Simulation Studies of Diffserv Policies for the Internet Traffic

L. Audah[1], N. Jusoh[1], A. Jamil[1], J. Abdullah[1], S.A. Hamzah[1], N.A.M. Alduais[1], M.A.A. Razak[2]
[1]*Optical Communications and Network Research Group (OpCoN), Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, Parit Raja, 86400 Batu Pahat, Johor, Malaysia.*
[2]*Biomedical Instrumentation and Electronics Research Group (BMIE), Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310 UTM Johor Bahru, Johor, Malaysia.*
hanif@uthm.edu.my

***Abstract*—Differentiated Services (Diffserv) is the Internet architecture that uses the queuing management schemes to provision the traffic flows in the Internet backbone system. It discriminates traffic flows to a finite aggregate of classes and provides scalability solution by simplifying the complexity functions at the edge routers. In this paper, we study the end-to-end (e2e) Quality of Service (QoS) performance of File Transfer Protocol (FTP) and Constant Bit Rate (CBR) traffics transmitted over a Diffserv network. The Diffserv system applied the Token Bucket, Time Sliding Window Three Color Marker (TSW3CM) and Single Rate Three Color Marker (SRTCM) traffic provisioning policies. The e2e QoS parameters include delay, jitter, loss ratio and throughput are analyzed and compared among the policy types against the increment of traffic connections in the network system. We conclude that the FTP traffic could achieved the best overall delay performance using SRTCM policy and the best jitter performance using TSW3CM. The lowest overall loss ratio and the best throughput for FTP could be achieved using Token Bucket. Besides that, the CBR traffic has achieved the best overall delay performance using TSW3CM policy while the SRTCM policy provides the best jitter, loss ratio and throughput performances. The future works aims to design the combination of QoS aware routing, scheduling, and Diffserv queuing schemes that can adaptively maintain QoS for each type of traffic at optimum level.**

***Index Terms*—Diffserv; Token Bucket; TSW3CM, SRTCM; QoS; NS-2.**

I. INTRODUCTION

The advent of Diffserv architecture has been initiated by the Internet Engineering Task Force (IETF) a long time ago as a better solution to provide QoS guarantees in the Internet protocol (IP) networks. Compared to its predecessor like the Integrated Services (Intsev) which provides services based on per micro flow state, Diffserv outsmarts Intserv in providing better e2e QoS and preferential treatment for large heterogeneous networks system [1]. Diffserv discriminates different traffic flows which have the same commonalities to finite aggregate of classes and provides a more scalable solution for e2e QoS in IP networks by simplifying the complexity functions such as traffic classification and traffic conditioning within the edge routers [2] [3].

Previous related studies by Kaur et al in [4] have analyzed the QoS parameters for the Internet traffic using Token Bucket, Round Robin and Priority based Diffserv algorithms. The study focus on the queue buffer occupancy for each node in order to calculate the number of packet sent, number of received packets and number of lost packets. Besides that, the study in [5] has proposed a Modified an Adaptive Factor Provision Aware Proportional Fair Sharing Three Color Marker (MAFPAPTCM) to improve the existing PAPTCM algorithm in term of fairness in bandwidth utilization. The study compares the new algorithm with SRTCM, TSW3CM and Two Rate Three Color Marker (TRTCM) and concludes that the new algorithm is better in term of fairness in bandwidth utilization.

In addition, the study in [6] has proposed a framework that provides an enhanced utilization of network resources through adaptive routing path selection process. The framework uses interior getaway protocol (IGP) for path discovery mechanism and QoS-aware policies for configuring the network elements. Moreover, the study in [7] has proposed a mathematical model for multi-flows QoS configurations that facilitates efficient property based verification over a large network. The study has analyzed the efficiency and scalability of the model for per-hop behavior (PHB) by varying the number of nodes used in network system configurations. Therefore, we summarize that none of the previous related studies have compared the e2e QoS performances of multi-flow FTP and CBR traffics using various Diffserv policies.

This paper aims to analyze and compare the e2e QoS performance parameters of FTP and CBR traffics (i.e. delay, jitter, loss ratio and throughput) using different types of Diffserv policies (i.e. Token Bucket, TSW3CM and SRTCM). Simulations analyses have been done from the network layer perspective using multiple Internet protocol (IP) connections of FTP and CBR traffics. The simulation results provide insight on the e2e performance comparison of each Diffserv policy which is used for IP communication over the Internet system. The remainder of this paper is organized as follows: Section 2 explains the NS-2 simulation configuration. The results and analysis are discussed on Section 3. Finally, Section 4 concludes the findings and suggests future research works.

II. SIMULATION SETUP

The simulation setup characterizes a simple ubiquitous e2e Internet system as depicted in Figure 1. The network simulations are done using NS-2.35 [8] and AWK programming tools [9]. The next subsections explain the details of network elements parameters involved in the simulations.

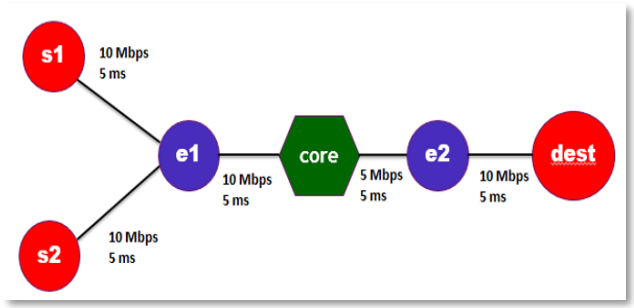

Figure 1: QoE versus Brightness (%) level in MOS

A. *Network Configuration*

The network configuration consists of 2 remote source nodes and a destination node connected via the Diffserv network as shown in Figure 1. Each remote source node is connected to the Diffserv Edge node via 10 Mbps full-duplex link with 5 ms of link delay. The Diffserv network is represented by 2 Edge nodes (i.e. ingress and egress) and a core node. The ingress (i.e. e1 node) is connected to the code node via 10 Mbps full duplex link and 5 ms link delay while the egress node (i.e. e2 node) is connected to the code via 5 Mbps full duplex link and 5 ms of link delay. The lower link bandwidth configuration is to reflect the bottleneck of a network system. The S1 node is connected to the FTP traffic while the S2 node is connected to the CBR traffic applications. Both traffics used 1000 bytes of packet size. The CBR traffic used constant bit rate of 100 kbps per connection. The transmission rate of FTP traffic depends on the maximum congestion window size which is approximately 50 packets. Simulations are done by incrementing each traffic connections from 2 until 12 for each Diffserv policy.

B. *Diffserv Queues Configurations*

The Diffserv used random early detection (RED) queue in edge and core routers. The RED queue consists of a single physical queue and 3 virtual queues to represent 3 per-hop-behavior (PHB) code precedences. The first level of virtual queue precedence (i.e. PHB code point 10) has the minimum and maximum buffer size thresholds of 20 and 40 packets respectively with 0.02 packet dropped probability. The second level of virtual queue (i.e. PHB codepoint 11) has the minimum and maximum buffer size thresholds of 10 and 20 packets respectively with 0.1 probability of packets drop. The lowest level of virtual queue precedence (i.e. PHB codepoint 12) has the minimum and maximum buffer size thresholds of 5 and 10 packets respectively with 0.4 probability of packets drop. The threshold and drop probability configurations are set in such a way to enforce strict transmission rate regulations within the Diffserv network.

Table 1 shows the parameter values used in the simulations for each Diffserv policy. The TSW3CM uses committed information rate (CIR) and peak information rate (PIR) with 3 drop precedences. Packet will be marked as 'green' if the rate is below than CIR, 'yellow' if the rate is between CIR and PIR, and 'red' if above PIR values [10]. Token Bucket policy uses CIR and committed burst size (CBS) with 2 drop precedences. Similarly with TSW3CM, packets will be marked as 'green' if the rate is lower than CIR, 'yellow' if between CIR and CBS, and 'red' if above CBS [11]. The SRTCM policy uses CIR, CBS and excess burst size (EBS) with 3 drop precedences. Packets will be marked as 'green' if the rate is below CIR, 'yelow' if between CBS and EBS, and 'red' if above EBS [12].

Table 1
Diffserv network parameters

Parameters	Source 1 (Node S1)	Source 2 (Node S2)
CIR (bps)	1000000	2000000
CBS (bytes)	5000	10000
EBS (bytes)	3000	6000
PIR (bps)	5000	10000
Traffic Type	FTP	CBR
Traffic Rate (bps)	100000	100000

The RED queue will start dropping packets randomly based on the probability of error (i.e. early drop) when the traffic's data rate exceeds CIR or when the the burst size more than CBS and the virtual queue buffer size exceeds the previously mentioned threshold. If the traffic's rate exceeds PIR or the burst size exceeds EBS and the virtual queue buffer size exceeds threshold, the RED queue will drop the packets excessively according to the random drop probability. The main purpose of early drop by Diffserv queues is to signal the traffic source to follow the predetermine service level requirement and to reduce the transmission rate when the network status is closer to congestion.

III. Results and Discussion

The simulation results and analysis have been divided into 4 QoS categories which are the average e2e delay, jitter, packet loss ratio and throughput. The QoS parameters are calculated as the average values based on each simulation output trace file using AWK programming script and then presented in the form of graphs as shown in the next sub-sections.

A. *Average End-to-End Delay*

Average e2e delay is the QoS parameter often used to describe the level of service interactivity and smooth data transmission. The average e2e delays for all connections of a traffic type in a simulation, D, is calculated by summing up all of the one way connection delay, Dc, and then divided with the total number of established active connections (i.e. n parameter) during the simulation time as in (1).

$$D = \frac{\sum_{i=1}^{i=n} (D_t)_i}{n} \quad (1)$$

Figure 2, 3 and 4 show the average e2e for FTP and CBR traffics over Diffserv network using Token Bucket,

TSW3CM and SRTCM policies respectively. The graphs show the delay variation for each traffic type between 2 and 12 active connections. The time for each simulation round is 85 seconds.

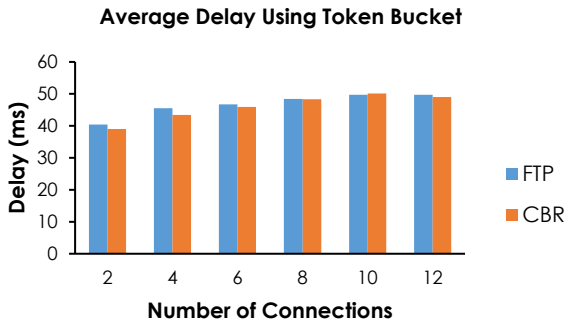

Figure 2: Average end-to-end delay using Token Bucket

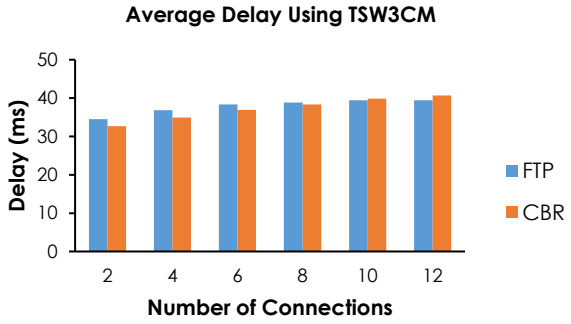

Figure 3: Average end-to-end delay using TSW3CM

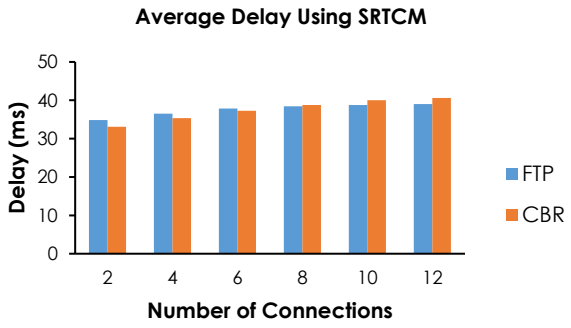

Figure 4: Average end-to-end delay using SRTCM

Figure 2 shows that the FTP traffic contributes the most delays compared to the CBR with the highest average e2e delays of 49.7009 ms when 12 active connections are created within the network. Besides that, the CBR traffic shows the lowest average e2e delay which is 39.0055 ms. Figure 3 and 4 show that the CBR traffic achieved the highest average e2e delay for 12 active connections which are about 45.6508 ms and 40.5742 ms respectively. The minimum delays achieved by CBR traffic which are 32.6654 ms and 33.1172 ms respectively.

Figure 5 shows the average accumulative delays for all policy type. The accumulative delay is calculated by averaging the one-way delays in all simulations rounds using different policy types. The TSW3CM policy provides the lowest delay for CBR traffic which is about 37.2197ms and this has made it the most suitable Diffserv policy type for delay-sensitive traffic in this network configuration

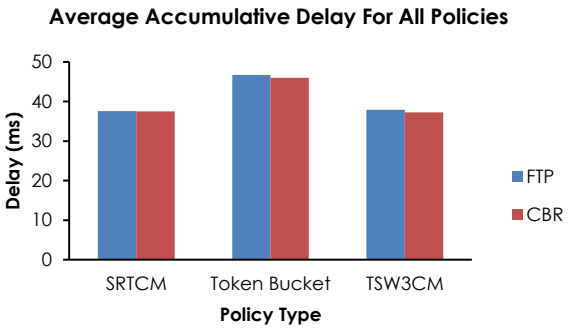

Figure 5: Average accumulative delay for all Diffserv policies

B. Average End-to-End Jitter

Jitter is the e2e one way delay variation between packets transmitted from source to destination by ignoring any lost packets [13]. Jitter causes the packets to arrive at different timing and possibility in different order. Equation (2) is the general equation used to calculate jitter per connection.

$$J(i+1) = J(i) + \left| \left(R(i+1) \quad S(i+1)\right) \quad \left(R(i) \quad S(i)\right) \right| \qquad (2)$$

where:

S(i) – Time at which packet 'I' is transmitted from the caller.

R(i) – Time at which packet 'I' is received at the receiver.

The average e2e jitter for all traffic connections in a simulation, J, is then calculated by summing up all of the one way connection jitter and then divided with the total number of established active connections (i.e. n parameter) during the simulation time as shown in (3).

$$J = \frac{\sum_{i=1}^{i=n} (J(x))_i}{n} \qquad (3)$$

The average e2e jitters shown in Figure 6, 7 and 8 are steadily increased between 2 and 8 connections. After that the values slightly decrease for 10 and 12 active connections for TSW3CM and SRTCM. The declining trend maybe because of too many packets have been dropped in the congested links as the network moves towards saturation point and the jitter counted in the simulations ignored the lost packets. The jitter variation could severely degrade the e2e performance of delay-sensitive traffic compared to the throughput-sensitive traffic. The ITU standard recommendation for jitter depends on the application type but it is better to keep the value to the minimum for better performance. Based on Figure 6, 7 and 8, the maximum jitters are produced by CBR traffic which are approximately 9.6852 ms, 7.0357 ms and 6.1882 ms respectively.

Figure 9 shows the average accumulative jitter for all policy types. The accumulative jitter is calculated by averaging the one-way jitters in all simulations rounds using different policy types. The minimum accumulative jitters for FTP and CBR traffics are 2.5944 ms and 5.2252 ms respectively. For this network scenario, the SRTCM policy shows the best jitter performance for CBR traffic while TSW3CM policy shows the best performance for FTP traffic.

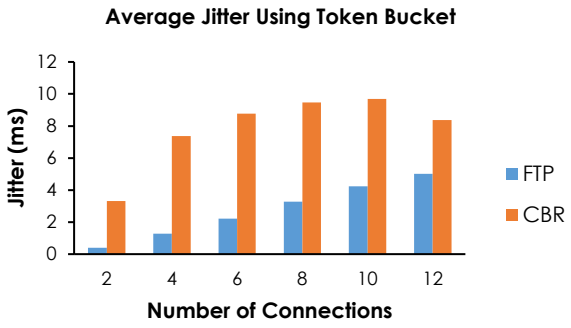

Figure 6: Average end-to-end jitter using Token Bucket

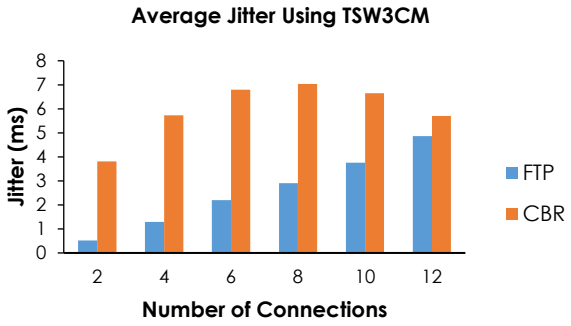

Figure 7: Average end-to-end jitter using TSW3CM

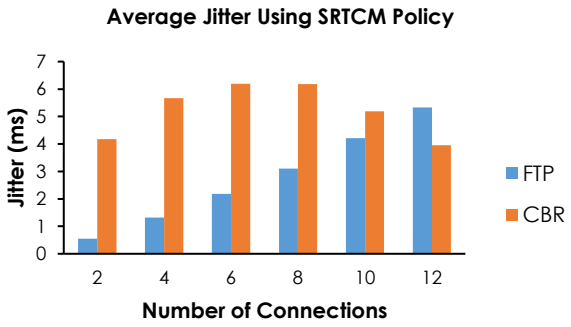

Figure 8: Average end-to-end jitter using SRTCM

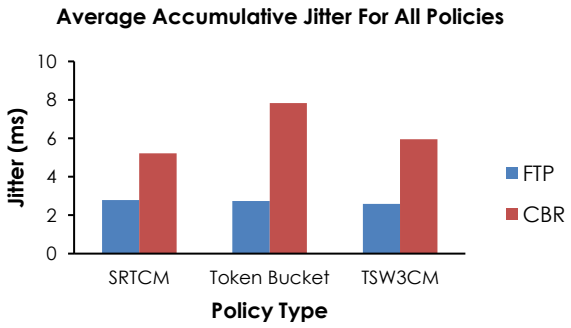

Figure 9: Average accumulative jitter for all Diffserv policies

C. *Average End-to-End Loss Ratio*

Packet loss ratio is the ratio of total packet loss over packet sent from the traffic source. The packet loss ratio for a traffic connection in a simulation is measured as in (4).

$$L(i) = \frac{\sum_{j=1}^{j=y} (P_l)_j}{\sum_{k=1}^{k=z} (P_s)_k} \quad (4)$$

where $\sum P_l$ is the total packets loss and $\sum P_s$ is the total sending packets from the source during a connection session. By considering all generated connections (i.e. n parameter) in a simulation run time, the average e2e packet loss ratio is calculated as in (5).

$$L = \frac{\sum_{i=1}^{i=n} L(i)}{n} \quad (5)$$

Figure 10, 11 and 12 show the average e2e packet loss ratio using Token Bucket, TSW3CM and SRTCM policies respectively. The average e2e packet loss ratio is proportional to the increment of average generated traffic connections in the netwok system. The lower the loss ratio, the better would be the e2e service quality. The maximum average e2e loss ratio for FTP traffic using Token Bucket, TSW3CM and SRTCM policies are 11.51%, 12.79% and 12.55% while for CBR traffic the values are 0.38%, 11.88% and 0.08% respectively. The maximum loss ratio occurs when the maximum number of active connections generated between source and destination. In addition, the minimum average e2e loss ratio for FTP traffic using Token Bucket, TSW3CM and SRTCM policies are 0.35%, 1.41% and 1.46% respectively while for CBR traffic the values are 0%, 1.88% and 0% respectively during 2 active connections.

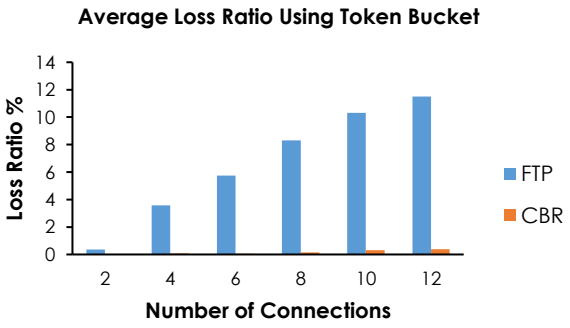

Figure 10: Average end-to-end loss ratio using Token Bucket

Figure 13 shows the average accumulative e2e loss ratio for all Diffserv policies. The values are calculated by averaging the loss ratio values in all simulation rounds for each Diffserv policy configuration. The minimum average accumulative loss ratio for FTP traffic is 6.63% using Token Bucket policy while the value for CBR traffic is 0.03% using SRTCM policy.

The packet loss occurs when the queue buffer on a link element becomes overflow as the results of network congestion. In a Diffserv network, the loss is also due to the early packet dropped by the Diffserv buffer either at the edge or core node if the transmitted rate exceeds either one of the CIR, PIR, EBS and CBS parameters. The packet loss is a critical factor for the throughput-sensitive traffic like FTP and the value must be kept at minimum. For this network

scenario, the best Diffserv policy for throughput-sensitive traffic is Token Bucket.

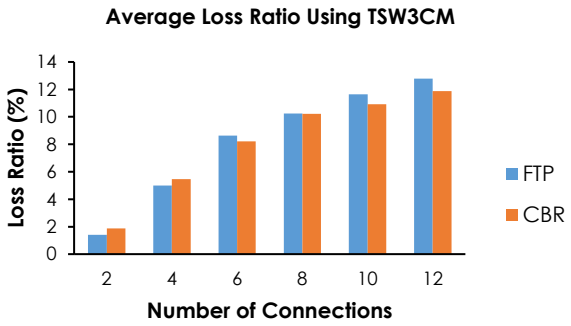

Figure 11: Average end-to-end loss ratio using TSW3CM

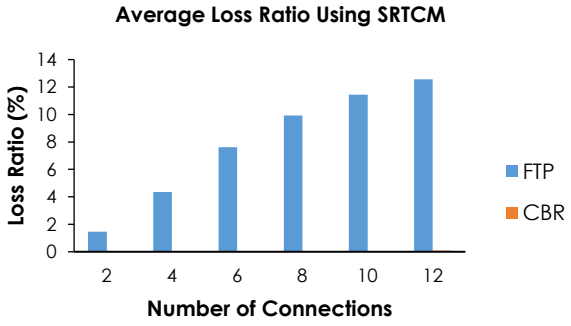

Figure 12: Average end-to-end loss ratio using SRTCM

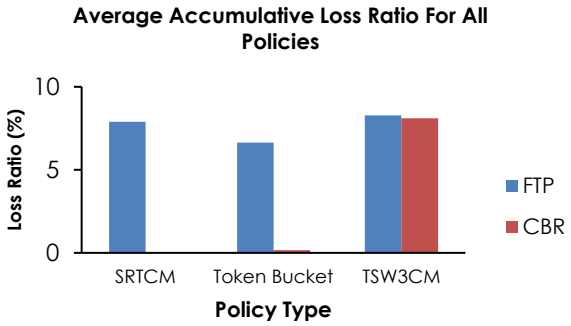

Figure 13: Average accumulative loss ratio for all Diffserv policies

D. Average End-to-End Throughput

The average e2e throughput summarize the previous QoS performance parameters as it measures the rate of successfully received packets at the receiver. The throughput is the critical parameter for throughput-sensitive traffic like FTP. The average e2e throughput in bit per second (bps) for all traffic connections (i.e. n parameter) in a simulation is calculated as in (6).

$$T = \frac{\sum_{i=1}^{i=n} \frac{P_b \times 8}{t_r \quad t_c}_i}{n} \qquad (6)$$

where P_b is the total received packets at the receiver in bytes, tr is the packet received time and tc is the packet sending time from the source of a traffic connection.

Figure 14, 15 and 16 show the average e2e throughput for traffic connections between 2 and 12. The average e2e throughput patterns are inversely proportional to the increment of average active connections. The higher the throughput the better would be the QoS. The average throughput for FTP traffic varies between 2133.43 kbps and 273.47 kbps using Token Bucket policy, 1831.14 kbps and 259.06 kbps using TSW3CM and also between 1876.28 kbps and 258.35 kbps using SRTCM policy.

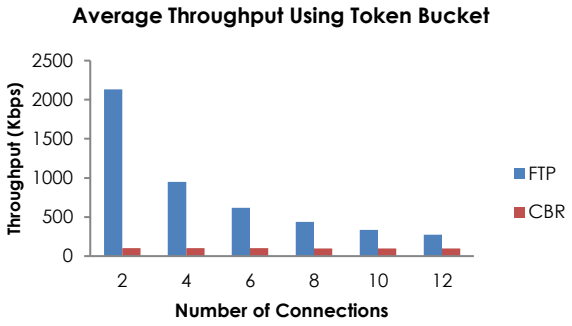

Figure 14: Average end-to-end throughput using Token Bucket

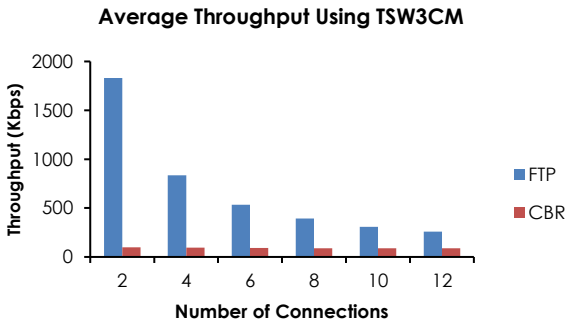

Figure 15: Average end-to-end throughput using TSW3CM

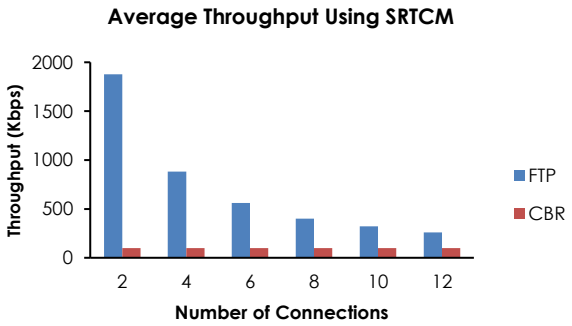

Figure 16: Average end-to-end throughput using SRTCM

Figure 17 shows the average accumulative e2e throughput for all traffic types. The average accumulative throughput is calculated by averaging the one-way throughputs in all simulation rounds using different policy types. The best average accumulative throughput that FTP could achieve is 791.46 kbps while the CBR achieved 100 kbps using SRTCM. There are 2 main factors that affecting the throughput which are the delay and packet losses. Unlike CBR, the FTP traffic uses reliable data transmission which waits for acknowledgement prior to sending the subsequent packets. This causes additional round-trip delays and retransmission if any packet loss occurs from source to destination. For this network system configuration, the Token bucket policy shown the best throughput performance for

FTP traffic while SRTCM policy shows the best performance for CBR traffic.

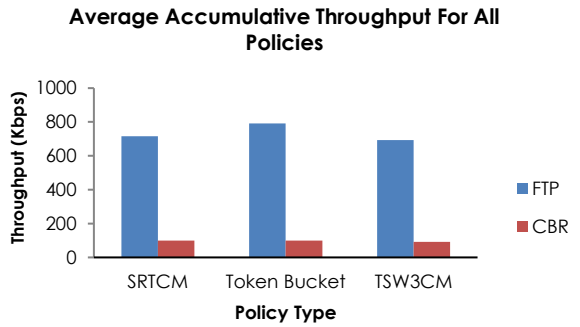

Figure 17: Average accumulative throughput for all Diffserv policies

IV. Conclusions and Future Works

This paper shows the simulation studies of e2e QoS performance for FTP and CBR traffics over a Diffserv network system. The studies compare the e2e QoS performance of both traffics using 3 different Diffserv policies which are the Token Bucket, TSW3CM and SRTCM. The network scenario is designed with the standard parameters without specific QoS improvement modification in order to make the general performance comparison.

The FTP traffic achieves the best overall delay performance using SRTCM policy which is approximately 37.5811 ms and the best jitter performance using TSW3CM which is approximately 2.5944 ms. In addition, the Token Bucket policy shows the best loss ratio and Throughput performances which are approximately 6.63% and 791.4592 kbps respectively.

Meanwhile, the CBR traffic achieved the best overall delay performance using TSW3CM which is approximately 37.2197 ms. In addition, the SRTCM policy shows the best jitter, loss ratio and throughput performances which are approximately 5.2252 ms, 0.03% and 100kbps respectively.

This paper does not dictate the best Diffserv policy for all types of applications but rather to provide a guideline for future researchers to design better Diffserv network system configurations. The future works aims at designing the combination of QoS aware routing, scheduling and Diffserv queuing schemes that can adaptively maintain each traffic type QoS requirements at optimum level.

Acknowledgment

The authors would like to thank the Ministry of Education Malaysia for the generous financial support under Research Acculturation Grant Scheme (R054).

References

[1] Audah, L., Sun, Z. and Cruickshank, H. 2010. End-to-End QoS of IP-Diffserv Network over LEO Satellite Constellation. *Proceedings of 2nd International ICST Conference on Personal Satellite Services (PSAT)*. Rome, Italy. 99-113.

[2] Nichols, K., Blake, S., Baker, F., Black, D. 1998. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. *IETF Network Working Group*, RFC 2474.

[3] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W. D. 1998. Architecture for Differentiated Services. *IETF Network Working Group*, RFC 2475.

[4] Kaur, S. and Singh, G. 2015. Implementation of Differential Services Based on Priority, Token Bucket, and Round Robin Algorithms. *International Journal of Computer Science and Mobile Computing*. 4(5): 810-818.

[5] Marzuki, A.N.A., Othman, M., Sembiyev, O. and Selamat, H. 2014. Enhancement of Adaptive Factor Provision Aware Proportional Three Color Marker in Diffserv Network. *Proceedings of International Conference of Industrial Technologies and Engineering (ICITE)*. Kazakhstan. 86-92.

[6] Katsikogiannis, G., Mitropoulos, S. and Douligeris, C. 2013. Policy-Based QoS Management for SLA-Driven Adaptive Routing. *Journal of Communications and Networks*. 15(3): 301-311.

[7] El-Atawy, A. and Samak, T. 2012. End-to-End Verification of QoS Policies. *IEEE Network Operations and Management Symposium (NOMS)*. 426-434.

[8] Fall, K. and Varadhan, V. 2011. The ns Manual (Formerly ns Notes and Documentation). The VINT Project (Researchers Collaboration). California: University California Berkeley.

[9] Altman, E. and Jimenez, T. 2003. NS Simulator for Beginners. Lecture Notes. Merida: University de Los Andes.

[10] Fang, W., Seddigh, N. and Nandy, B. 2000. A Time Sliding Window Three Color Marker (TSW3CM). *IETF Network Working Group*, RFC 2859.

[11] Stallings, W. 2007. Data and Computer Communications. 8th edition. Upper Saddle River, New Jersey: *Pearson Prentice Hall*.

[12] Heinanen, J., Finland, T. and Guerin, R. 1999. A Single Rate Three Color Marker (SRTCM). *IETF Network Working Group*, RFC 2697.

[13] Szigeti, T. and Hattingh, C. 2014. QoS Design Overview. End-to-End QoS Network Design: Quality of Service in LANs, WANs and VPNs. 1st edition. Indianapolis: Cisco Press.