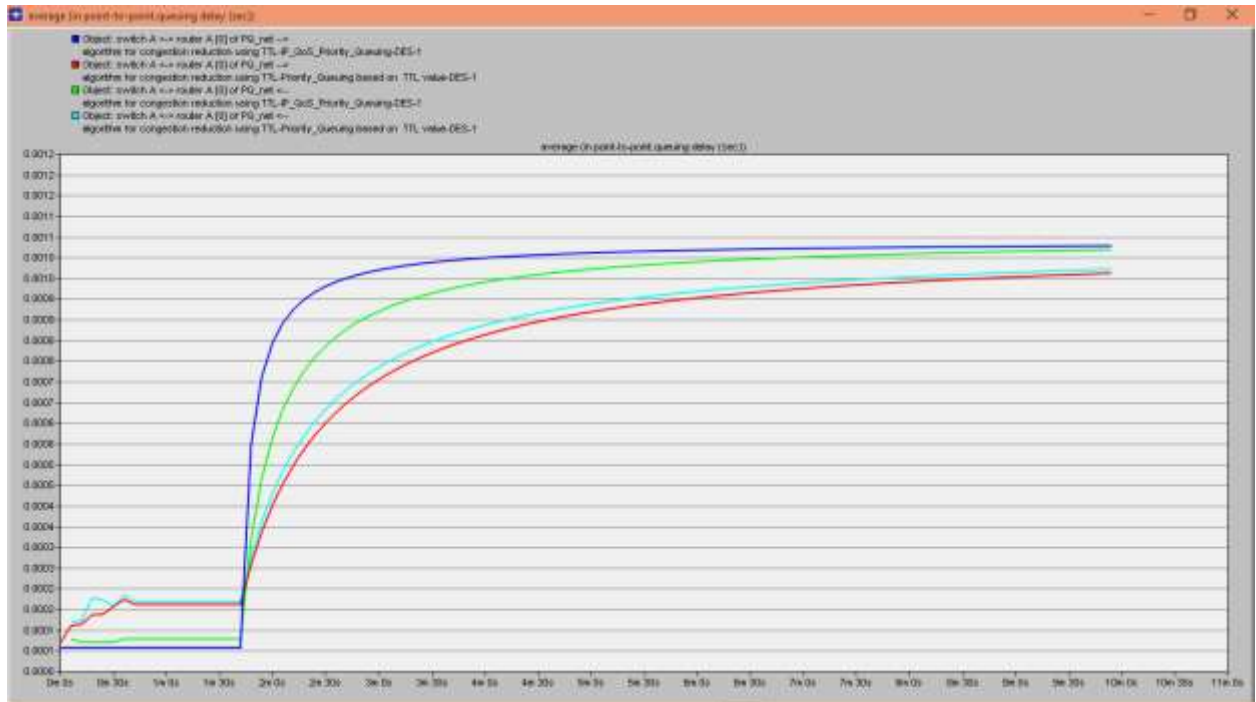
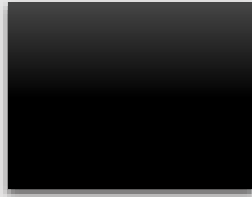


Figure (4-13) Switch A <-> Router A in point-to-point-queuing delay (sec) (<--) (-- >)

As shown in the figure above the red and poplars curve for proposed algorithm, and blue, green curve for compared algorithm. Each curve explained in Figure (4-11) and Figure (4-12). But in this figure companied them in one curve to seen complete picture of queuing delay.



5

Chapter

**Conclusion and
Recommendations**

Conclusion and Recommendations

5.1 Conclusion:

This research proposed a new algorithm (priority Queuing based on TTL value) for re-ordering the packets in the router queue which is based on the Time-To-Live (TTL) value of the packet. This algorithm enables the router to reduce the overall delay, it also reduces the packet dropping rates as well as the overall delay for the packets with lower TTL. For more information please refer to figure (4-4).

As can be seen in figure (4-4), it is clearly seen that the proposed algorithm outperform to algorithm0 (IP QoS Priority Queuing) in term of delay, also it contributes to the congestion avoidance since it reduce the overall retransmission rate. Overall means that we are dealing with packet per multiple hops.

This algorithm is useful for core routers in the telecom operator's backbone rather than the edged routers in the enterprise networks (since there TTL values will be quite similar), and this will indirectly enhance the response time for far users.

This algorithm guaranty that the router can forward a completely and safety packets to receiver as much as possible with little delay and a few retransmission with less congestion.

5.2 Recommendations:

There's a room for improvement waiting the future researchers such as:

Although this algorithm is considered as a great support for queue management in core routers, future researchers can apply it in all of edged and core routers as well as for check the overall performance it could be done by checking certain performance metrics.

This algorithm deals with TTL as a very important metric for queuing priority, however this algorithm ignores the phenomenon of duplicated packets (the same packet with the same TTL) will get higher priority rather than other packets which consider a waste of resource. So future researchers must consider the duplicated packets.

References:

- [1] Zhu, C., et al., A comparison of active queue management algorithms using the OPNET Modeler. *Communications Magazine, IEEE*, 2002. 40(6): p. 158-167.
- [2] Comer, D., *Internetworking with TCP/IP Vol. 1: Principles, Protocols and Applications*. Prentice-Hall, 2000.
- [3] Comer, D.E., *Computer networks and internets*. 2008: Prentice Hall Press.
- [4] Nichols, K. and V. Jacobson, Controlling queue delay. *Communications of the ACM*, 2012. 55(7): p. 42-50.
- [5] Villamizar, C. and C. Song, High performance TCP in ANSNET. *ACM SIGCOMM Computer Communication Review*, 1994. 24(5): p. 45-60.
- [6] Appenzeller, G., I. Keslassy, and N. McKeown, Sizing router buffers. Vol. 34. 2004: ACM.
- [7] Bashir, O.A., Y.A. Mohamed, and M.A. Elshaikh, The automatic calculation of the length of the queues (auto). *Journal of Engineering and Computer Science*, 2015. 16(2).
- [8] Ott, T.J., T. Lakshman, and L.H. Wong. Sred: stabilized red. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE. 1999. IEEE*.
- [9] Modeler, O., *OPNET Technologies Inc Documentations*, 2008.
- [10] Psaras, I., V. Tsaoussidis, and L. Mamas. CA-RTO: a contention-adaptive retransmission timeout. In *Computer Communications and Networks, 2005. ICCCN 2005. Proceedings. 14th International Conference on. 2005. IEEE*.

- [11] Psaras, I., Transmission and Retransmission Scheduling for Terrestrial and Space Internetworks. 2008.
- [12] Arianfar, S., P. Sarolahti, and J. Ott. Deadline-based resource management for information-centric networks. in Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking. 2013. ACM.
- [13] Kaur, J., et al., Analyzing Load and Delay in Wireless LAN using TTL and Fragmentation. 2014.
- [14] Klampfer, S., J. Mohorko, and Z. Cucej, Impact of hybrid queuing disciplines on the VoIP traffic delay. Electrotechnical Review, 2009.
- [15] Vijayakumar, M., V. Karthikeyani, and M. Omar, Implementation of Queuing Algorithm in Multipath Dynamic routing architecture for effective and secured data transfer in VoIP. International Journal of Engineering Trends and Technology, 2013. 4(4): p. 1226-1230.
- [16] Bindra, P., J. Kaur, and G. Singh, Investigation of Optimum TTL Threshold value for Route Discovery in AODV. International Journal of Computer Applications, 2013. 79(9): p. 45-49.
- [17] Matthew, S.M.M.M.T. and N.S.P.H. Obiomon, Modeling and Simulation of Queuing Scheduling Disciplines on Packet Delivery for Next Generation Internet Streaming Applications.
- [18] Psaras, I. and V. Tsaoussidis, On the properties of an adaptive TCP Minimum RTO. Computer Communications, 2009. 32(5): p. 888-895.

- [19] Almofary, N., H. Moustafa, and F. Zaki, Optimizing QoS for voice and video using diffserv-MPLS. *International Journal of Modern Computer Science & Engineering*, 2012. 1(1).
- [20] Kesselman, A. and Y. Mansour, Optimizing TCP retransmission timeout, in *Networking-ICN 2005*. 2005, Springer. p. 133-140.
- [21] Balasundaram, K., T. Velmurugan, and R. Suresh, Performance Analysis of Queuing Disciplines for Difference Services Using OPNET Tool. *International Journal of Scientific Engineering and Technology*, Volume, 2013(3): p. 47-50.
- [22] Mohammed, H.A., A.H. Ali, and H.J. Mohammed, The Affects of Different Queuing Algorithms within the Router on QoS VoIP application Using OPNET. *arXiv preprint arXiv:1302.1642*, 2013.

Appendix A

```
#include <iostream.h>
#include <conio.h>
int main (int argc, char* argv[])
{
    const limit=100;
    float x[limit],smallest;
    int k,i,j,spos,size;

    cout <<"\nThis is routine simulate how can (priority Queuing based on
TTL value) PQT algorithm works, using C++ program \n" ;

    cout <<"\nDetermine the queue size of TTL value from key board
then press enter : \n" ;

    cin >> size ;

    cout <<"Inter the value of TTL from key board then press enter after
each value : \n" ;

    for (k=0; k < size ; k++)
        cin >> x[k] ;

    //now start re-ordering the value of TTL from smallest to biggest.
    for (i=0; i<size-1 ; i++)
    {
        smallest = x[i]; spos = i ;
        for (j = i ; j < size ; j++)
            if (x[j] < smallest)
```

```

        {
            smallest = x[j];
            spos = j;
        }
        x[spos] = x[i];
        x[i] = smallest ;
    } // for i

    //now print on screen the re-order TTL value from smallest to biggest
    according to the PQT algorithm

    cout << "The re-order of TTL value from smallest to biggest according
    to the PQT algorithm is : \n" ;

    for (k=0; k<size; k++)
        cout << x[k]<< " ";

    getch();
return 0;
}

```