End-to-End QoS Evaluation of IP-Diffserv Network over LEO Satellite Constellation

Lukman Audah, Zhili Sun, and Haitham Cruickshank

Centre for Communication System Research (CCSR)
University of Surrey, Guildford, UK
{l.audah,z.sun,h.cruickshank}@surrey.ac.uk

Abstract. In this paper, we present an end-to-end QoS simulation studies on internetworking of remote LAN and long range communications over LEO-Iridium satellites constellation taking SuperJARING network in Malaysia as an example. A macro level network simulation scenario based on actual network topology in Malaysia is implemented as Diffserv network model using the Network Simulator-2 (NS-2). Web traffic (HTTP) is used as the internet traffic models in the simulation analysis. All simulations are carried out in error-free and link-loss environment. In error-free simulations, the accumulative network traffic loads are varied from 20%, 50% and 80% while in link-loss environment simulations only 20% traffic load is used with bit error rate (BER) varied from $1x10^{-5}$, $1x10^{-4}$ and $2x10^{-4}$. The results show clearly that QoS can be achieved with IP Diffserv over satellites constellation like Iridium.

Keywords: End-to-end QoS, IP over satellites, Differentiated Services (Diffserv).

1 Introduction

The current Internet architecture operates mostly based on connectionless Internet Protocol (IP) system that provides best effort services. This means the IP routers treat all packet streams equally without given any preferential treatment to the higher priority traffic streams. Furthermore, the IP routers route packets based on shortest path first (SPF) algorithm without regard to the overall link utilization. Consequently, the all shortest paths links become congested and over-utilized capacity while the other paths with slightly longer distance become under-utilized in link capacity. These whole things eventually cause poor Quality of Service (QoS) to the entire packet transmission services.

The IETF has proposed Differentiate Services (Diffserv) as a better solution to provide QoS guarantees in IP networks. Compared to its predecessor like the Integrated Services (Intserv) which provides services based on per-microflow state, Diffserv outsmarts Intserv in providing better end-to-end QoS and preferential treatment. Diffserv discriminates different traffic flows which have same commonality to finite aggregate of classes and provides a more scalable solution for QoS in IP networks by simplifying the complexity functions such as traffic classification and traffic conditioning within the edge routers [1][2].

However, Diffserv alone is not the complete solution without adopting Traffic Engineering (TE) to overcome the link congestion and inefficiency of network resource distribution. Traffic Engineering might enhance the end-to-end QoS performance and resource utilization in any IP packet network system. It is important to achieve the end-to-end QoS target because the Internet user's perception of service quality is based on end-to-end network performance.

Previous related studies on end-to-end Internet QoS of IP-Diffserv [3][4][5][6][7] only analyzed a micro scale of wired/wireless terrestrial network topology. None of them have analyzed the global scale of Internet data transmissions over both terrestrial and satellite and also adopting actual network topology in the simulation scenario. We believe that it is important to analyze end-to-end QoS based on the actual network topology of both terrestrial and satellite networks because this reflects the current and future Internet data transmission. Furthermore, simulation measurement and evaluation of end-to-end QoS parameters using common Internet traffic (HTTP, FTP, VoIP) for multiple client-server communications might give better foresight on the future network design with QoS guarantees for many users.

We aim to evaluate the network performance and build a base-work for future generation Internet over satellite to study the outcome of IP-Diffserv and TE in the end-to-end QoS performance. This paper presents the end-to-end QoS simulation studies from a macro level perspective which involves both terrestrial and inter-satellites communications. The end-to-end QoS parameters include packet delivery ratio, packet dropped distribution, average end-to-end packet delay and average session throughput using Hyper Text Transfer Protocol (HTTP) application on long range communications over LEO-Iridium satellites constellation between a remote local area network (LAN) and Diffserv network. A macro level network scenario that imitates the actual geographical topology in Malaysia is proposed as an example in implementing the IP-Diffserv and TE model. In addition, the simulation analysis also emulates the common speed of Internet services in Malaysia which is 1Mb/s. Simulations are done in error-free and link-loss environments. In error-free simulations, the accumulative background traffic loads are varied from 20%, 50% and 80% of the Diffserv core-link capacity while in link-loss environment simulations only 20% background traffic load is used with bit-error-rate (BER) varied from $1x10^{-5}$, $1x10^{-4}$ and $2x10^{-4}$. All simulations and analysis of the above mentioned network model are done using the Network Simulator-2 (NS-2- version 2.33).

2 Simulation Framework and Configuration

This section describes in detail the parameters used for the IP-Diffserv and the network system of both terrestrial and LEO-Iridium satellites simulation configuration. In addition, measurement methodologies for QoS parameters like packet delivery ratio (PDR), total packets dropped distribution, average end-to-end packet delay and average session throughput are also explained in this section.

2.1 Satellite Networking Simulation in NS-2

Simulation of satellite networks that follows the exact technical parameters often requires a detailed modeling of radio frequency characteristics (interference, fading), protocol interactions (e.g. interaction of residual burst errors on link with error checking codes), and second-order orbital effects (precession, gravitational anomalies, etc.). However, in order to study the fundamental characteristics of satellite networks from a networking perspective, some features might be omitted. As an example, simulation analysis of TCP performance over satellite has little effect with detailed propagation channel model which could be characterized to first order by the overall packet loss probability [8][9]. In this paper, LEO-Iridium satellites constellation are used in order to create a framework to study the QoS effects of transport, routing and handover protocol in end-to-end data transmissions. The following are the parameters description and Table- 1 shows all LEO-Iridium satellites constellation parameters that can be simulated in NS-2 [8].

Basic constellation definition: Define the satellite altitude, number of satellites, number of planes, and number of satellites per plane.

Orbits: Define the orbit inclination ranging from 0 to 180 degrees. Inclination above 90 degrees corresponds to retrograde orbits. However, orbit eccentricity and nodal precession are not modeled in NS-2. In addition, inter-satellites spacing within a given plane and relative phasing between planes are set to be fixed.

Inter-satellite (ISL) links: Define the polar orbiting constellations, intraplane and interplane satellite links. Intraplane ISL correspond to the communications between satellites in the same plane which are never deactivated or handed off. In addition, Interplane ISL referring to the communications between satellites of neighboring co-rotating planes. Both ISL will be deactivated near the poles when exceeding ISL threshold because the satellite antenna unable to track these links in the Polar Regions.

Table 1. LEO-Iridium satellites parameters [8][9]

Parameter	Value
Altitude	780Km
Planes	6
Satellites per plane	11
Inclination (degree)	86.4
Interplane separation (degree)	31.6
Seam separation (degree)	22
Elevation mask (degree)	8.2
Intraplane phasing	YES
Interplane phasing	YES
ISL per satellite	4
ISL bandwidth	25 Mb/s
Uplink/downlink bandwidth	1.5 Mb/s
Cross-seam ISL	NO
ISL latitude threshold (degree)	60

Ground to Satellite (GSL) links: Define the communications between satellites and terrestrial links network. GSL are periodically handed off when the elevation angle drop below the elevation mask. In this paper, there are two GSL which locations are set in London, UK (51.53^0, -0.08^0) and Kuala Lumpur, Malaysia (3.13^0, 101.70^0).

Elevation Mask: Define the elevation angle of GSL link can be operated. When a GSL terminal that correspond to a satellite drops below the elevation mask, it will search for a new satellite above the elevation mask. Each GSL terminal will check for handoff opportunities when the timeout interval specified by the user is exceeded. Both GSL in this paper initiate handoff asynchronously.

2.2 Simulation Scenario

The proposed simulation scenario as shown in Figure 1 consists of two main components which are the terrestrial and satellite networks. The terrestrial network on the right side is designed such that it imitates the macro level of actual geographical topology in Malaysia. The way of imitation is by assigning an edge router to represent each of the 11 Cities/Counties which connects to other 6 interconnected routers (red color) that form the Diffserv core links. Each of the 11 edge router (green color) is further connected to client (orange color) and server (blue color) nodes. So, there are 11 pairs of client and server nodes, 11 Diffserv edge routers and 6 core routers which in total are 39 nodes in the terrestrial network. The Diffserv parameters and policies are assigned to those 11 edge routers and 6 core routers which control the packet streams transmission within the Diffserv network domain. The assignment of client-server pair location is done randomly with the fact that all links must be utilized by the traffic flows. The path taken by the traffic flows from each pair of client/server node is determined by the Link State Routing Protocol.

In addition, there are another 66 LEO-Iridium satellites constellation and 2 ground to satellite links (GSL) located at London and Kuala Lumpur. The satellites form a bridge from a server node located in a remote LAN in UK to a client node in Malaysia. It should be noted that a single client node generates HTTP connections from a *"cloud"* of web clients while a single server node accepts and serves HTTP connection destined for a *"cloud"* of web servers. The link bandwidth for Core Links, C1 to C7, follow the Optical Carrier (OC-192) specification which is approximately 10Gb/s while the Edge Links, E1 to E11, follow the OC -24 specification which is approximately 1.25Gb/s [10]. A reduction of a hundred folds in the link bandwidth is done in order to speed up the simulation time and to accommodate the limited capacity of computer hard disk space.

The Core Links propagation delays approximation varies according to actual distance in Malaysia geography. The propagation delay for link C1=20ms, C2=25ms, C3=25ms, C4=30ms, C5=30ms, C6=25ms and C7=25ms. The Edge Link propagation delay is fixed to 5ms while the client/server Link is 1ms. Based on Figure 1, each pair of client and its designated server node is labeled with the same name (e.g. HTTP1, HTTP2 and etc). The paths that packet streams take across the network system are determined by the Link State (LS) routing protocol. The LS governs all the Layer 3 network routing process for terrestrial network while routing process in LEO satellites network is govern by a centralize routing genie. The routing genie determines the global satellites topology, computes new data routes for all nodes and built a forwarding table on each node [8]. In addition, it computes the shortest path data route based on the current propagation delay of a link as the cost metric.

Moreover, all simulations in NS-2 involve one type of Internet application traffic which is the Hyper Text Transfer Protocol (HTTP). A client-server pair generates thousands of different HTTP request-response sessions based on 20%, 50% and 80% of traffic loads. Calculation of the traffic load is divided in two categories which are based on the terrestrial Diffserv core links capacity and the satellite links bandwidth. As an example for 20% of traffic load simulation, all 10 server nodes in terrestrial network generate an average aggregate background traffic that approximate to the 20% of the Diffserv core links capacity which is 20Mb/s while the other server node (HTTP5) located in remote LAN in UK generates an average data rate that approximate to the 20% of uplink/downlink of GSL bandwidth which is 300Kb/s. These traffic loads parameters are determined by the inter-arrival time of each new request-response session. The traffic load also corresponds to the amount of packet streams been injected to the network system. In this paper, we want to demonstrate the effect of the background traffic generated by client/server pairs (except HTTP5) to the QoS of Internet data transmission over LEO-Iridium satellites constellation.

In addition, each HTTP session involves an average of 10Kbytes of HTTP response file transfer size. The average inter-arrival time between each generated HTTP session is modeled by Exponential distribution while the average HTTP response file transfer size is modeled by Pareto distribution. A HTTP session represents a complete request-response pair that follows both Exponential and Pareto distribution between a server-client pair. NS-2 has both distribution functions built in it and could be generated using Random Number Generator (RNG). In order to set the Pareto file transfer size distribution, the average value of 10Kbytes and Pareto shape parameter of 1.5 are passed to the Pareto type of Random Variable function. It produced a series of file size distribution with an average of 10Kbytes. It should be noted that the 10Kbytes parameter is taken based on the majority of Internet file transfer size as measured in the previous studies [11][12]. On the other hand, an average of HTTP session inter-arrival time parameter is passed to the Exponential type of Random Variable.

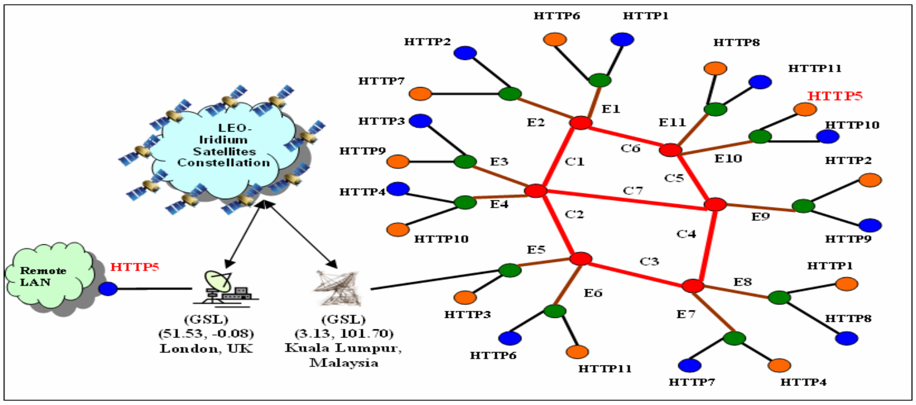
LEO-Iridium Satellites Constellation
Remote LAN
HTTP5
(GSL) (51.53, -0.08) London, UK
(GSL) (3.13, 101.70) Kuala Lumpur, Malaysia
HTTP1
HTTP2
HTTP3
HTTP4
HTTP6
HTTP7
HTTP8
HTTP9
HTTP10
HTTP11
E1
E2
E3
E4
E5
E6
E7
E8
E9
E10
E11
C1
C2
C3
C4
C5
C6
C7

Fig. 1. Simulation scenario in Network Simulator 2 (NS-2)

As mentioned previously, the two types of traffic loads are generated according to the inter-arrival time of new HTTP sessions which varies from 20%, 50% and 80%. Therefore, there are two HTTP session inter-arrival variables used in all simulations which one is according to Diffserv core links capacity (background traffic) and the other one according to uplink/downlink GSL bandwidth (traffic from HTTP5). As an example for 20% of traffic load simulation, the average inter-arrival time, *i*, measured in second is calculated using the following formula:

$$i(s) = \frac{(N)\times(Fs)\times 8}{Bw} \tag{1}$$

The *N* parameter is the number of nodes involve in generating the traffic flows while the *Fs* parameter is the average HTTP Response file size (10400 bytes) sent by the server nodes. An average of one TCP segment size is 1040 bytes of which the 1000 bytes is the Data and 40 bytes is the TCP header. An average of a file transfer is assumed to contain 10 Kbytes of Data which in total including header is approximately $1.04 \times 10^4 \times 8$bits . The *Bw* parameter is the link bandwidth measured in b/s. In this paper, we define *Bw* in two different values which one of them (100 Mb/s) is used to calculate the inter-arrival time of the background traffic generated by 10 pairs of client/server nodes (except HTTP5) and the other one (1.5 Mb/s) is for the main traffic flows across LEO-Iridium satellites constellation. Table 2 lists the average inter-arrival time of HTTP session parameters for every pair of client/server (HTTP1-HTTP11) according to 20%, 50% and 80% of traffic loads.

Moreover, all link-loss environment simulations in this paper used one-way random link loss error model. Link loss error model is configured on all Diffserv Edge Links with bit error rate (BER) $1\text{x}10^{-5}$, $1\text{x}10^{-4}$ and $2\text{x}10^{-4}$. The loss module is placed right after link's queue element and before the link's delay element. This means a packet will be marked as 'error' and dropped as soon as it enters the Diffserv edge link. The error model follows uniform distribution with minimum and maximum value of 0 and 1 respectively.

Table 2. Average inter-arrival time of HTTP sessions according to 20%, 50% and 80% of traffic loads

Traffic Load	Average Inter-arrival Time of New HTTP Session (second)	
	Main Traffic (HTTP5)	Background Traffic (except HTTP5)
20%	0.27733	0.04160
50%	0.11093	0.01664
80%	0.06933	0.01040

2.2.1 Flow Path and Propagation Delay Estimation

Table 3 shows the paths taken by the HTTP5 flow from a server node in the remote LAN to a client node in the Diffserv network (based on Figure 1), and its estimated propagation delay. The estimated propagation delay might vary based on the path variation taken in the LEO-Iridium satellites network. It was obtained without taking into account the queuing delay at each link. The value of 157.162ms is calculated based on the paths taken by HTTP5 flows in the early data transmission as stated in the NS-2 satellite output trace file. Based on the output trace file, the propagation paths in LEO-Iridium satellites network stated in term of (latitude, longitude) locations are $GSL_{(UK)}$ (51.53°, -0.08°), node(2) (65.21°,7.83°), node(1) (32.67°,2.31°), node(12) (48.97°, 35.75°), node(23) (32.66°, 65.51°), node(34) (48.97°, 98.94°), node(33) (16.33°,95.86°) and $GSL_{(KL)}$ (3.13°, 101.70°). The one way link propagation delay ($t_{sat\text{-}prop}$) in LEO-Iridium satellites network is calculated using some trigonometry formulas as discussed in [13], without taking into account the queuing delay at each satellite links. The value is equal to the summation of propagation delay from earth terminal in UK to the current nearest satellite above it (t_{uplink}), propagation delay within satellites network ($t_{inter\text{-}sat}$) and propagation delay from satellite to earth terminal in KL, ($t_{downlink}$). Following are the formulas used to calculate the estimated $t_{sat\text{-}prop}$ [13]:

$$t_{sat-prop} = t_{uplink} + t_{inter\text{-}sat} + t_{downlink} \tag{2}$$

$$t_{uplink} = \frac{d_{ts}}{c} \tag{3}$$

$$d_{ts} = \sqrt{(x_{sat} - x_{term(UK)})^2 + (y_{sat} - y_{term(UK)})^2 + (z_{sat} - z_{term(UK)})^2} \tag{4}$$

$$t_{inter\text{-}sat} = \frac{d_{is}}{c} \tag{5}$$

$$d_{is} = \sqrt{(x_{sat(i+1)} - x_{sat(i)})^2 + (y_{sat(i+1)} - y_{sat(i)})^2 + (z_{sat(i+1)} - z_{sat(i)})^2} \tag{6}$$

$$t_{downlink} = \frac{d_{st}}{c} \tag{7}$$

$$d_{st} = \sqrt{(x_{term(KL)} - x_{sat})^2 + (y_{term(KL)} - y_{sat})^2 + (z_{term(KL)} - z_{sat})^2} \tag{8}$$

Where:

d_{ts}	= Distance from earth terminal in UK to satellite
d_{is}	= Distance between satellite$_{(i)}$ and satellite$_{(i+1)}$
d_{st}	= Distance from satellite to earth terminal in KL
x_{sat}	$= (R+h)\cos\theta_{sat}\cos\phi_{sat}$, $y_{sat} = (R+h)\cos\theta_{sat}\sin\phi_{sat}$, $z_{sat} = (R+h)\sin\theta_{sat}$
$x_{term()}$	$= R\cos\theta_{term()}\cos\theta_{term()}$, $y_{term()} = R\cos\theta_{term()}\sin\theta_{term()}$, $z_{term()} = R\sin\theta_{term()}$
θ	= latitude, ϕ = longitude
R	= 6378.137 Km (earth radius), c = 299792 Km/s (light speed), h = 780 Km

Table 3. Flow paths from server node to client node and its estimated propagation delay

HTTP Flow	Paths Taken	Estimated Propagation Delay
HTTP5	$GSL_{(UK)}$→ LEO Satellites→ $GSL_{(KL)}$→ E5→ C2→ C1→ C6→ E10	≈ 157.162ms

In order to calculate the link propagation delay from $GSL_{(UK)}$ to the nearest satellite (node(2)) above it, both coordinate locations of $GSL_{(UK)}$ and node(2) are inserted in equation (4) and then equation (3) which yield t_{uplink} = 6.173ms. The inter-satellite links propagation delays are calculated using equation (6) and equation (5) for every pair of satellite (sat_{i+1} and sat_i), using node(2), node(1), node(12), node(23), node(34) and node(33) coordinates which then yield $t_{inter\text{-}sat}$ = 13.448ms + 12.253ms + 11.358ms + 12.253ms + 13.458ms = 62.770ms. In addition, the downlink propagation delay is calculated using equation (8) and equation (7) with $GSL_{(KL)}$ and node(33) coordinates which then yield $t_{downlink}$ = 6.219ms. Therefore, the total propagation delay in LEO-Iridium satellites network is $t_{sat\text{-}prop}$ = 6.173ms+ 62.770ms + 6.219ms = 75.162ms. Meanwhile, the propagation delay across E5, C2, C1, C6 and E10 nodes within Diffserv network domain from $GSL_{(KL)}$ to the designated HTTP5 client node is 82ms, as previously described in section 2.2. Finally, the end-to-end link propagation delay for HTTP5 is $t_{tot\text{-}prop} = t_{sat\text{-}prop} + t_{terrestrial\text{-}prop}$ = 75.162ms + 82ms = 157.162ms.

2.2.2 Link Loss

All link-loss environment simulations in this paper used one-way random link loss error model. Link loss error model is configured on all Diffserv edge links with bit error rate (BER) $1\text{x}10^{-5}$, $1\text{x}10^{-4}$ and $2\text{x}10^{-4}$. The loss module is placed right after link's queue element and before the link's delay element. This means a packet will be marked as 'error' and dropped as soon as it enters the edge link. The error model follows uniform distribution with minimum and maximum value of 0 and 1 respectively.

3 Result and Discussion

The following are the results and discussions based on HTTP simulations in NS-2. The results are divided into error-free (traffic loads of 20%, 50% and 80%) and with link-loss condition (BER = $1\text{x}10^{-5}$, $1\text{x}10^{-4}$ and $2\text{x}10^{-4}$) for traffic flows over LEO-Iridium satellites network. All QoS parameters results obtained from the NS-2 simulations are calculated as averages for all HTTP sessions generated by HTTP5 client/server pair. Simulation time is set to 300 second because there are more than 2GB of the output trace file produced for each traffic load category.

3.1 Packet Delivery Ratio (PDR)

Figure 2 shows the Packet Delivery Ratio (PDR) for HTTP5 Request and Response packets transmission over LEO-Iridium satellites network. The PDR is calculated as the ratio between the Received Request/Response packets and Sent Request/Response packets type. It measures the percentage of successful end-to-end data transmission. Although all packets are guaranteed to be delivered from source to destination by the TCP, not all packets received are the original packets sent by the source. Some of the packets are lost and need to be retransmitted. Therefore, the PDR shows the ratio between total sent packets including retransmission packets and total received packets.

The PDR is inversely proportional to the increment of traffic load, the lower the traffic load the higher would be the PDR. The PDR is higher in lower traffic load because the links still could sustain the traffic burst. Fewer packets are lost in lower traffic load.

However, the PDR is lower in higher traffic load because the links become saturated with traffic burst which eventually cause many packets being dropped. Moreover, compared to the two types of PDR in HTTP5 flows, most of the Response PDR is a lower than the Request PDR. This mainly because the average total size of response packets (10Kbytes) is much higher than the response packets (550bytes) and many of them will be dropped either by Diffserv elements or due to the narrowband in satellite links. Based on Figure 2, the PDR values are much above 96% in all traffic load variation. This is due to the flow control by TCP in order to provide a reliable data transmission.

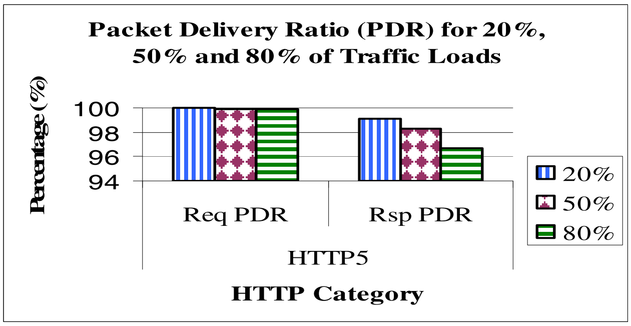

Fig. 2. Packet Delivery Ratio (PDR) for 20%, 50% and 80% of traffic loads

Figure 3 shows the PDR of HTTP5 flows for 20% traffic load with BER from $1x10^{-5}$, $1x10^{-4}$ and $2x10^{-4}$. The PDR of HTTP5 flows are inversely proportional to the increment of BER. The higher the BER, the lower would be the number of successful transmitted packets. Based on the graph, the PDR could be considered higher which is above 80% in the worst case of BER equal to $2x10^{-4}$. The main reason is because the HTTP5 flows operate in low bandwidth of 20% traffic load in which the links could still sustained the traffic burst. Furthermore, the results shows that link-loss in Diffserv network domain did not give significant effect on short HTTP sessions when operate in lower traffic load.

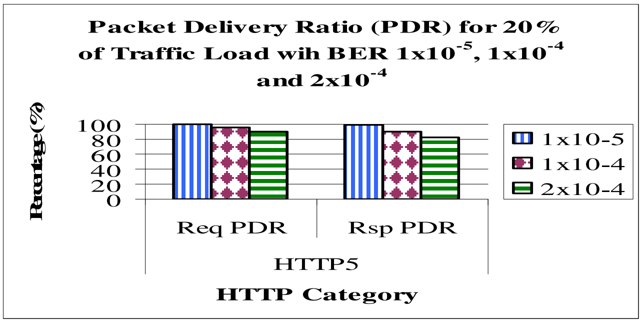

Fig. 3. Packet Delivery Ratio (PDR) for 20% of traffic load with Bit Error Rate (BER)

3.2 Total Packets Dropped Distribution

Figure 4 shows the distribution of total dropped HTTP Request, Response, SYN/ACK and FIN/ACK type of packets during 300 second of simulation time. It should be noted that the SYN/ACK and FIN/ACK are small packets of 40 bytes size which sent by the HTTP client and server upon connection establishment or connection tear down. The SYN/ACK means SYN or ACK packet type sent by the client. In addition, the FIN/ACK means FIN or ACK packet type sent by the server. The total packets dropped include the packets dropped due to buffer overflow and also Diffserv RED buffer early packets dropped. As traffic loads increase from 20% to 80%, the numbers of packets dropped are drastically increased. This is because greedy flows are severely punished by the Diffserv RED buffers. The increment of traffic loads will cause the current average buffer size to grow larger as many packets need to queue before being transmitted. Diffserv marks the packet flows that have accumulative sending rate more than the 1 Mb/s and dropped those packets probabilistically when the current average RED buffer size exceeds the minimum threshold. All packets are then dropped when the buffer size exceeds the maximum threshold. Based on the graph, the HTTP Response packets are dropped much more than the SYN/ACK and FIN/ACK packets. This is because the HTTP Response packets are larger (average size of 10 Kbytes) than the SYN/ACK and FIN/ACK packets (40 bytes each). Larger packets will quickly fill the queue buffer which will then trigger the Diffserv RED monitoring element that estimates the current queue size. Besides that, the total packets dropped not only due to the Diffserv RED buffers but also due to the narrowband links in the satellites network.

Figure 5 shows the total packets dropped distribution in link-loss simulation environment for 20% of traffic load with BER 1x10^{-5}, 1x10^{-4} and 2x10^{-4}. The number of packets dropped increase proportionally with the increment of BER. Compared to Figure 4, flows in lower traffic load did not much penalized by Diffserv and the packets dropped mainly due to the link-loss error model implemented on the Diffserv network boundary.

Based on Figure 4, the total HTTP packets dropped in 20%, 50% and 80% of traffic load are 102 packets, 545 packets and 1877 packets respectively. Meanwhile, the total HTTP packets dropped for 1x10^{-5}, 1x10^{-4} and 2x10^{-4} of BER are 104 packets, 1284 packets and 2734 packets respectively. From these values, we could see that the

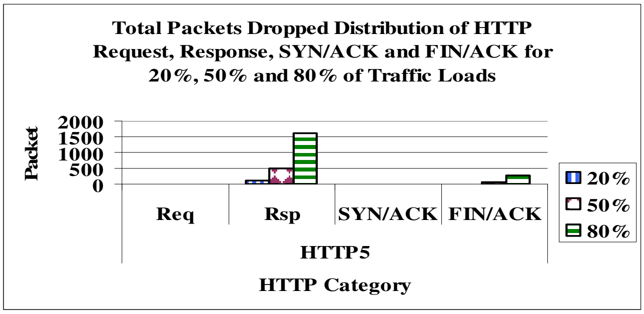

Fig. 4. Total packets dropped distribution for 20%, 50% and 80% of traffic loads

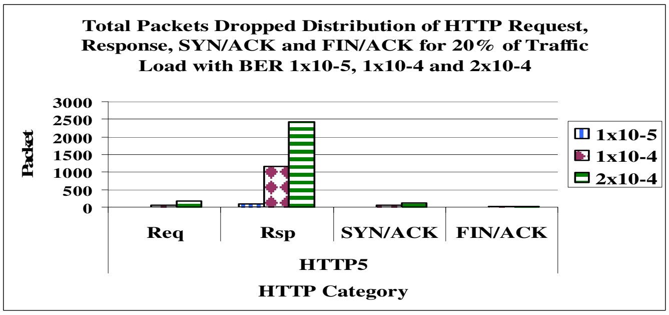

Fig. 5. Total packets dropped distribution for 20% of traffic load with Bit Error Rate (BER)

increment of BER in lower traffic load (20%) has cause larger number of packets to be dropped compared to the number of packet dropped in higher traffic load.

3.3 Average End-to-End Packet Delay

Unlike in the previous sections that describe QoS parameter based on per packet basis, this section and the following section discuss the QoS parameters based on average number of completed HTTP sessions,. The average end-to-end packet delay involves three main factors which are the propagation delay, queuing delay, and delay due to other traffic condition (e.g. link-loss with bit-error-rate). Table 3 already states the one-way propagation delay over LEO-Iridium satellites network from server node to its correspondence client node without taking into account the queuing delay. The propagation delay in satellites network may vary due to the handover process and various paths taken by the packets streams. For all error-free simulations (20%, 50% and 80% of traffic loads) in this paper, the additional factor due to other traffic condition could be neglected and the end-to-end delay only involves propagation delay and queuing delay. We estimate the average end-to-end delay (D_{avg}) using the following equation based on summation of session duration (t_s) per total received of HTTP request/response packets ($P_{s\text{-}tot}$) and then divided the value with the total number of completed HTTP sessions (S_{tot}) generated by HTTP5 client/server pair for the whole simulation time.

$$D_{avg} = \left[\frac{\sum \frac{t_s}{P_{s-tot}}}{S_{tot}} \right] \tag{9}$$

Based on Figure 6, as the traffic loads increase from 20% to 80%, the links become busy with traffic burst and the service time at each queue buffer become lower than the incoming traffic flows which eventually cause buffer to overflow. This has caused many packets need to be retransmitted to complete a session transfer. The session duration becomes higher in order to complete a HTTP request-response and as the result the average end-to-end delay becomes higher too in every HTTP session. In addition, many packets are dropped in higher traffic load due to Diffserv RED buffer

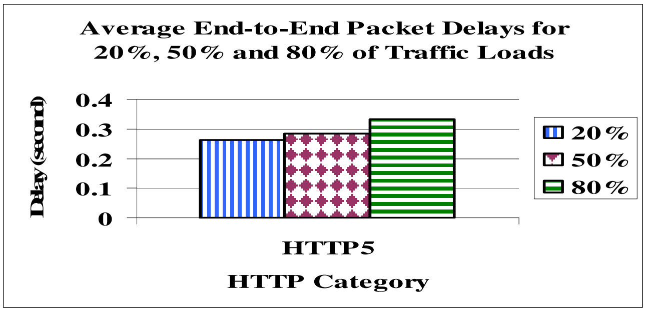

Fig. 6. Average end-to-end packet delays for 20%, 50% and 80% of traffic loads

early drop action which also cause many packets need to be retransmitted to complete a HTTP session. This Diffserv policy had severely punished greedy flows and eventually causes the increment of average end-to-end delay.

Figure 6 shows the average end-to-end packet delays (D_{avg}) for 20%, 50% and 80% of traffic loads are 0.2626 second, 0.2837 second and 0.3324 second respectively. Based on previous mentioned assumption that the delay due to other network condition (e.g. link-loss with bit-error-rate) could be neglected in all error-free simulations, we then estimate the average end-to-end queuing delay using the following formula:

$$D_q = D_{avg} - t_{tot-prop} \tag{10}$$

Therefore, the average end-to-end queuing delays for 20%, 50% and 80% of traffic loads are 105.438ms, 126.538ms and 175.238ms respectively.

Figure 7 shows the average end-to-end packet delays for HTTP5 flows in 20% of traffic load with BER of $1x10^{-5}$, $1x10^{-4}$ and $2x10^{-4}$. Based on the graph, the average end-to-end packet delay is proportionally increased with the increment of BER. The higher the BER, the longer time needed to send a packet from server node to client node or from client node to server node. This is mainly because many packets are dropped in higher BER and need to be retransmitted. Based on Figure 5, the average end-to-end delays ($D_{avg(BER)}$) with BER $1x10^{-5}$, $1x10^{-4}$ and $2x10^{-4}$ are 0.2690 second, 0.3594 second and 0.5184 respectively.

The integration of random error-model to create some network scenario variations in all link-loss simulations has caused additional packet delay apart from the propagation delay ($t_{tot-prop}$) and queuing delay (D_q). Based on the delay parameters obtained previously, we then estimate the additional average end-to-end delay (D_{add}) due to the BER variation using the following formula:

$$D_{add} = D_{avg(BER)} - D_{q(20\%)} - t_{tot-prop} \tag{11}$$

Therefore, the additional average end-to-end delays due to the network condition variation with BER of $1x10^{-5}$, $1x10^{-4}$ and $2x10^{-4}$ are 6.4 ms, 96.8 ms and 255.8 ms respectively.

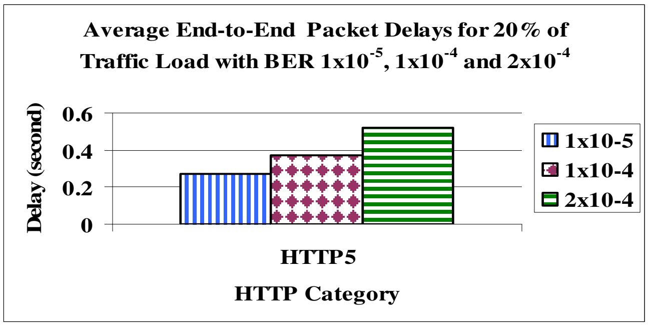

Fig. 7. Average end-to-end packet delays for 20% of traffic load with Bit Error Rate (BER)

3.4 Average HTTP Session Throughput

The instantaneous session throughput is measured in bit/s based on the amount of successfully received bit in each HTTP session divided by the session duration time. The instantaneous throughput is estimated by summing all the amount of successfully received HTTP Request and Response bits (B_{tot}) and then divided by the session duration time (t_s). The average session throughput ($T_{s\text{-}avg}$) is then estimated using equation (12) by summing all instantaneous session throughputs and divided by total number of completed HTTP session (S_{tot}). This section could be regarded as the conclusion of the previous sections because it shows the final results after taking into account the effect of all the QoS parameters mentioned previously.

Figure 8 shows the average HTTP5 session throughputs for 20%, 50% and 80% traffic loads. Indeed the variation of average session throughput is very much depending on the end-to-end delay and consequently the session duration. The longer the paths taken from source to destination, the longer would be the time needed to transmit a packet due to propagation delay and queuing delay at each node, and as the result the longer time needed to complete a HTTP session. In addition, the low PDR and high drop rate also contribute to the lower throughput in higher traffic load. The average session throughputs for 20%, 50% and 80% of traffic loads are 40.157 Kb/s, 37.75 Kb/s and 32.601 Kb/s respectively.

$$T_{s-avg} = \left[\frac{\sum \frac{B_{s-tot}}{t_s}}{S_{tot}} \right] \tag{12}$$

Figure 9 shows the average session throughput for HTTP5 flows for 20% of traffic load in link-loss environment simulations. Based on the graph, the average session throughput is inversely proportional to the increment of BER. Based on the graph, the average session throughputs with BER of $1\text{x}10^{-5}$, $1\text{x}10^{-4}$ and $2\text{x}10^{-4}$ are 39.099 Kb/s, 28.167 Kb/s and 21.188 Kb/s respectively. From those values, we found that the decrements of average session throughput are much higher in link-loss environment compared to the decrement of those values in higher traffic load. This mainly due to the high packets loss and high average session duration as the BER increased from $1\text{x}10^{-5}$ to $2\text{x}10^{-4}$.

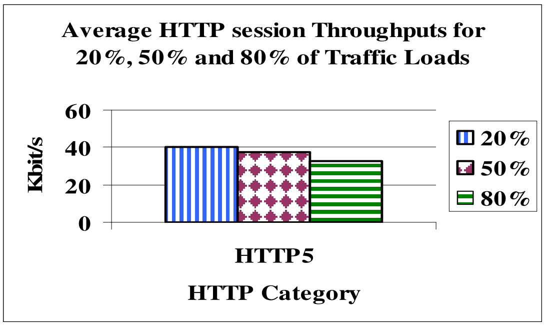

Fig. 8. Average HTTP session throughputs for 20%, 50% and 80% of traffic loads

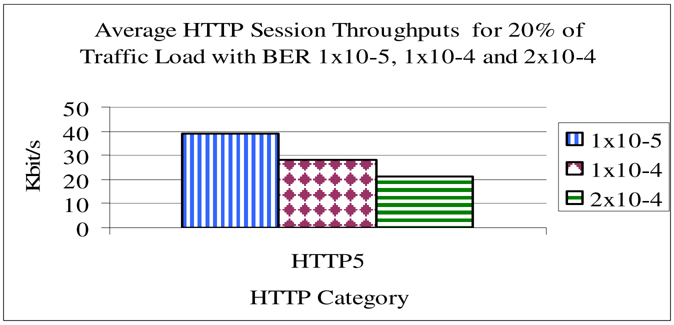

Fig. 9. Average HTTP session throughputs for 20% of traffic load with Bit Error Rate (BER)

4 Conclusion and Future Research

In this paper, we studied end-to-end QoS of IP-Diffserv over satellites network model in NS-2. The QoS simulation parameters are considered including packet delivery ratio (PDR), total packets dropped distribution, end-to-end packet delay and average session throughput. Network traffic involves many short session of web transmissions in both terrestrial and over LEO-Iridium satellites. In addition, the simulations involve error-free (20%, 50% and 80% of traffic loads) and link-loss (only 20% of traffic load with BER $1x10^{-5}$, $1x10^{-4}$ and $2x10^{-4}$) simulation environment. The results show that the end-to-end service quality is inversely proportional to the increment of both background traffic load and BER. The higher the background traffic load and BER, the lower would be the service quality. Diffserv is just the mechanism to reduce those effects and preserve the end-to-end internet service quality and network performance.

As for the future research, some improvements can be done on both satellite and the Diffserv network system in order to provide better end-to-end QoS for satellite-terrestrial communications. As for the Diffserv network system, we suggest that it should be combined with traffic engineering mechanism like multi protocol label switching (MPLS) for fast switching and optimum packet routing. This will greatly reduce the buffering process delay at all Diffserv routers. However, this can be further improved by adding the adaptive admission control at the Diffserv ingress/egress routers for service differentiation and traffic flow management. The adaptive admission control mechanism will be based on periodic rate measurement information at every Diffserv routers which will then passed to the ingress/egress routers for further actions. Based on that information, the system must be able to provide alternative routing paths and efficient resource reservation for the traffic flows in case of network environment variation (e.g. link-loss, traffic load and etc). Meanwhile, the satellites network must be enhanced in the software part with on-board traffic flows processing and efficient buffer management.

References

1. Nichols, K., Blake, S., Baker, F., Black, D.: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. IETF Network Working Group, RFC 2474 (1998)
2. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.D.: An Architecture for Differentiated Services. IETF Network Working Group, RFC 2475 (1998)
3. Zhang, G., Mouftah, H.T.: End-to-End QoS guarantees over Diffserv Networks. In: 6th IEEE Symposium on Computer and Communications (2001)
4. Yang, J., Ye, J., Papavassiliou, S.: Enhancing End-to-End QoS Granularity in Diffserv Networks via Service Vector and Explicit Endpoint Admission Control. Journal of IEEE Communications Proceedings 151, 77–81 (2004)
5. Yang, J., Ye, J., Papavassiliou, S., Ansari, N.: Decoupling End-to-End QoS Provisioning at Routers in the Diffserv Network Model. In: IEEE Global Telecommunications Conference, GLOBECOM (2004)
6. Zhou, L., Pung, H.K., Ngoh, L.H.: A Cross-Domain Framework for Coordinated End-to-End QoS Adaptation. In: 33rd IEEE Conference on Local Computer Networks, LCN (2008)
7. Myounghwan, L., Copeland, J.A.: An Adaptive End-to-End Delay Assurance Algorithm with Diffserv Architecture in IEEE 902.11e/IEEE 802.16 Hybrid Mesh/Relay Networks. In: Proceeding of 18th International Conference on Computer Communications and Networks, ICCCN (2009)
8. Fall, K., Varadhan, K.: The NS Manual. University California, Berkeley (2008)
9. Pattan, B.: Satellite-Based Cellular Communication. McGraw-Hill, New York (1997)
10. Optical Carrier, `http://en.wikipedia.org/wiki/Optical_carrier`
11. Crovella, M.E., Bestavros, A.: Self-Similarity in World Wide Web traffic: Evidence and possible cause. IEEE/ACM Transaction on Networking 5, 835–846 (1996)
12. Sikdar, B., Kalyanaraman, S., Vastola, K.S.: An Integrated Model for the Latency and Steady-State Throughput of TCP Connections. Performance Evaluation 46, 139–154 (2001)
13. Makki, S., Pissinou, N., Daroux, P.: A New Routing Algorithm for Low Earth Orbit Satellite Networks. In: 10th International Conference on Computer Communications and Networks Proceedings (2001)