REVISTA DO DETUA, VOL. 4, Nº 9, JUNHO 2008

QoS in 4G scenarios using NSIS protocol

Fábio Ferreira, Susana Sargento, Rui L. Aguiar

Abstract - **This paper presents quality of service mechanisms, based on the NSIS (Next Steps In Signaling) protocol. For that, it was implemented a testbed with NSIS running through IPv6. The network's QoS was measured by the number of drops, for each flow (either QoS or Best Effort traffic), occurred in routers. The first experiment shows the captured packets during a node association and request for QoS, namely a GIST session and a QoS NSLP session. The second experiment illustrates a more realistic scenario, where different flows with different QoS parameters and Best Effort traffic, were simultaneously sent through a path across different networks. The experimental study addressed in this work helped on the enhancement of NSIS developments for IPv6 environments.**

.

I. INTRODUCTION

One of the Internet main features, on its creation, was that all packets have the same treatment. This is also known as the Best Effort paradigm, where packets are forwarded in the routers based on a First Come First Served politic. Therefore, no arrangements were made to guarantee bandwidth, delays or any other services summarized under the term Quality of Service (QoS). However, due to the explosive growth of the Internet, and to the exponential use of multimedia services, like VoIP, QoS mechanisms are being proposed and developed. One of these brought the ability of creating reservations between nodes for carrying the QoS information along the data path, providing routers with the particular QoS needs of each flow. Particularly, the design of the Resource reSerVation Protocol (RSVP) by Internet Engineering Task Force (IETF) fit these needs for a while. However, RSVP turned out to suffer from flexibility to meet today's requirements [6].

Meanwhile, new signaling mechanisms started rising up, and in 2001 the Next Steps In Signaling (NSIS) [9] was formed to create a new signaling framework, capable of supporting future needs. This working group focused primarily on the architecture and design of a new signaling protocol for the Internet. From the beginning of these discussions, new implementations were created and used to test and validate the feasibility of the designs proposed. As a matter of fact, students at the University of Kentucky and University of Göettingen completed their implementations based on the early drafts, with modular interfaces of generic signaling services NSIS Signaling Layer Protocols (NSLP). There are known implementation activities at Siemens Roke Manner Research, NEC, Nokia, Alcatel and University of Coimbra; Ericsson, the University of Karlsruhe, the University of Twente, and Samsung are working towards independent implementations of QoS NSLP. In this paper we used the University of Goettingen implementation, since it seamed to be the most functional and robust. We implemented a testbed with NSIS running through IPv6. The tests performed had the purpose of seeing in action not only the NSIS signaling mechanisms, but also the captured packets, nodes associations, as well as the QoS mechanisms implemented in the network, in order to provide a solid NSIS background in future work.

II. BACKGROUND

A. NSIS basics

As a signaling protocol, NSIS focus on the manipulation of state in nodes along the data path, taken by a data flow. Here, data flow means a number of packets with the same source and destination address, marked with the same flow identifier. It is assumed that the data path between nodes is defined by routing protocols and so, NSIS works in a seamless way, interacting with all nodes along the data path.

To get the flexibility needed to meet today's requirements, NSIS splits itself in two layers: the first one is a lower level layer, responsible for the transport of signaling packets; the second is an upper level layer, responsible for the signaling between the intervenient nodes.

NSIS was designed to support many signaling applications that manipulate states in the NSIS nodes along the data path. Note that some nodes may not support NSIS and, as a consequence of that, two NSIS neighbours can have one or more nodes between them. Therefore, the NSIS Initiator (NI) starts the signaling process, while the others NSIS Entities (NE) along the data path intercept and forward the messages until they reach the NSIS Receiver (NR).

B. NSIS main principles

NSIS, as said before, splits itself in two protocol layers. The first of them is called NSIS Transport Layer Protocol (NTLP) and insures the transport of all signaling messages between all nodes. That is possible because NTLP is primarily composed of a specialized messaging layer, denoted as General Internet Signaling Transport (GIST). GIST is responsible for the discovery of signaling aware nodes along a flow path, as well as for the maintenance of transport layer connection along the discovered paths, and operates on top of existing transport protocols (TCP, UDP, SCTP, DCCP or any other one).

The second is denoted as NSIS Signaling Layer Protocol (NSLP) and deals with signalling application-specific functionality. NSLP refers to actual signaling operation or signaling applications like QoS NSLP (used for QoS reservation) and NAT/FW-NSLP.

The concept of splitting NSIS in two layers makes itself more generic, extensible and flexible, since each layer has its responsibility, and together they provide its functionality.

Unlike RSVP, the NSIS decoupling of peer discovery from the signaling message transport mechanism makes possible the use of standard security protocols or transport layer protocols. That is done by introducing a discovery component in NTLP, which can rely on IP router alert option or other approaches, such as routing tables.

In the other hand, with the creation of a session identifier is possible to identify a signaling session and signaling state, independent of a flow identifier.

Furthermore, NSIS signaling is applicable in different parts of the Internet, as well as may be triggered in different ways, facts that allow the signaling to be initiated and terminated in different parts of the network, such as end hosts, domain boundaries or interior routers. Thus, NSIS protocol offers support for many signaling exchanges: end-to-end (performed between end hosts), edge-to-edge (performed between boundary nodes of the same domain) and end-to-edge (host-to-network scenarios).

III. NSIS TRANSPORT LAYER PROTOCOLS (NTLP)

The functionality of NTLP is based on the following principle: its mechanisms will only operate on its neighbour NSIS Entities (NE). Thus, NTLP consists on a set of *hop-by-hop* protocols.

Taking this in consideration, its functional mechanism can be described in the following: when a certain NE is ready to send a signaling message, delivers it to NTLP with its data flow information. Therefore, it is the NTLP responsibility to deliver it to the next NE in the data path. From the perspective of a NE that receives a signaling message, one of two things can happen: 1) NTLP forwards it to the next NE in the data path (case it exists); 2) If an appropriate local signaling application exists in the NE, it will receive the message from NTLP. In this last case, the signaling application will not only process the received signaling message, but also creates another message to be sent by NTLP to the next NE.

So, NTLP offers transport-layer services to higher-layer signaling applications for two purposes: sending/receiving signaling messages, and exchanging control and feedback information. Since all messages are treated locally, NTLP functional mechanism is quite simple, taking in consideration that operations such as endpoints discovery, security and NAT translations are no longer required.

A. GIST (General Internet Signaling Transport)

GIST has two goals: NE’s discovery along the data path and establishing a Message Routing State (MRS) in each session. Instead of creating a new transport protocol, GIST reuses existing transport and security protocols, in order to provide a universal message transport service. As a soft-state protocol, GIST is responsible for the creation and maintenance of two different states, both related to signaling transport: a per-flow message routing state for managing the processing of outgoing messages, and a message association state for managing per-peer state associated with connection mode messaging to a particular peer. This consists of signaling destination address, protocol and port numbers, internal protocol configuration and state information. Besides information about its neighbour NE, GIST also maintains certain message routing information such as the flow identifier, the NSLP type and session identifier, to uniquely identify the signaling application layer session for a flow.

GIST can operate in two modes: *datagram* or *connection*. While the first one uses an unreliable unsecured datagram transport mechanism (taking UDP as a first choice), the second uses any stream or message-oriented transport protocol (being TCP the first choice). Both modes can be used in the different nodes that compose the data path, without coordination or manual configuration, allowing the use of datagram mode at the edges of the network and connection mode in its core.

B. GIST Messages:

GIST defines 6 different messages.

The GIST-QUERY is always sent before any association between nodes, to test if the destiny node can or cannot proceed to the message association. This message is only sent in datagram mode and must include a Stack Proposal. Since it always elicits for a response, the FLAG R must be set (R=1).

The GIST-RESPONSE is present on datagram or connection mode. However, in the first case it is necessary to include a Responder Cookie, as well as its own Stack Proposal and Configuration Data. It must echo the Message Routing Information (MRI) (with inverted direction) Session ID (SID) and Query-Cookie of the Query.

The GIST-CONFIRM may be sent in datagram or connection mode (if a messaging association has been reused). It must echo the MRI (with inverted direction) SID and Responder-Cookie if the Response carried one.

A plain GIST-DATA message, in the other hand, contains no control objects, but only the MRI and SID associated with the NSLP data being transferred.

The GIST-ERROR message goal is to report a problem occurred at the GIST level. In datagram mode, this message includes a Network Layer Information object for the originator of the error message.

Finally, the GIST-HA-HELLO message is sent only in Connection Mode, in order to indicate that a node wishes to keep a messaging association open [5].

C. Security in the 3-Handshake

In order to prevent from several possible attacks, GIST uses a cookie mechanism. It starts with the Querying node inserting a cookie into the Query message. This cookie will be echoed by the Responder Node, which will also add its own cookie. This last cookie will be included in the confirm message, as shown below:

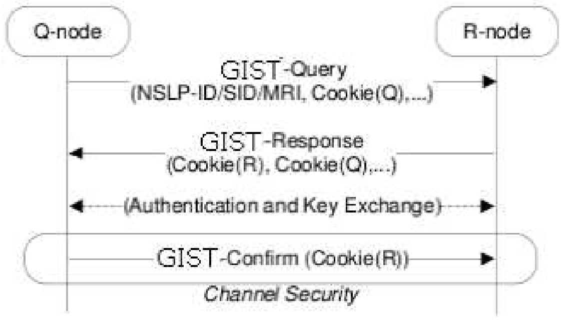

Q-node
R-node
GIST-Query
(NSLP-ID/SID/MRI, Cookie(Q),...)
GIST-Response
(Cookie(R), Cookie(Q),...)
(Authentication and Key Exchange)
GIST-Confirm (Cookie(R))
Channel Security

Figure 1 – GIST: MRS setup

This mechanism is not only a way of transferring information between nodes with authentication, but also prevents against spoofing of the Query, Response and Confirm messages, since the hacker would face the need to guess the cookie.

IV. QOS SIGNALING APPLICATION PROTOCOL

One of the elementary NSIS principles is that all signaling applications use the generic functionality provided earlier by the NTLP. In a NSIS node, the request for QoS may be initiated either by network management or by a local application request, initiated by a user application. Only messages related to QoS are passed up to the QoS NSLP processing module. This signaling application can signal for any QoS model, namely Intserv or Diffserv. Reservation-specific parameters, such as available bandwidth and token bucket sizes, are encapsulated in a QSPEC object, and then carried from one QoS NSLP node to another. These parameters ensure some degree of interoperability in several QoS Models, providing a common language to be re-used.

In each QoS NSLP node, it is present a RMF (Resource Management Function) responsible for handling the QoS requests, specifically the QSPEC. There is also a local QoS Model that describes how the RMF should interpret the QSPEC as well as how to grant and configure the resource. In the other hand, the grant processing involves two additional local decision modules, namely *policy control* and *admission control*. In the end, the QoS NSLP node may resort to acknowledge messages to indicate that the required resources have been correctly configured. These messages are unidirectional and the QoS NSLP node may propagate the resource request further along the path towards the data receiver.

A. QoS NSLP messages:

The QoS NSLP is a soft state protocol and defines four different types of messages. These are:

The RESERVE message, unlike all the other QoS NSLP messages, manipulates QoS NSLP reservation state, by creating, modifying, refreshing or removing it.

The QUERY message, without making a reservation, requests information about the data path. This can be used to “probe” the network for path characteristics, either for support of certain QoS models or for receiver-initiated reservations.

The RESPONSE message provides information about the result of the QUERY message.

Finally, the NOTIFY message provides information to a QoS NSLP node, differing from a RESPONSE message in the particular fact that it is sent asynchronously and need not refer to any peculiar state or previously received message. Therefore, the information conveyed by a NOTIFY message is typically related to error conditions.

Note that, unlike RSVP, QoS NSLP messages are sent NSIS peer-to-NSIS peer, and support both sender initiated and receiver initiated reservations.

V. TESTBED IMPLEMENTATION

In order to observe NSIS functionality, it was built a testbed using University of Goettingen NSIS implementation [1]. The used testbed is presented in figure 2.

In order to differentiate traffic, this implementation uses IPtables or IP6tables marking, corresponding to IPv4 or IPv6 addressing, respectively. However, this strategy has never turned out to be functional when using IPv6 addressing, since all incoming traffic in routers was recognized as unclassified traffic. In this sense, the source code was changed (namely the nsis-0.5.1-dev/nslp/qos/rmf/IpTablesWrapper.cpp file) to use U32 filters, based on source and destiny IPv6 address, instead of IP6tables marking. After this modification, as different flows were classified as Best Effort and QoS traffic, we could assume that the problem was in the IP6tables marking. This modified version can be downloaded in http://hng.av.it.pt/~fferreira/downloads/nsis-0.5.1-dev_u32.rar.

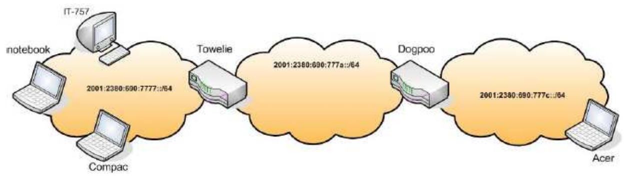


Figure 2 – Schmatic of the used testbed

Note that, in both experiments, ‘notebook’ was used as the QoS NSIS Initiator (QNI) while ‘Acer’ was used as the QoS NSIS Responder (QNR). IT-757 was responsible for background traffic generation (also having ‘Acer’ as its destiny). The ‘Compac’ terminal, used in the last experiment, was a QNI in a second QoS request to ‘Acer’.

Meanwhile, ‘Towelie’ was elected the network’s bottleneck, as the maximum overall bandwidth of both the interfaces was specified to 500kbps. Also, the default bandwidth requested on a QoS NSLP Reserve is 80 kbps, so traffic generated in ‘Notebook’ having ‘Acer’ as destiny (Flow 1) will have the same rate. Finally, the background traffic rate is 6 Mbps (much greater than the bottleneck capacity), generated with a Poisson distribution. These parameters were unchangeable for the following 2 experiments:

- <u>TEST 1 – Simple QoS Request:</u>

After IT-757 starts generating the background traffic having ‘Acer’ as destiny, ‘Notebook’ makes a QoS request and starts generating its traffic (Flow 1).

- <u>TEST 2 – Double QoS Request with different parameters:</u>

Similar to TEST 1, but ‘Compac’ terminal also does a QoS request and its traffic generation (Flow 2 has a rate of 160 kbps and the QoS request is for a bandwidth of the same rate).

Note that all traffic (Flow 1, Flow 2 and background) was generated with Mgen [2]. Also, using *bash* scripting, different scripts were created in order to make a QoS request before generating traffic for the correspondent flow. Additionally, a *clear.sh* script was done with the purpose of clearing *qdiscs* and classes created on both routers. The *Tcpdump* tool was used in ‘Acer’ in order to capture and control incoming traffic.

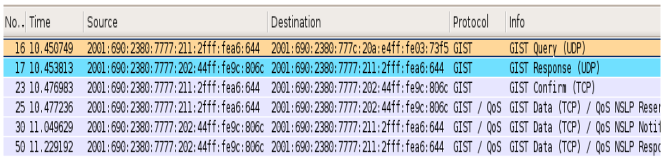
In TEST 2, the source code (particularly the file /nsis-0.5.1-dev/nslp/qos/qos_client.cpp) was changed in ‘Compac’ terminal, so that a reservation with different parameters could be done.

VI. RESULTS

This section presents the results of each experiment, as well as their respective details.

A. TEST 1

Using Ethereal – Network Protocol Analyser [3], as well as the adequate dissectors [4], all packets related to node association (either transport or signaling layer) were captured, visualized and presented in the figure below.

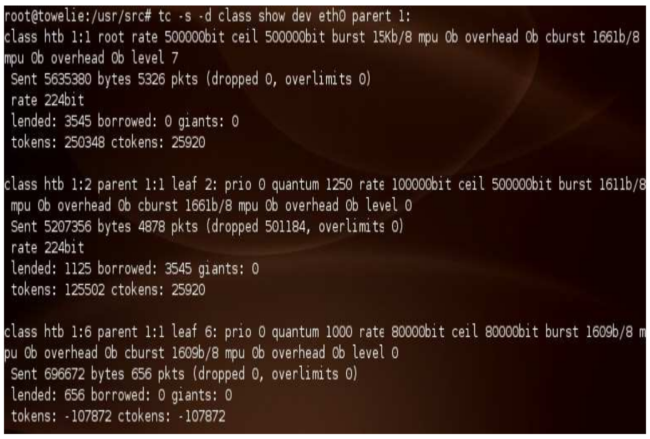


No..	Time	Source	Destination	Protocol	Info
16	10.450749	2001:690:2380:7777:211:2fff:fea6:644	2001:690:2380:777c:20a:e4ff:fe03:73f5	GIST	GIST Query (UDP)
17	10.453813	2001:690:2380:7777:202:44ff:fe9c:806c	2001:690:2380:7777:211:2fff:fea6:644	GIST	GIST Response (UDP)
23	10.476983	2001:690:2380:7777:211:2fff:fea6:644	2001:690:2380:7777:202:44ff:fe9c:806c	GIST	GIST Confirm (TCP)
25	10.477236	2001:690:2380:7777:211:2fff:fea6:644	2001:690:2380:7777:202:44ff:fe9c:806c	GIST / QoS	GIST Data (TCP) / QoS NSLP Reser
30	11.049629	2001:690:2380:7777:202:44ff:fe9c:806c	2001:690:2380:7777:211:2fff:fea6:644	GIST / QoS	GIST Data (TCP) / QoS NSLP Notif
50	11.229192	2001:690:2380:7777:202:44ff:fe9c:806c	2001:690:2380:7777:211:2fff:fea6:644	GIST / QoS	GIST Data (TCP) / QoS NSLP Respo

Figure 3 – Captured packets in a NSIS Session

While the first 3 captured packets are related to the transport layer, and therefore use GIST protocol, the last 3 packets are related to signaling for the QoS requested. As explained before, the node association (transport layer) is done with the 3-handshake performed with GIST Query-Response-Confirm messages. In the other hand, the intervenient nodes negotiate the request for QoS (signaling layer) with the QoS NSLP Reserve-Notify-Response messages.

Additionally, in this test it was possible to see in real-time the different classes created on routers (Figure 4). Particularly, the existence of class 1:1 root, as well as two subclasses: Class 1:2 and Class 1:6. While the former handles with unclassified traffic, the latter handles with QoS traffic (Flow 1). This explains why there are so many drops of packets in the first case (Best Effort treatment), while all packets in the latter subclass are forwarded (QoS service treatment).



```
root@towelie:/usr/src# tc -s -d class show dev eth0 parert 1:
class htb 1:1 root rate 500000bit ceil 500000bit burst 15Kb/8 mpu 0b overhead 0b cburst 1661b/8
mpu 0b overhead 0b level 7
 Sent 5635380 bytes 5326 pkts (dropped 0, overlimits 0)
 rate 224bit
 lended: 3545 borrowed: 0 giants: 0
 tokens: 250348 ctokens: 25920

class htb 1:2 parent 1:1 leaf 2: prio 0 quantum 1250 rate 100000bit ceil 500000bit burst 1611b/8
 mpu 0b overhead 0b cburst 1661b/8 mpu 0b overhead 0b level 0
 Sent 5207356 bytes 4878 pkts (dropped 501184, overlimits 0)
 rate 224bit
 lended: 1125 borrowed: 3545 giants: 0
 tokens: 125502 ctokens: 25920

class htb 1:6 parent 1:1 leaf 6: prio 0 quantum 1000 rate 80000bit ceil 80000bit burst 1609b/8 m
pu 0b overhead 0b cburst 1609b/8 mpu 0b overhead 0b level 0
 Sent 696672 bytes 656 pkts (dropped 0, overlimits 0)
 lended: 656 borrowed: 0 giants: 0
 tokens: -107872 ctokens: -107872
```

Figure 4 – Different traffic classes on router Towelie (Test 1)

Furthermore, it was possible to see the ‘Acer’ incoming traffic, as well as the rate of each traffic flow (Figure 5). At the beginning, background traffic, represented in red, arrives at a 500kbps rate (maximum overall bandwidth of the network’s bottleneck: Towelie). However, approximately 10 seconds later, Flow 1 (with 80 kbps rate) is started, making background traffic’s rate drop approximately to 420 kbps.

Two conclusions can be drawn: 1) the QoS treatment applied to Flow 1 assures that all its packets will arrive to its destiny as soon as possible; and 2) even more drops of background traffic will occur in the network’s bottleneck, router Towelie, as less bandwidth for unclassified traffic is available in router Towelie, due to the start of Flow 1.

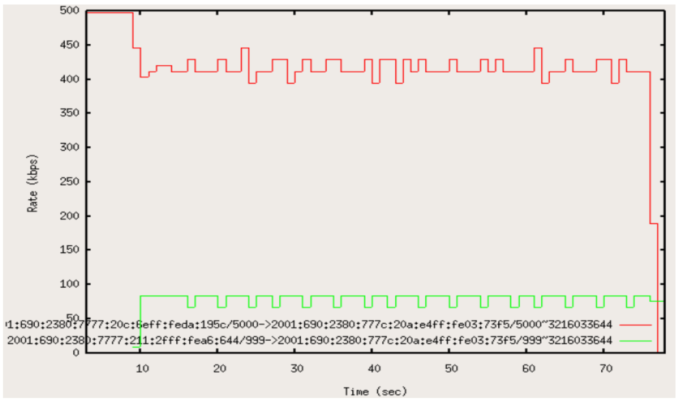
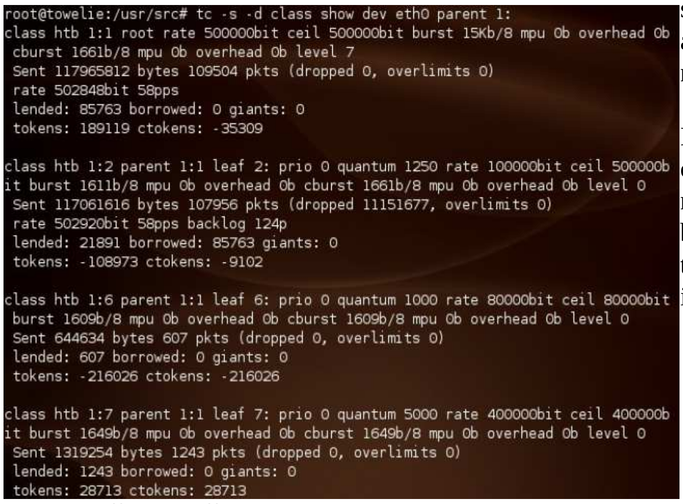


Figure 5 – Incoming traffic in ‘Acer’ terminal (Test 1)

B. TEST 2

As a new QoS request was made from ‘Compac’ to ‘Acer’, it was possible to see a new traffic subclass created on both routers, particularly Class 1:7 (Figure 6). Again, no packets of these classes with QoS treatment were dropped. However, and as expected, the same did not happen with Class 1:2 (Best Effort treatment), where a significant number of drops occurred.



```
root@towelie:/usr/src# tc -s -d class show dev eth0 parent 1:
class htb 1:1 root rate 500000bit ceil 500000bit burst 15Kb/8 mpu 0b overhead 0b
 cburst 1661b/8 mpu 0b overhead 0b level 7
 Sent 117965812 bytes 109504 pkts (dropped 0, overlimits 0)
 rate 502848bit 58pps
 lended: 85763 borrowed: 0 giants: 0
 tokens: 189119 ctokens: -35309

class htb 1:2 parent 1:1 leaf 2: prio 0 quantum 1250 rate 100000bit ceil 500000b
it burst 1611b/8 mpu 0b overhead 0b cburst 1661b/8 mpu 0b overhead 0b level 0
 Sent 117061616 bytes 107956 pkts (dropped 11151677, overlimits 0)
 rate 502920bit 58pps backlog 124p
 lended: 21891 borrowed: 85763 giants: 0
 tokens: -108973 ctokens: -9102

class htb 1:6 parent 1:1 leaf 6: prio 0 quantum 1000 rate 80000bit ceil 80000bit
 burst 1609b/8 mpu 0b overhead 0b cburst 1609b/8 mpu 0b overhead 0b level 0
 Sent 644634 bytes 607 pkts (dropped 0, overlimits 0)
 lended: 607 borrowed: 0 giants: 0
 tokens: -216026 ctokens: -216026

class htb 1:7 parent 1:1 leaf 7: prio 0 quantum 5000 rate 400000bit ceil 400000b
it burst 1649b/8 mpu 0b overhead 0b cburst 1649b/8 mpu 0b overhead 0b level 0
 Sent 1319254 bytes 1243 pkts (dropped 0, overlimits 0)
 lended: 1243 borrowed: 0 giants: 0
 tokens: 28713 ctokens: 28713
```

Figure 6 - Different traffic classes on router Towelie (Test 2)

With TCPdump tool it was possible to analyse the per-flow bandwidth rate for each traffic flow on routers (Figure 7). In red, Best Effort traffic is send to ‘Acer’ with a rate of 500kbps (maximum overall bandwidth of router Towelie). This rate dropped approximately to 420 kbps with the appearance of Flow 1 (80 kbps), represented in green. However, about 40 seconds later, Flow 2 (160kbps) is stated, also with QoS treatment, and due to that, background traffic’s rate drops again, this time to approximately 260 kbps.

Twenty seconds later, the green line drops (Flow 1 is stopped) and due to that, 80kbps more of bandwidth are now available to unclassified traffic in router Towelie: the red line will arise that rate. As expected, the same behaviour will happen with the end of Flow 2: this time 160kbps of bandwidth will become available to unclassified traffic.

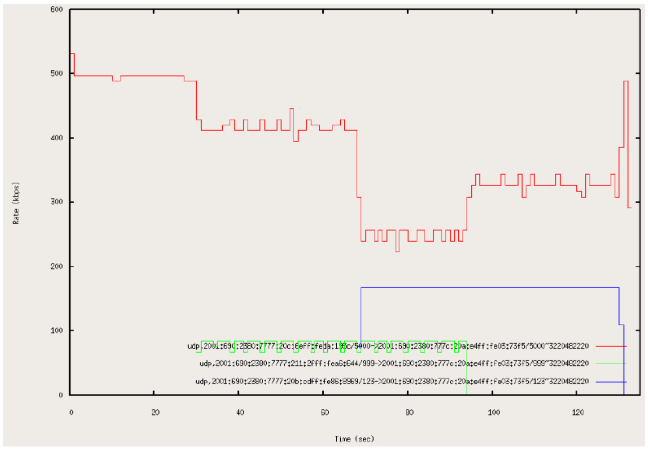


Figure 7 - Incoming traffic in ‘Acer’ terminal (Test 2)

Background traffic’s available bandwidth depends on the number of QoS flows and its reserved bandwidth. Therefore, when both QoS flows are active simultaneously, less bandwidth is available to unclassified traffic, and more drops will occur. For each instant, the sum of the rates of the active flows, in that instant, is always equal to the maximum overall bandwidth of the network’s bottleneck, router Towelie (500kbps).

TCPdump tool also permitted a different analysis: packet Inter-arrival time, as shown below on Figure 8. Taking in consideration that both Flow 1 and Flow 2 (again, represented in green and blue respectively) have reserved bandwidth for each, it is expected their inter-arrival time to be constant. Since Flow 2 has twice the rate of Flow 1, its inter-arrival time is half the value of the Flow 1.

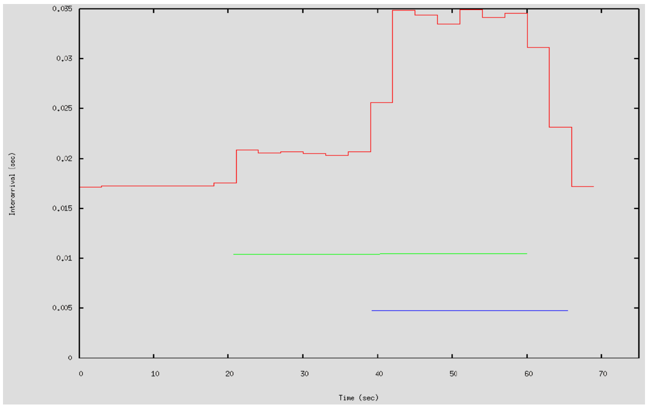


Figure 8 – Inter-arrival time of each flow

However, the analysis of the background traffic inter-arrival time may be more complex. Represented in red, this value is approximately constant until the start of Flow 1. As less bandwidth is available in router Towelie for unclassified traffic, its inter-arrival time will rise. After the start of Flow 2, the red line rises even more and becomes also uncertain, instead of being approximately straight as before. Finally, as both Flow 1 and Flow 2 were stopped, the background traffic's inter-arrival time successively decreases.

VII. CONCLUSION AND FUTURE WORK

One of the main goals of these experiments was to provide QoS mechanisms to a testbed using NSIS protocol and observe it functionality. As QoS traffic is forwarded in routers as soon as possible with no packet drops, the QoS in the testbed is verified. The bottleneck's link was always congestioned, but only unclassified traffic was dropped.

On the other hand, the inter-arrival time of both flows with QoS is constant. This proves that, for both flows, no packets are dropped or lost. The same does not happen with unclassified traffic, as its inter-arrival time rises with the fall of available bandwidth.

A major problem of this implementation is the fact that only one QSPEC model is supported. Therefore, all tested scenarios are not as realistic as they could be. Additionally, one future step in this work could be the inclusion of a new Ethernet interface on router Towelie, allowing both Notebook and IT-757 to be directly connected to it, permitting experiments using much higher transmission rates (more realistic scenario). NTP (Network Time Protocol) could be used to synchronize all the machines and determine packet delays for all traffic classes. However, QoS packets are forwarded as they arrive in the routers and therefore, their delay is practically the sum of its processing time in each router.

The major step for future work would be the inclusion of NSIS mechanisms in the existing Advanced Router Mechanisms (ARM) module, used in Daidalos I and Daidalos II architecture, currently using RSVP, unsuited for today's demanding requirements.

REFERENCES

[1] University of Goettingen, *Next Steps in Signaling Implementation*, http://user.informatik.uni-goettingen.de/~nsis/home.html

[2] Network and Communication Systems Branch, *Multi-Generator (MGEN)*, http://cs.itd.nrl.navy.mil/work/mgen/

[3] Ethereal: A network protocol analyzer. http://www.ethereal.com.

[4] NSIS Ethereal dissector: NSIS protocol analyzer for Ethereal.

[5] http://user.informatik.uni-goettingen.de /~nsis/

[6] FU, X., Juchem, I., Dickmann, C., *Design Options of NSIS Diagnostics NSLP* , Univ. Goettingen, 2006

[7] DICKMANN, C.; *An Implementation and Evaluation of the General Internet Signaling Transport (GIST) Protocol*, September, 2005

[8] SCHULZRINNE, H., Hancock, R., GIST: *General Internet Signaling Transport*, 2007

[9] FU, X., Bader, A., et alli, *NSIS: A New Extensible IP Signaling Protocol Suite* , 2005

[10] FU, X., Manner, J., RFC:4094 - *Analysis of Existing Quality-of-Service Signaling Protocols*, 2005

[11]