Project Number:	**IST-2001-32603**
Project Title:	**6NET**
CEC Deliverable Number:	**32603/Partner/DS/4.4.2/A1**
Contractual Date of Delivery to the CEC:	31 January 2005
Actual Date of Delivery to the CEC:	05 April 2005
Title of Deliverable:	Report on IPv6 QoS tests
Work package contributing to Deliverable:	WP4
Type of Deliverable*:	R
Deliverable Security Class**:	PU
Editor:	K. Stamos, D. Primpas
Reviewers:	Ch. Edwards, M. Dunmore
Contributors:	P. Tipper, Ch. Bouras, Ath. Liakopoulos, G. Van de Velde, Julio Orozco, David Ros

* Type: P - Prototype, R - Report, D - Demonstrator, O - Other

** Security Class: PU- Public, PP – Restricted to other programme participants (including the Commission), RE – Restricted to a group defined by the consortium (including the Commission), CO – Confidential, only for members of the consortium (including the Commission)

Abstract:

This document presents the second phase of QoS tests performed in 6NET. A team of 6NET partners performed QoS tests across 6NET's backbone network, and in some local test-beds. The goal was to make the 6NET network QoS enabled and to analyse the operation and interaction of QoS mechanisms, such as shaping and queuing, with IPv6 production hardware network infrastructure.

Keywords: QoS, IPv6, DiffServ, QoS mechanisms, Policing, Shaping, EF. AF

Executive Summary

This deliverable is the report from the second phase of QoS activity in 6NET. The first version of this deliverable only briefly described the results from this task as it was mostly concerned with local QoS tests performed during the first phase.

The second phase of QoS tests took into account the defined QoS model and extended the QoS activity across 6NET's backbone network. Initially, the proposed QoS model, which is based on the Differentiated Services (DiffServ) architecture, was studied taking into account several aspects, such as network dimensioning, policing and admission control. In particular, the 6NET QoS model includes the implementation of the IP Premium service, which is based on the EF (expedited forwarding) PHB (Per Hop Behavior) and provides absolute prioritization to this type of traffic. It also includes the BE (Best Effort) and the LBE (Less than Best Effort) services. Afterwards, the appropriate configuration to allow 6NET to provide QoS services was formed and possible conflicts with other network services were identified. The configuration was applied to all 6NET's backbone network routers providing the aforementioned QoS services to the connected NRENs. Finally, a number of tests were defined in order to evaluate the performance of the QoS services in the 6NET network. The prioritization of IP Premium traffic was examined, by testing the marking, traffic policing and traffic shaping mechanisms. A wide range of scenarios with various patterns of foreground and background traffic were performed in order to thoroughly investigate the applied QoS model and its impact.

The performed QoS tests in 6NET backbone indicated that this IPv6 QoS model can be the basis for production services. The mechanisms under test operated efficiently, as the experimental results prove, without any performance degradation on backbone routers. The above observations remained true even in heavy network congestion and the final performance results that a user or an application experienced correspond to the QoS service's specification.

In this document a method is also presented for streaming MPEG-2 video over IPv6 networks supporting the Differentiated Services (DiffServ) architecture, in particular the Assured Forwarding (AF) per-hop behaviour. The scheme takes advantage of both AF's packet drop priorities and MPEG-2 video semantics, so as to protect the most important information in terms of visual quality and reduce distortion under network congestion. The proposed method has been implemented in the VideoLAN streaming system, deployed in ENST's local IPv6 testbed and evaluated by means of subjective video quality measurements. The results show that the proposed method achieves a higher quality than traditional best-effort streaming.

In this document a discussion is also made on the IPv6 Flow Label field, which (similar to the Traffic Class field which is used for the marking of premium traffic in our experiments) is also related to traffic management issues. It is noted that the IPv6 flow label is not currently used in the IPv6 world but its use is expected in the near future. The IPv6 flow label has been recently standardized and prior to it becoming widely adopted by applications, many aspects, such as security and admission control, have to be investigated in depth.

Table of Contents

1 Introduction

The structure of this document is organized as follows: Chapter 2 describes the 6NET QoS framework where the main aspects of the DiffServ architecture are presented and the QoS model that the 6NET project adheres to is explained. Chapter 3 analyses the whole setup of 6NET QoS tests paying attention to measurements, traffic generation and the network testbed. Chapter 4 presents the tests that were performed including their results. The following chapter, i.e. chapter 5, describes some tests that were conducted by individual partners in local testbeds using the AF (Assured Forwarding) per-hop behaviour class of service. Chapter 6 gives a short description of the usage of IPv6 flow label, as it has been recently standardized. Finally, chapter 7 has the overall conclusions, chapter 8 the relevant bibliography and Appendix A the detailed configuration templates that were applied on the 6NET backbone network.

2 6NET Quality of Service Framework

2.1 Introduction

Several approaches for supporting QoS in best effort IP networks have been developed and described in research papers and articles , and a number of research projects could demonstrate their viability in the context of IPv4 networks. In principle, the same results should apply in the case of IPv6. 6NET is addressing this issue by actually demonstrating QoS support in IPv6 networks.

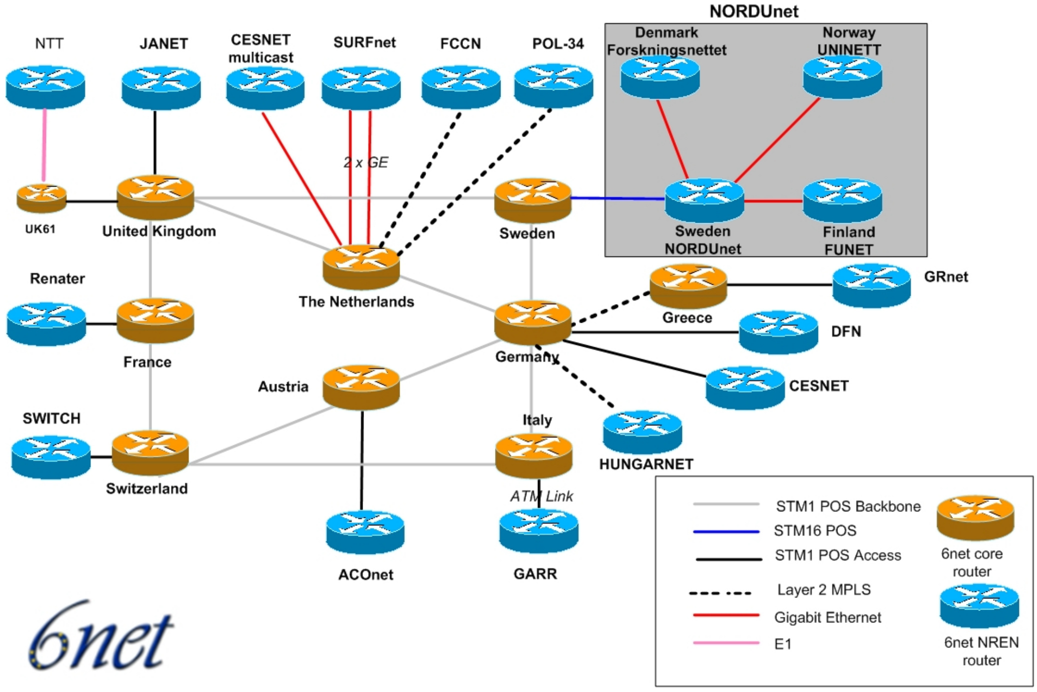

Figure 1: 6NET's backbone network

The goal of the IPv6 QoS activity is to show that approaches for supporting QoS in IPv4, smoothly migrate to the IPv6 environment. At the first phase of the project several QoS mechanisms were examined by different partners in their local testbeds. It was demonstrated that QoS mechanisms in IPv6 routers perform as expected and, hence, they can be taken as basis for wide-scale deployment of QoS services. The next step was, inevitably, the deployment of QoS services in the actual 6NET network (Figure 1). It had to be validated that adequate QoS support could be realized in the 6NET network and the special requirements of various applications, e.g. real time applications, could be fulfilled with the available mechanisms supported in the 6NET core routers.

The QoS mechanisms that are applied in the 6NET backbone adhere to the Differentiated Services (DiffServ) architecture, which ensures both the scalability and efficiency required in this portion of the network. Based on the DiffServ architecture, 6NET QoS activity proposes a suitable QoS framework, which defines a small set of QoS classes that are deployed in the core network.

The deployed QoS service model comprises of three different service classes. A high priority service IP Premium, which is based on the EF-PHB, is provided to a small portion of network traffic, especially for traffic generated with real time applications. The BE (best effort) and LBE (less than best effort) services can be used by elastic applications that do not require delay or bandwidth guarantees.

The deployed QoS service was tested through multiple distinct tests that aim to verify that basic QoS mechanisms, such as traffic policing, are performing as expected in the 6NET WAN environment. Most of the tests are “objective”, which means that standardised methodology is followed in order to evaluate the performance of the network. Software tools are used to generate both the testing and background traffic and the collected data is analysed.

2.2 The DiffServ QoS Architecture

The 6NET backbone adheres to the Differentiated Services (DiffServ) framework in order to offer Quality of Service (QoS) to different portions of traffic passing through the access and core network.

The DiffServ architecture minimizes the number of actions to be performed on every packet at each node and builds a configuration that does not use a signaling protocol. Individual DiffServ mechanisms are applied on traffic aggregates rather than individual flows.

DiffServ operates on a per-hop basis, with each router examining the IPv6 Traffic Class bits in each packet header to obtain the DiffServ code point (DSCP) for that packet. When the packet is ready to be queued for transmission, the router places the packet into an appropriate queue according to the DSCP value.

Service definitions can be specified with relative simplicity, and therefore allow the service deployment in a short timescale. The end-to-end path is composed of parts that belong to consecutive DiffServ-enabled domains. No particular knowledge is required about the internal topologies, physical structure or transmission technology of any of the domains in the path.

Where DiffServ mechanisms are deployed across network borders, close co-ordination is needed to ensure that each domain is aware of the DSCP values used for each class of service from other domains. If the DSCP value for the same service is different in two neighbor networks, all traffic passing through the network boundaries must have the DSCP value changed (remarked) to the correct local value.

A number of decisions are required prior implementing a DiffServ-enabled QoS network; how many classes of service should be provided, and how and where should the bandwidth limit for each class be set? If a limit is exceeded, should out-of-profile packets be dropped or be placed in the lower quality queue? On the one hand, if out-of-profile packets are dropped, network resources may be underutilised. On the other hand, if high-priority (EF) out-of-profile packets are placed in a low quality queue, then packet reordering may take place and bandwidth may also be wasted. (Note that real time applications usually drop out-of-sequence packets, so they prefer the packet dropping of exceeded traffic).

Use of high-priority class of service (CoS) should be strictly controlled to prevent their use by unauthorised data flows. For example, a commercial service provider would not accept traffic from a customer that is not paying for premium services. This implies that some form of admission control is needed, to prevent abuse of preferential queuing services. Admission control can be “undertaken” by the sending side (e.g. use of "shaping") or by the network receiving the traffic (e.g. use of "policing").

Policing admits traffic up to a certain bandwidth rate limit, and then either drops packets (usually the default), or places them in a different queue as it remarked them with different DSCP value. As policing drops packets indeterminately, it can have a serious effect on throughput and deteriorate services to the preferential traffic class. In most cases, the sending side shapes the traffic, using buffering techniques. This smoothes out bursts of traffic, keeping the traffic rate within the limit so that packets are not dropped by policing.

Policing is usually performed according to three parameters: IP source and destination prefixes and the agreed sending rate. Policing is performed by means of a token bucket. The token bucket depth is chosen larger than one MTU and varies according to the provided services. For example, high priority services usually have smaller bucket size in order to minimize jitter. Also, bucket size is usually larger in the access than in the core interfaces. The size of the bucket size may influence the packet loss, at the price of a small increase of delay variation. This can be experimentally verified.

At the initial policing point, packets successfully admitted to the service are marked with an appropriate DSCP value. It is possible for the policing configuration to simply enforce the bandwidth rate limit, and trust the DSCP values sent from the other domain. This would obviously be open to abuse from unauthorized sources marking their packets for preferential service.

Where DiffServ is in use between two network service providers, it is common for each provider to police ingress traffic to a reasonable limit, to prevent misconfigurations or denial of service (DoS) attacks from degrading every class of service on the network.

In many cases, the sending host or source should be required to shape flows it sends according to its allowed sending rate. While shaping by the router at the edge of the DiffServ domain should prevent traffic being policed in the next domain, shaping on a per-host basis should help to ensure fair access to the class of service between several hosts in the same domain.

2.3 6NET QoS Model

In the context of the QoS activity, WP4 partners deployed a limited number of traffic services in the core network. On the one hand, the number of services supported in the network was intentionally kept small in order to minimise complexity but, on the other hand, it was adequate for testing equipment functionality and supporting targeted applications in an IPv6 environment.

Therefore, 6NET supported three classes of service (in descending order of quality):

- IP Premium (IPP)
- Best Effort (BE)
- Less than Best Effort (LBE)

IP Premium (IPP) is a service that provides minimum latency and negligible packet loss to traffic and is suitable for real-time applications that require strict QoS guarantees. It may also be used to offer the equivalent of an end-to-end virtual leased line service at the IP layer across multiple administrative domains. IP Premium is a service defined at the IST SEQUIN project [17] and currently implemented for QoS treatment of IPv4 traffic over GEANT (the pan-European Research Network).

Best Effort (BE) is a service that does not provide any qualified guarantees to traffic and is suitable for elastic Internet applications, such as email applications. BE is based on the model of fair resource sharing where "sufficient" resources are available and applications are not starved by other greedy applications running in the background.

Less that Best Effort (LBE) is a service that exploits network resources without risk of jeopardizing or negative impact other traffic classes in the network. Typically, LBE service is used for background operations or bulk data transfer applications that require large scale network resources. Having less strict timing requirements, LBE applications behave in a "greedy" way, i.e. trying to expand as long network resources are available. However, LBE applications can quickly back down to a minimum share if other (higher priority) traffic is present. Note that a minimum bandwidth share is allocated to LBE traffic to prevent complete starvation of LBE flows, which would break established connections.

2.4 Class Specifications

2.4.1 IP Premium - Expedited Forwarding (EF)

The IP Premium implementation is based on the Expedited Forwarding (EF) Per- Hop Behavior of the DiffServ framework and IPP packets are forwarded immediately by the backbone routers, provided that IPP traffic does not exceed its maximum acceptable rate. IP Premium traffic is marked with a DSCP value of "46" and it is mapped to a dedicated queue at each output interface in the backbone routers. In 6NET, IPP traffic is intended for the use of real-time audio and video traffic, therefore policing will be configured to drop out-of-profile traffic, i.e. traffic exceeding the permitted access rate of a partner. An alternative action for out-of-profile traffic would be to remark it as best effort traffic. However, this would lead to packet reordering that significantly impacts the performance of TCP flows. Also, note that out-of-order packets are no use in real time applications.

IP Premium provisioning model is realised in three steps. Initially, a partner requests to use a portion of traffic that enters and exits from two specific 6NET access routers. The 6NET NOC will estimate whether there is enough resources across the network and, if yes, it will install the appropriate policers at the 6NET edge interfaces. Obviously, the provisioning model follows the "destination aware" approach and the 6NET network administrators perform the admission control manually. For each access link, the maximum IPP traffic that is allowed, corresponds to 5% of the access link's capacity.

2.4.2 Default/Best Efforts (BE)

Best effort traffic is marked with DSCP value of "0". No policing of BE traffic is configured, so BE traffic has access to all bandwidth not occupied by higher priority traffic classes.

Note that all traffic not authorised for accessing other QoS classes will be marked as BE by the 6NET access routers.

2.4.3 Less than Best Efforts (LBE)

LBE traffic will be marked with DSCP value of “8” and it is allocated a minimum portion at the access links, approximately 1% of the total link capacity. Although some networks make no minimum reservation for LBE traffic, a small reservation such as this should enable large data flows over TCP to continue at a low bit rate.

2.5 QoS Semantics – DSCP values

The “Traffic Class” (TC) one-byte field in the IPv6 packet header has the same semantics with the “Type of Service” (ToS) field in the IPv4 packet header. The 6 most significant bits in the TC byte define the DSCP value of the packet. According to the IPP, BE and LBE services, the DSCP values are set according to the following table (Table 1):

DSCP bits						**Decimal value of DSCP**	**Description**
1	0	1	1	1	0	46	IP Premium
0	0	0	0	0	0	0	Best Effort
0	0	1	0	0	0	8	Less than Best Effort (LBE)

Table 1: 6NET's valid DSCP values

The following actions are configured at the input interfaces of 6NET edge routers (in the direction from the NRENs towards to 6NET):

a) Police IPP traffic up to a predefined level according to already granted partners requests. Traffic exceeding this level will be discarded. The maximum allowed IPP traffic from each access links is 5% of total link’s capacity.

b) Traffic with invalid DSCP values is remarked as best effort.

Next, in internal output interfaces of 6NET domain, the following actions are configured:

a) IPP traffic is serviced via a strict priority or a high priority queue.

b) LBE traffic is treated as best effort in the core links and the policing at access interfaces is trusted. This choice has been made, as the network’s utilization is low and also there was not major interest for LBE service, therefore the expected traffic was low. In a real production network, usually the LBE traffic is granted at least 3% of the link capacity during heavily congested periods.

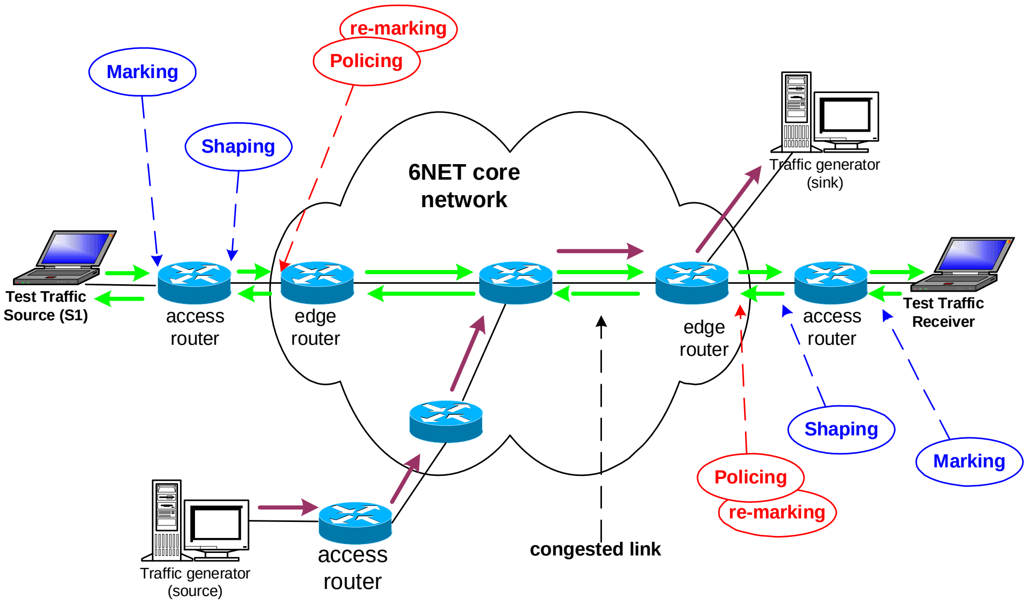

Figure 2: QoS framework's actions

2.6 Implementation of Framework

The implementation of the QoS framework in the 6NET network required the appropriate configurations to be enabled at the core and access routers, at least for the portion of the network that was used during the QoS tests (Figure 3). As QoS testing is a complex and time consuming process and because large amount of traffic has to be injected into the network under test, special care had to be taken in order to perform the QoS tests without any impact to other services.

In order to prevent problems caused by misconfiguration, the policing and DSCP marking/re-marking mechanisms were configured first on the 6NET access routers. Afterwards, the preferential queuing in the core network was enabled and all packets marked with the EF DSCP value started to be forwarded in the priority queue. If, for some reason, the EF queuing were misconfigured or not functioning as expected, this would cause disruption to the network and the supported services. Finally, the remaining part of QoS configuration was applied to 6NET's core routers.

The configuration can be separated into 2 categories; the configuration of the core 6NET interfaces and the configuration of the access interfaces. In the core interfaces, as shown in Figure 2, the only action that is done is the proper configuration of the queues. In detail, packets marked with DSCP value 46 that are received by a core router should be transmitted through a high priority queue. For this reason, the relevant configuration had been defined in the core interfaces (POS interfaces). This configuration consists of class-map called EF that matches all packets with DSCP value 46. Finally, the policy-map *my_policy* is defined to locate all packets with DSCP value 46 to the low latency (high priority) queue. All the other packets experience default queuing and therefore receive best effort service.

The second point that needs the appropriate QoS configuration is the access interfaces, where the policing and remarking actions take place. At this point, the 6NET framework distinguishes 3 different occasions.

- The first one corresponds to access interfaces that are not participating to the QoS tests. In these interfaces, the 6NET QoS framework applies a policy-map called NON-PARTICIPANT, which remarks all incoming traffic to DSCP value zero (0). The remarking is due to the necessity to prevent unauthorized traffic from using the high priority queues and therefore reducing the level of quality that the authorized traffic will experience.

- The second one corresponds to access interfaces that participate in QoS tests and the NREN (that is connected with this access interface) performs the marking of the IP Premium traffic. In this case, the access interfaces match the packets with DSCP value 46 and perform an aggregated policy. The 6NET framework defines that the maximum traffic profile that can be accepted from an access interface is 5% of the total capacity of the access interface. In addition, the LBE traffic, which corresponds to packets marked with DSCP value 8, is policed to 1% of the total capacity of the access interface. This case is implemented with a policy-map called TRUSTED-IN.
- The third occasion corresponds to access interfaces where the 6NET routers verify DSCP markings are valid, against an access list supplied by an NREN in advance and remark anything not authorised to BE. The policing is also performed on an aggregated basis and the IP Premium allowed traffic profile corresponds to 5% of total capacity of access link. LBE traffic is treated the same as in second occasion. This case is implemented with a policy-map calls FILTERED-IN.

The configuration, as described above, is presented in Appendix A: QoS configuration in 6NET routers, where it is written according to Cisco IOS format combined with the necessary comments. The policy-maps had been activated on all 6NET routers (in the backbone) and in the access interfaces the policy-maps NON-PARTICIPANT and TRUSTED-IN were used. The TRUSTED-IN was activated on the access interfaces of all the participants in the QoS tests, which were DFN, GRnet, JANET, NORDUnet and Renater. All other access interfaces were configured with the NON-PARTICIPANT policy-map to prevent unauthorized traffic from using the IP Premium service.

The whole QoS configuration was examined for conflicts or malfunctions with other services for a specific period during which no problems were mentioned. The QoS configuration was first activated in May 2004, making 6NET’s backbone a QoS enabled network and remained so until the network’s decommission (January 2005), which proves that QoS mechanisms operated steadily under IPv6. Their performance was evaluated in the QoS tests described in the following sections.

3 QoS tests in 6NET’s backbone network

3.1 Introduction

The first step for the QoS tests was to decide upon the tests that would be performed and what traffic metrics would be used to evaluate them. Second was to decide upon the software tools and applications that would be used to measure, monitor and generate the traffic. The final step was to setup the QoS testbed through 6NET’s backbone. The goal was to acknowledge that the QoS service operates well and also evaluate its performance by making the test as realistic as possible.

3.1.1 Traffic Metrics

There are multiple performance metrics proposed to measure the services provided in a QoS-enabled network. Most of them are defined by the IETF IP Performance Metric working group [16].

During the 6NET tests, the following parameters were used to qualify the provided QoS services:

- One-way or round trip delay
- Inter-packet delay variation (jitter)
- Packet loss
- Packet reordering

One-way delay is defined as the time needed by a packet to be transmitted and fully received by the destination. The overall time consists of the propagation delay, e.g. the time to transmit a bit over long-distance circuits, and the transmission time, e.g. the time to transmit a bit over a specific-speed circuit. As one-way delay measurements require strict synchronization among the monitoring systems in order to be able to reliably measure delay, clocks at the testing stations were synchronized using stratum 1 NTP server at ntps1-0.cs.tu-berlin.de

Inter-packet delay variation is measured for packets belonging to the same packet stream and shows the difference in the one-way delay that packets experience in the network. Large values for jitter usually reveal queuing delays in the network.

Packet loss is measured as the portion of packets transmitted but not received by the destination compared to the total number or packets transmitted. Large values of packet loss usually shows highly congested networks or frequent sharp increases in the traffic load.

Packet reordering is measured as the portion of packets that are delivered to the destination in the wrong order compared to the total number of packets. There are multiple reasons that lead to packet reordering; parallel forwarding engines in high performance routers, per packet load balancing on parallel physical links, routing path changes, etc. There is a significant impact to the TCP (application) performance even for small packet reordering values.

3.1.2 Monitoring and Measurement Requirements

Subscription to a premium class of service implies a Service Level Agreement (SLA) is signed between the customer and the service provider. In 6NET, the provided SLAs are clearly not commercial agreements but only define performance guarantees that are experimentally provided to portions of traffic.

Monitoring and measuring activities in 6NET should demonstrate the network infrastructure is able to provide services guarantees to portions of the traffic. For example, traffic marked as EF receives preferential treatment compared to BE and LBE under congestion conditions in the core or the access networks. Also, LBE traffic should also be reduced to a minimum level when other traffic is present.

QoS measurements can also be performed with software tools that are able to generate traffic with pre-defined characteristics and measure the performance of the network. A tool that is already extensively used in measurements is *iperf* , which is able to provide accurate throughput and jitter measurements for flows under test.

Iperf’s statistics were produced at the server instance of the Iperf traffic generator and included the average throughput and the average jitter of the UDP traffic and the average throughput of the TCP traffic. Iperf calculates jitter using the RFC 3550 definition that defines jitter as:

$$\mathbf{J_i = J_{i-1} + (| D(i-1,i) | - J_{i-1}) / 16}$$

where D(i,j) is the difference of the interval between two successive packets at the receiver from the interval between two successive packets at the sender, defined as

$$\mathbf{D(i,j) = (R_j - R_i) - (S_j - S_i)}$$

3.1.3 Traffic generators

In order to perform the QoS tests, the most suitable traffic sources have to be identified. Traffic sources should be able to generate data streams that "simulate" backbone traffic and/or specific application traffic profiles, e.g. VoIP calls. Also, traffic sources should be able to generate traffic that can congest the network and stress the QoS mechanisms enabled at the routers. For the 6NET tests, there were the following alternatives:

- Embedded software in routers
- Software tools
- Hardware traffic generators
- Real applications

The software tools were selected as they can be used to generated traffic in large amounts and they are easily installed. A common server equipped with a gigabit Ethernet interface could be able to create congestion on 6NET STM-1 core links. From all available software tools, partners selected the *iperf* tool that is able to produce foreground and background traffic for the QoS tests and also take network measurements. It works as a client/server application in which, once the connection is established, test packets are sent from the client to the server, which records the amount of bytes received and calculates the average throughput of the test. Iperf works over IPv6 and allows the tuning of several parameters of the test flows like packet size, test time and the server's port number. For some of the tests the *mgen* tool was also used, because of its advanced capabilities in producing variations in the artificial traffic according to predetermined scenarios. Also, the Ethereal network protocol analyzer was used in order to capture the packets and then be able to extract metrics like packet reordering, delay, and throughput.

3.2 QoS testbed

For the purposes of the experiments we used three (3) PC-based servers located at various points in the network. Premium traffic in the foreground was sourced from the United Kingdom (JANET network), was routed through the Netherlands and Germany, and was received at Greece. Furthermore, in the experiments where we wanted to artificially create background traffic, a generating PC at the Netherlands was used, which sent traffic through Germany to Greece.

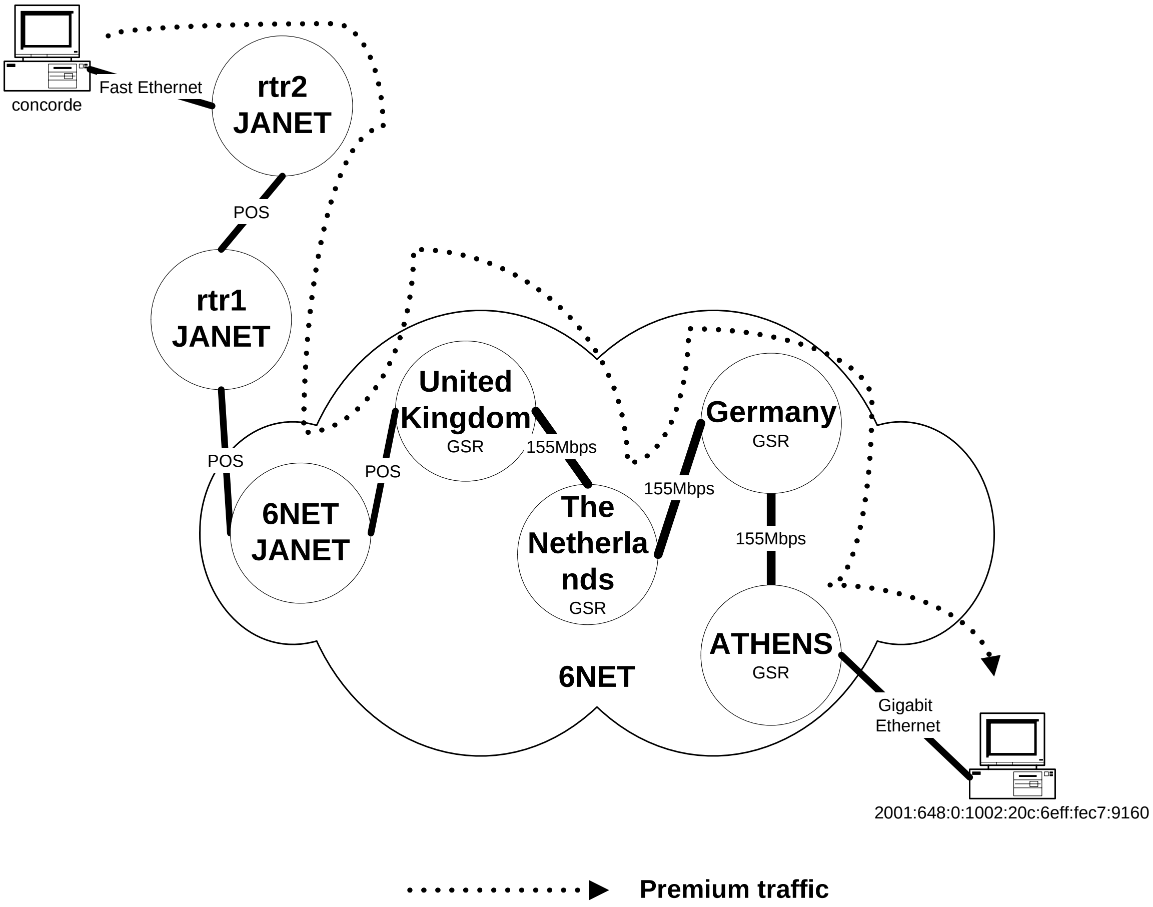

Figure 3: 6NET's QoS testbed

Table 2 displays the output of the *traceroute* utility to show the path used for packets sent from the UK PC (concorde) towards the testing PC at Greece.

```
traceroute to 2001:648:0:1002:20c:6eff:fec7:9160
(2001:648:0:1002:20c:6eff:fec7:9160)
from 2001:630:8:4:204:23ff:fe52:1b4a, 30 hops max, 16 byte packets


1 2001:630:8:4::1 (2001:630:8:4::1) 0.446 ms 0.367 ms 0.361 ms
2 po3-1.rtr1.ipv6.ja.net (2001:630:0:1::31) 0.607 ms 0.641 ms 0.476 ms
3 janet.uk6.uk.6net.org (2001:798:28:200::1) 0.736 ms 0.94 ms 0.605 ms
4 uk.nl6.nl.6net.org (2001:798:0:6::1) 9.354 ms 9.425 ms 9.225 ms
5 nl.de6.de.6net.org (2001:798:0:5::1) 18.099 ms 109.866 ms 17.851 ms
6 de.gr6.gr.6net.org (2001:798:0:b::2) 58.321 ms 58.293 ms 58.202 ms
7 2001:798:17:200::2 (2001:798:17:200::2) 58.075 ms 58.167 ms 58.072 ms
8 2001:648:0:1002:20c:6eff:fec7:9160 (2001:648:0:1002:20c:6eff:fec7:9160)
58.568 ms 58.205 ms *
```

Table 2: 6NET traceroutes for traffic in QoS tests

4 Description of QoS tests

The first stage of tests verified the correct operation of the marking, policing and shaping mechanisms for IP Premium Traffic. After this, 6NET partners conducted a number of scenarios that included simultaneous UDP and TCP background traffic, while the foreground traffic was alternated between the 2 transport protocols.

4.1 Investigation of the IP Premium mechanisms

The setup for this part of the experiments was simple and aimed to verify that traffic marked with the DSCP decimal value of 46 (IP Premium traffic) was indeed handled preferentially compared to the rest of the traffic. The verification of the configuration of the 6NET core was achieved by a series of small tests. These tests were the following:

4.1.1 Marking test

The objective was to validate that the marking mechanisms were correctly implemented at the access and core routers. Using the *iperf* traffic generator, we created a flow of unmarked packets from the testing PC at Athens (GRNET) to the one at London, (University of Lancaster). By examining received traffic, we were able to verify that packets were marked in Athens' router with the proper IPP DSCP value (decimal 46), while they were traversing the backbone network. On the contrary, packets that were sent by the Netherlands testing PC arrived with a DSCP value of 0, verifying that they were treated as best-effort traffic and we could therefore use that flow for simulating background traffic.

We then artificially congested the network by injecting 200 Mbps of background UDP traffic in the network, while simultaneously sending IP Premium traffic. As Table 3 shows, premium marked traffic was able to pass through the congested links almost without any losses while, in total, there was a very high packet loss in the transmitted traffic. As shown in the last column of the Table 3, almost half of the transmitted background traffic packets were lost.

According to the results, packet loss did not reach the ideal (200-155%/200) (transmission rate – physical link's capacity / transmission rate) because this rate could only be achieved at an ideal byte level. In our case however, a lost packet means that all of its bytes were lost (even if part of the packet could have been delivered). Had the experiment been conducted with multiple UDP flows, the higher aggregation level could have led to a higher achieved bandwidth. However, with our current experiment setup (1 UDP flow), we believe the reported result is reasonable.

	Achieved bandwidth (Mbps)	**Jitter (ms)**	**Packet loss (%)**
UDP foreground	5.11	4.1780	0.082
UDP background	105.00	0.1430	49.000

Table 3: Comparing premium to best-effort traffic under congestion

The above results verify that the prioritization mechanism was correctly treating marked packets with higher priority compared to unmarked or improperly marked packets. The packet loss in the foreground traffic is significantly low but it is not zero, as it was ideally expected. This was probably caused by the PC generators themselves (the measurements came through the *iperf* traffic generator) but, in any case, the packet loss is too low to cause any considerable problem.

4.1.2 Traffic Policing test

The objective of the traffic policing test was to validate that policing mechanisms are performing as expected at input interfaces. In general, input policing takes place at the upstream providers edge routers towards the customers direction. The applied configuration specified that excessive packets were to be dropped (and not demoted to best-effort traffic). The rate limit was configured at 5% of the total capacity of the backbone links, which corresponds to around 7.5Mbps of traffic at the physical level. The traffic policing test did in fact verify that when trying to send 10Mbps of UDP foreground traffic from the GRNET testing PC to the UK testing PC, only 7.10 Mbps of actual traffic got through. Packet loss was at 29%, and average jitter at 0.273ms. The operation of the policing mechanism was further verified during the scenarios which are described in section 4.2.

4.1.3 Shaping test

The objective of the shaping test was to validate that the shaping mechanism is functioning as expected at the output interfaces of the NREN’s access routers and at the input interfaces of the 6NET core routers. In general, output shaping takes place at the customer's side, preferably as close as possible to the source. Input shaping is performed by the upstream provider as an additional service to his/her customers, usually when customers are not able to perform output shaping in their routers. We therefore applied shaping at the access router and run this test to verify that the traffic forwarded to the core domain remained within the bandwidth limit.

For this test we used the *mgen* traffic generator to generate bursty UDP traffic at the GRNET testing PC destined for a receiver at the UK testing PC. The *mgen* generator was configured so that it was starting and stopping sending traffic every 5 seconds (5 seconds of constant traffic were followed by 5 seconds of pause etc.). During the 5-second sending intervals, *mgen* was sending 5000 packets of 1Kbyte in a period of 5 seconds for an average rate of about 8Mbps.

The shaping mechanism smoothed the transmission rate and the traffic was not policed by the core routers. As a result, the sink point of the experiment received traffic that always remained under the rate limit.

4.2 Scenarios

The setup for each scenario is displayed in Table 4. They have been designed so that we could investigate the effectiveness and characteristics of the QoS mechanisms implementing the IP Premium service. In order to more closely simulate actual network conditions, the scenarios include combinations of either UDP, TCP or both types of traffic in the background (best-effort traffic).

Also, Table 5 summarizes the results from the above scenarios for the experiments with UDP foreground traffic. The results from Table 5 for scenarios 7 and 9, which are comprised of multiple tests, are also visualized in Figure 4 and Figure 5 respectively.

Scenario	Background	Foreground	Notes
1	50Mbps (30% TCP- 70% UDP)	UDP traffic (1.5Mbps)	
2	50Mbps (30% TCP- 70% UDP)	TCP traffic (1.5Mbps)	
3	80Mbps (30% TCP- 70% UDP)	UDP traffic (1.5Mbps)	
4	80Mbps (30% TCP- 70% UDP)	TCP traffic (1.5Mbps)	
5	120Mbps (30% TCP- 70% UDP)	UDP traffic (1.5Mbps)	
6	120Mbps (30% TCP- 70% UDP)	TCP traffic (1.5Mbps)	
7	80Mbps (30% TCP- 70% UDP)	UDP traffic (1.5Mbps)	And increase it in steps of 0.5Mbps
8	80Mbps (30% TCP- 70% UDP)	TCP traffic (1.5Mbps)	And increase it in steps of 0.5Mbps
9	120Mbps (30% TCP- 70% UDP)	UDP traffic (1.5Mbps)	And increase it in steps of 0.5Mbps
10	120Mbps (30% TCP- 70% UDP)	TCP traffic (1.5Mbps)	And increase it in steps of 0.5Mbps

Table 4: Testing scenarios

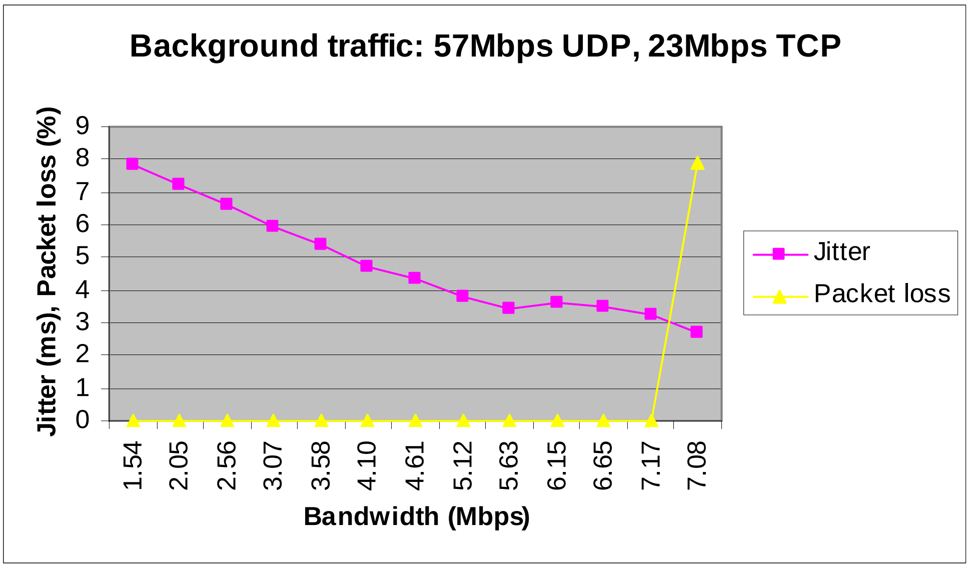

Figure 4: UDP foreground traffic with 80Mbps background traffic

Scenario	Achieved foreground bandwidth (Mbps)	Foreground jitter (ms)	Foreground packet loss (%)
1	1.54	7.813	0.0000
3	1.54	7.810	0.0000
5	1.53	7.808	0.0770
7	1.54	7.814	0.0000
	2.05	7.217	0.0000
	2.56	6.634	0.0000
	3.07	5.927	0.0000
	3.58	5.382	0.0000
	4.10	4.727	0.0000
	4.61	4.348	0.0000
	5.12	3.785	0.0110
	5.63	3.459	0.0000
	6.15	3.629	0.0000
	6.65	3.513	0.0088
	7.17	3.240	0.0082
	7.08	2.717	7.9000
9	1.53	7.799	0.1100
	2.04	7.317	0.1400
	2.56	6.321	0.2300
	3.07	5.579	0.1500
	3.57	5.035	0.3000
	4.08	4.639	0.3600
	4.59	4.368	1.2000
	5.09	4.103	0.5500
	5.60	3.315	0.5400
	6.12	3.340	0.4900
	6.62	3.346	0.6300
	7.11	3.320	0.7900
	7.01	3.221	8.8000

Table 5: Results for UDP foreground traffic

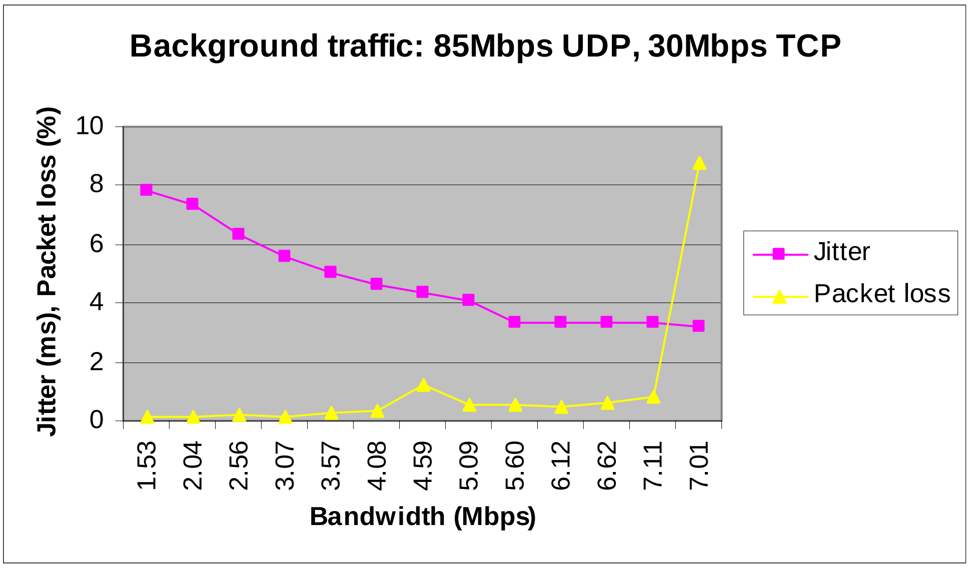

Figure 5: UDP foreground traffic with 120Mbps background traffic

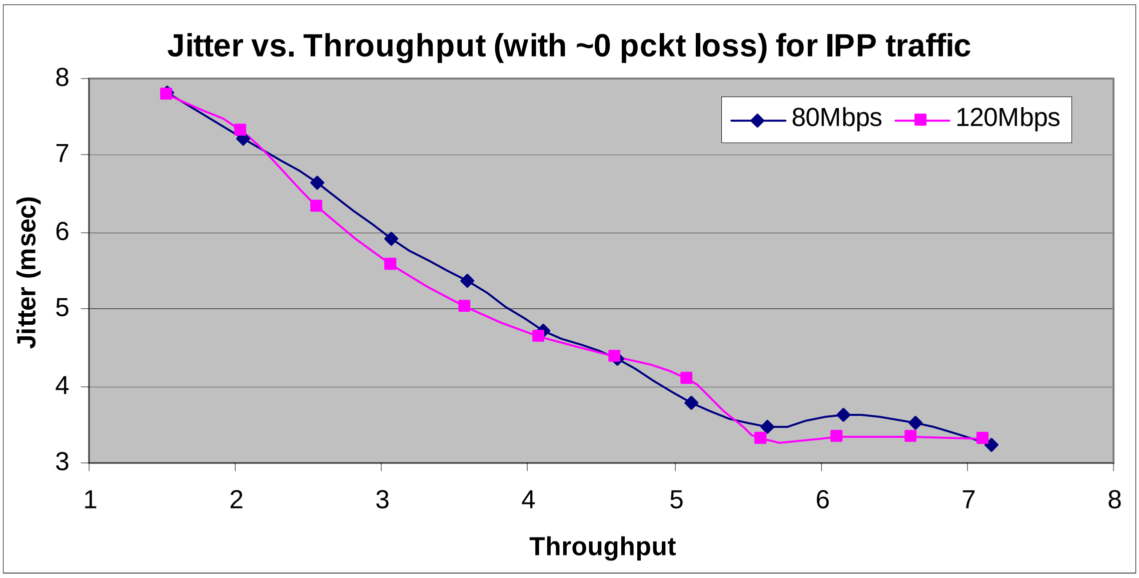

Figure 6: Jitter vs Foreground's throughput

The jitter decreases as throughput gets higher, since packets arrive closer together, and therefore the variation in their inter-arrival times becomes smaller in absolute numbers. The fact that the relation is almost linear implies that the ratio of jitter to throughput remains relatively steady for the different scenarios. Figure 7 shows packet loss vs foreground’s throughput, where the packet loss seems to increase slightly, when the background’s traffic throughput increases. But in any case, the packet loss is quite small to be considered problematic.

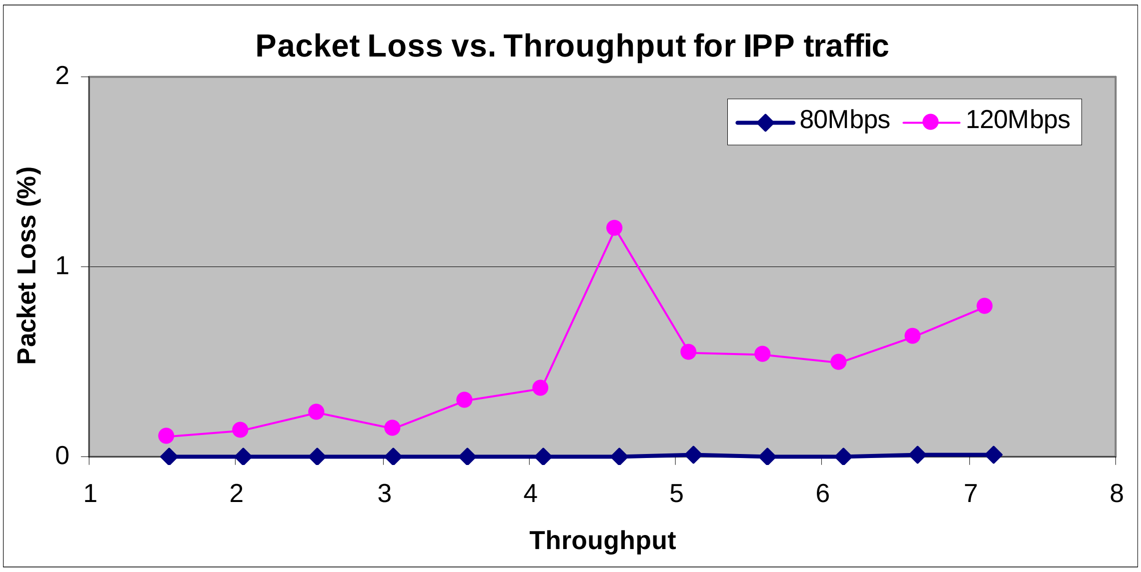

Figure 7: Packet loss vs Foreground's throughput

Figure 4 and Figure 5 clearly demonstrate the effectiveness of the policing mechanism that was applied at the input interface for the premium traffic. The configuration of the policing mechanism was done using the option of completely dropping excess packets (instead of simply treating them as best-effort traffic, which is also a viable solution depending on the requirements and the policies of each organization). Therefore, as soon as foreground traffic exceeded the allocated bandwidth (5% of the total available bandwidth or about 7.5 Mbps at the physical level), packet losses increase dramatically. Our choice for dropping exceeding packets instead of simply handling them as best-effort is more suitable for real-time applications, since for that type of application timing in the reception of the packets matters more than late delivery. In such case, late delivery of packets can be useless if the data should already have been presented to the user.

Another interesting observation is that the jitter for the foreground traffic steadily decreases as we are increasing the transmission rate. This observation is explained by taking into account the way jitter is calculated. A higher transmission rate leads to packets arriving closer together at the destination, and therefore variations in the inter-arrival time are smaller (although in reality they can be steady when weighted against the inter-arrival times).

Table 6 summarizes the results for TCP foreground traffic, which is using the IP Premium service (the TCP window size was set to 256 Kbytes for all experiments) and the corresponding characteristics (jitter, packet loss) for the background traffic that was artificially created for each experiment and is treated as Best-Effort traffic. It is interesting to note that for scenarios 8 and 10, we were gradually increasing the TCP foreground transmission rate by configuring iperf to generate ever more TCP flows. As soon as the transmission rate approached the allocated threshold, the transmission rate could no longer be increased, no matter how many additional TCP streams we were generating, since the policing mechanism was dropping excessive packets and the TCP congestion avoidance mechanism was using this information to reduce the transmission rate.

Scenario	Number of TCP foreground streams	Achieved foreground bandwidth (Mbps)	UDP background jitter (ms)	UDP background packet loss (%)
2	9	1.95	0.439	0.1300
4	6	1.57	0.246	0.0160
6	5	1.52	0.230	0.1100
8	6	1.54	0.320	0.0490
	7	1.97	0.714	0.0450
	8	2.01	0.465	0.1600
	9	2.36	0.234	0.3100
	10	2.47	0.265	0.2200
	11	2.68	0.239	0.2000
	14	3.49	0.221	0.1500
	17	3.85	0.268	0.1400
	20	4.57	0.379	0.0980
	25	5.44	0.219	0.0670
	30	6.07	0.300	0.1200
	35	5.76	0.313	0.2900
10	6	1.54	0.195	1.5000
	7	1.59	0.209	1.1000
	25	4.62	0.148	1.1000
	30	4.90	0.156	1.5000
	35	5.09	0.206	1.4000
	40	5.29	0.170	1.5000
	45	5.03	0.149	1.3000

Table 6: Results for TCP foreground traffic

Finally, Figure 8 compares the number of TCP flows vs the achieved throughput for different background loads. This figure clearly demonstrates that there is a “peak” point where the number of TCP flows (foreground traffic aggregation) is the ideal to achieve the best throughput. This is due to the fact that TCP uses a congestion control mechanism (via the TCP window mechanism) and therefore the level of aggregation of such flows, clearly influence the achieved throughput.

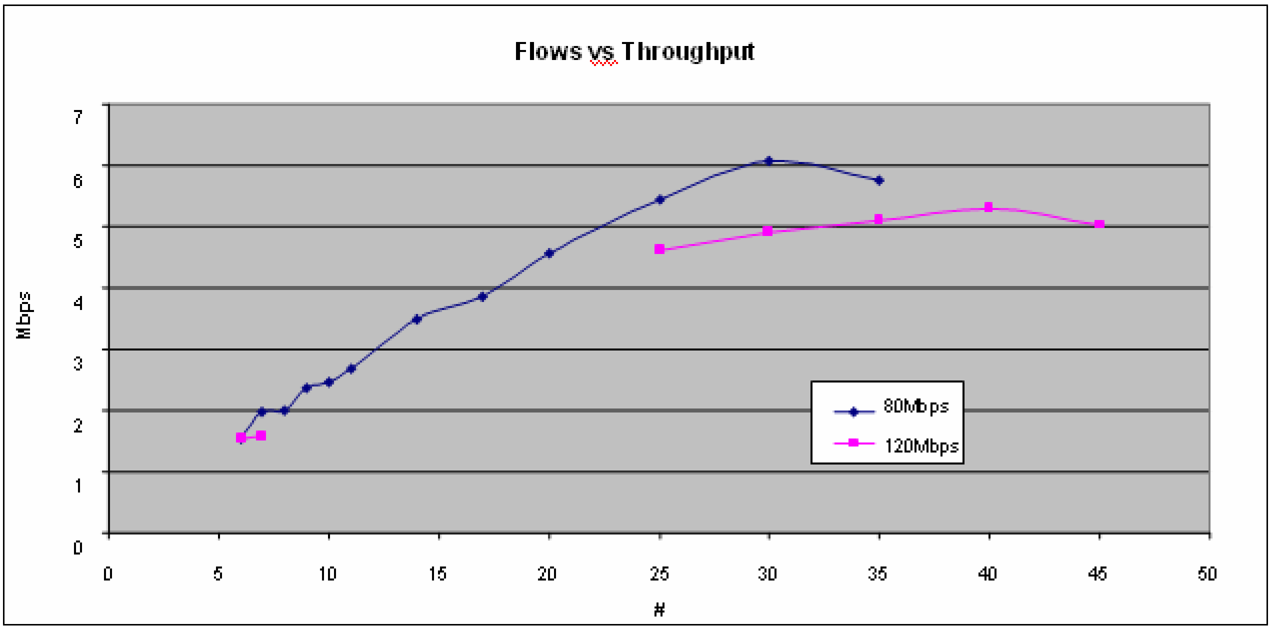

Figure 8: Number of TCP Flows vs achieved throughput

Since the last experiment in scenarios 8 and 10 are the ones that present the most interest (as the foreground traffic approaches the upper bound of the policing profile), Figure 9 through Figure 12 display the variation of the throughput rate from the beginning until the end of each experiment.

In particular, Figure 9 and Figure 10 present the corresponding throughput rates from the last two experiments for scenario 8, while Figure 11 and Figure 12 present the throughput rates from the last two experiments for scenario 10.

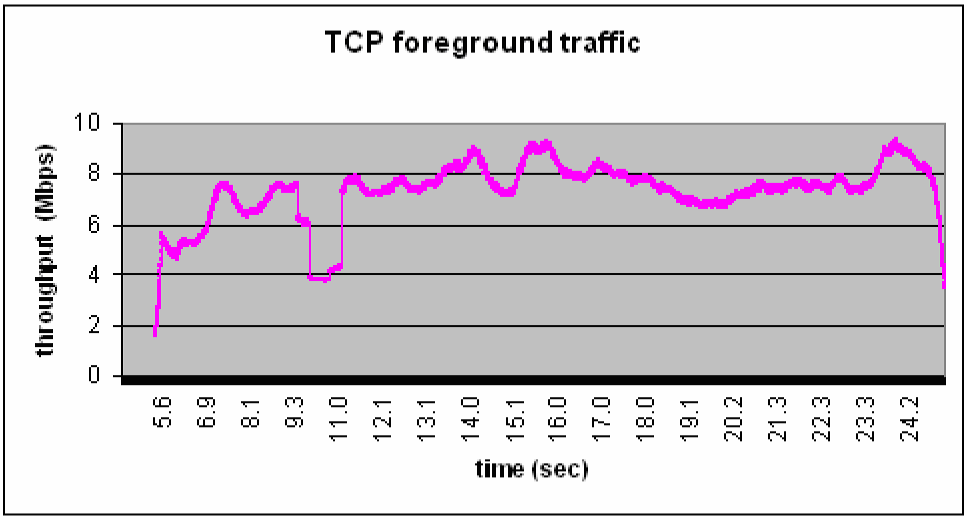

Figure 9: TCP foreground throughput for scenario 8, average throughput 6.07 Mbps (30 TCP streams)

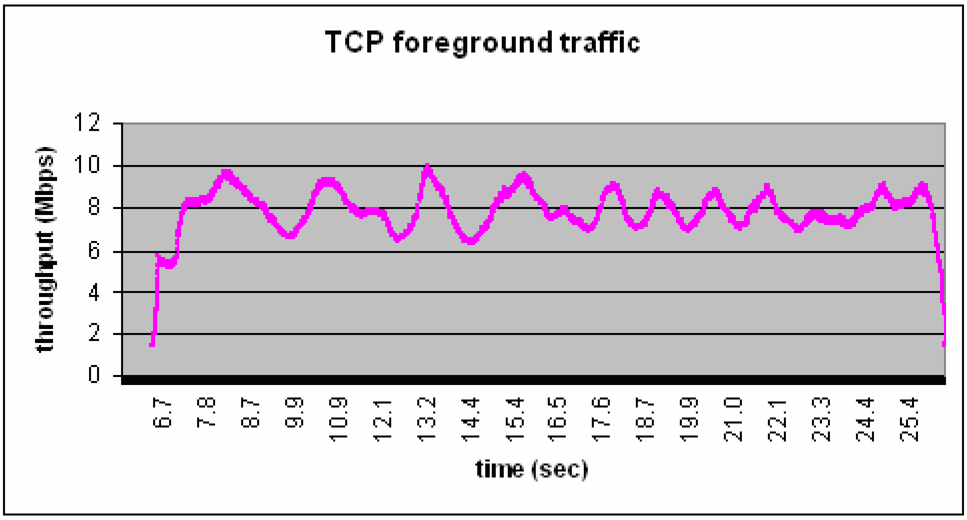

Figure 10: TCP foreground throughput for scenario 8, average throughput 5.76 Mbps (35 TCP streams)

The most interesting observation when comparing Figure 9 and Figure 10 is that in the latter case the throughput seems to vary more regularly, while in Figure 9 the throughput is more stable. The reason is probably the fact that during the second experiment the artificially generated traffic was setup in order to transmit more traffic than the policing limit (through the usage of multiple TCP streams). The result was that the congestion control algorithm of TCP was backing off when it was sensing the packet losses due to the QoS policing mechanism.

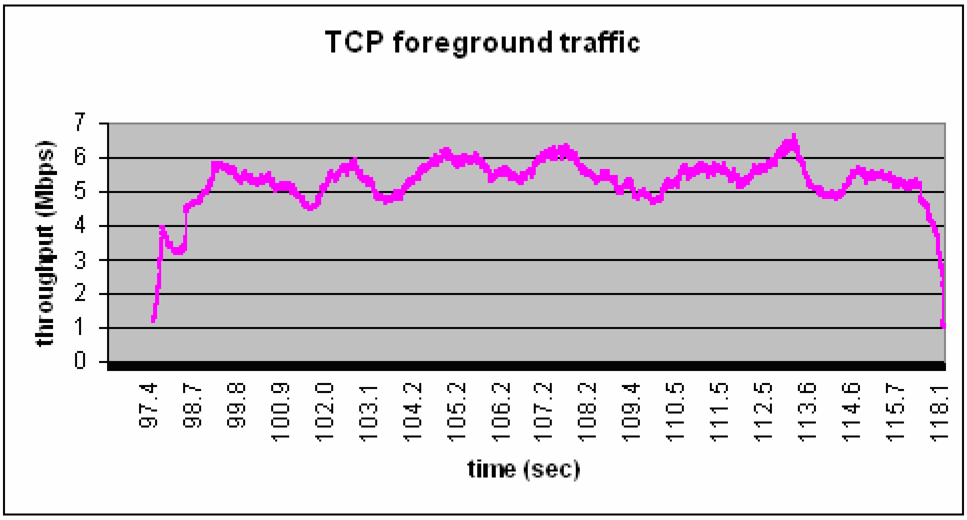

Figure 11: TCP foreground throughput for scenario 10, average throughput 5.29 Mbps (40 TCP streams)

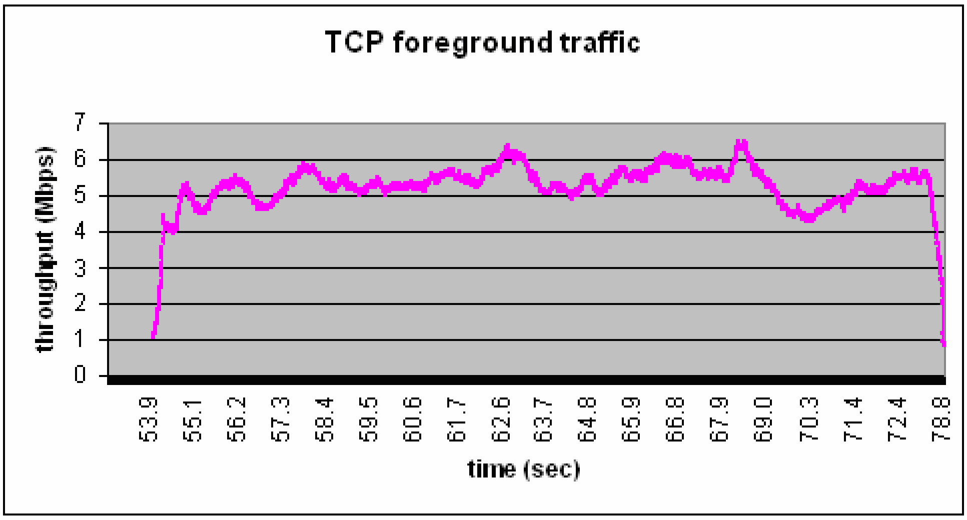

Figure 12: TCP foreground throughput for scenario 10, average throughput 5.03 Mbps (45 TCP streams)

Although in the last two figures (Figure 11 and Figure 12) the background traffic was significantly increased (120 Mbps that almost saturated the core links) the figures demonstrate that the implemented QoS mechanism over IPv6 properly prioritizes the Premium traffic and therefore the sensitive to congestion TCP traffic is unaffected and displays the same behavior as before.

5 QoS tests conducted by 6NET partners in local testbeds

5.1 Tests conducted by ENST using the AF per-hop behaviour

5.1.1 DiffServ-Aware Video Streaming

The term DiffServ-aware video streaming is used here to describe a type of streaming system characterized by the integration of two main components: a DiffServ-enabled IP network and a DiffServ-aware video streaming application.

The general idea behind this concept is that performance may be improved if applications are aware (make use) of the enhanced network capabilities offered by the DiffServ architecture. Thus, a DiffServ-aware application marks its packets in order for them to take advantage of the advanced rate, delay, jitter and/or loss features offered by the network. This approach is called application-driven (or application-controlled) marking, and is different from the typical DiffServ router-based marking which is done by edge routers based on rate-metering or flow identification.

The compressed streams generated by video codecs are usually organized in a hierarchical structure. For example, at the frame level, an MPEG-compressed video stream contains three types of frames: I, P and B. The loss of each type of frame has a different impact on the visual quality of the received signal. The loss of an I frame has a bigger impact than that of a B frame. If an I frame is lost, in addition to the loss of the frame itself, some neighbouring frames can also be considered lost for practical reasons. This is due to the fact that they cannot be properly decoded even if they are correctly received, because the reference information from the lost I frame is not available. One can speak then of error propagation.

In the case of layered coding, a video stream is made up of one base layer and one or more enhancement layers. The base layer represents the video sequence with the lowest quality but also the lowest required bit-rate. Each successive enhancement layer adds up quality and bit-rate. In a multicast environment, clients with low access rates may ask only for the base layer, while clients with higher access rates may additionally ask for as many available enhancements layers as their links can accept.

In any of the cases above, a DiffServ-aware video application marks the outgoing packets taking into account the type of video information they carry. This way, the differentiated treatment inside the DiffServ network should yield a better visual quality by assuring a particular rate, limiting delay and jitter, or discarding first less important packets when congestion arises. This type of marking has also been called semantic marking [14]. Note that when we speak of DiffServ-aware streaming, we refer to methods in which a video stream is somehow separated in "segments" (layers, frame types, partitions, etc.) that are marked and forwarded using different DiffServ classes or drop priorities. This means that we do not regard as DiffServ-aware streaming the case in which a complete video flow is isolated and transported in a single class (like a premium class, for instance).

5.1.2 The DSv6 system

5.1.2.1 Mapping strategy and architecture

Figure 13 shows the DiffServ mapping strategy behind the implemented IPv6 DiffServ-aware video streaming. It's a straightforward mapping that consists of forwarding an MPEG-coded stream in a single three-precedence AF class where the discard priorities (the *colors*) of each packet are set according to the type of frame they carry. This way, packets carrying I frames are marked green, packets carrying P frames are marked yellow and packets carrying B frames are marked red. The goal of this mapping strategy is to protect the reference frames, highly relevant for quality, under heavy network congestion. The prioritized discard of the AF network (enforced by Active Queue Management in the routers) should assure that packets carrying I frames are the last to be discarded if congestion builds up.

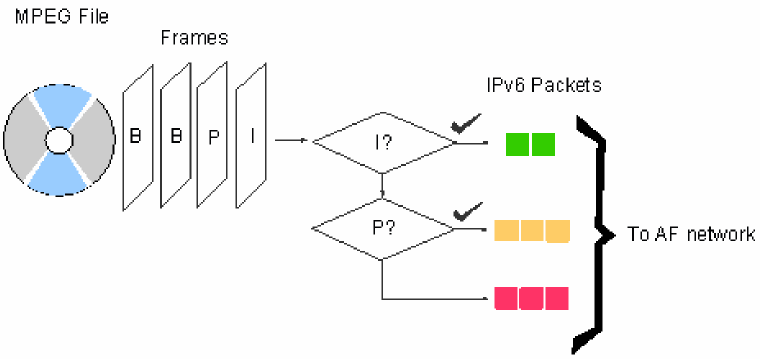

Figure 13: Straightforward AF mapping strategy for MPEG streams

Based on the general mapping strategy described above, the architecture for IPv6 DiffServ-aware streaming would look like the one shown in Figure 14: a video client-server system that sends the MPEG source-marked stream through an IPv6 AF network.

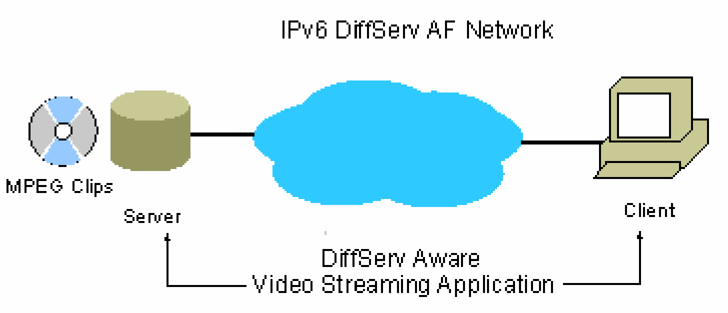

Figure 14: IPv6 DiffServ-aware streaming system

5.1.2.2 Implementation

We have selected the VideoLAN streaming system for the implementation of the DiffServ-aware features because it is free, open source, multi-platform and is already IPv6-enabled. The DS-aware features were added to VLS (Video LAN Server version 0.5.6), the server side of the system. We developed a parser for MPEG-2 Transport Streams (TS) that identifies the frame type carried in the transport packets and assigns them the corresponding priority. This development was done in a station running FreeBSD version 4.8 with the Kame IPv6 snapshot stack. The snapshot (a more experimental version of Kame) is required because the *release* version does not implement the API for setting the IPv6 DiffServ Code Point (DSCP) at the socket level. The client is called VLC (Video LAN Client). We used version 0.7.2, which has better error robustness than previous versions.

5.1.3 Tests

5.1.3.1 Specification

The goal of the tests is to verify that the proposed method and implementation of DiffServ-aware streaming over IPv6 AF gives a better visual quality than streaming over Best Effort. To this end, we did the following process: we sent the marked video stream through a congested router. The experiment was first run in Best Effort and then with AF. We repeated both cases (BE and AF) for increasing levels of congestion. The received video streams were captured and shown to human observers who rated their overall quality.

Figure 15 shows the test platform used in these tests. The local IPv6 network is made up of three FreeBSD computers: two end-nodes and an AF router in the middle. Fric, at the left in Figure 15, is the traffic source. It generates both the video stream and the background traffic. The video stream is generated by the DS-aware VLS server described in section 5.1.2.2. The background traffic used to congest the router is generated by the Iperf tool [6], which can measure TCP and UDP throughput.

Monnaie is the router where AF and rate limiting are activated by means of the ALT-Q software layer [11]. ALT-Q (ALTernate Queuing) is a software framework for BSD that provides queuing disciplines (schedulers, AQM mechanisms) and other components required for QoS networking. ALTQ supports IPv6 and is fully integrated with the Kame stack. In our case, ALT-Q version 3.1 is used to support several functions. In the Best Effort tests, it simply does rate limiting at the output interface by means of a token bucket. This way, the 100 Mbit/s bandwidth from the Fast Ethernet link is reduced to 1 Mbit/s in order to create congestion on the link. In the AF tests, an ingress conditioner classifies the incoming flows: it lets pass the source-marked video stream and it marks the background traffic with the lowest priority (red). At the egress, apart from rate limiting, RIO (RED with In and Out) enforces the differentiated discard of marked packets.

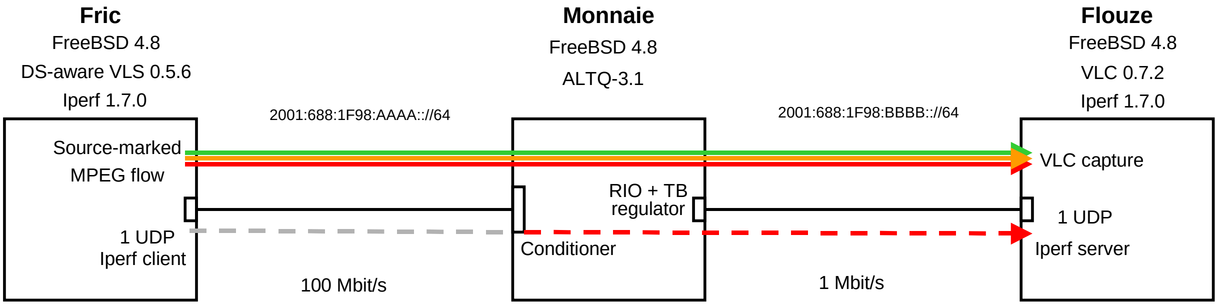

Figure 15: Testbed for EF tests

Finally, the Flouze station acts as the traffic sink. It receives both the video stream and the background traffic, by way of the VLC video client and Iperf, respectively.

A total of six scenarios were tested, three congestion levels for both Best Effort and AF, respectively. The congestion levels are set as a total network load of rate *L*, composed of the video stream (of rate *R*) and the background traffic (of rate *B*). Thus:

$$L = R + B$$

Given the bottleneck's capacity C = 1 Mbit/s , the desired values for L were:

(a) $L1 = 1.1 * C = 1.1$ Mbit/s

(b) $L2 = 1.3 * C = 1.3$ Mbit/s

(c) $L3 = 1.5 * C = 1.5$ Mbit/s

We chose an MPEG-2 video sequence (hipo.mpg), with a duration of 62 s and a bitrate R = 650 kbit/s. Given the values of *R* and *L*, the required values for *B* are then:

(a) $B1 = L1 – R = 1.1 – 0.65 = 0.45$ Mbit/s = 450 kbit/s

(b) $B2 = L2 – R = 1.3 – 0.65 = 0.65$ Mbit/s = 650 kbit/s

(c) $B3 = L3 – R = 1.5 – 0.65 = 0.85$ Mbit/s = 850 kbit/s

Table 7 summarizes the six test scenarios.

Scenario	Network Load *L*	Forwarding
1	1.1 * *C*	BE
2		AF
3	1.3 * *C*	BE
4		AF
5	1.5 * *C*	BE
6		AF

Table 7: Scenarios for AF tests

5.1.3.2 AF configuration

The configuration for the AF scenarios was done in the following way: a single AF queue was enabled in the router output interface by means of the RIO mechanism with the DSCPs presented below:

Green = 0x28

Yellow = 0x30

Red = 0x38

At the ingress interface, a conditioner marks all the background traffic as *Red* and lets the video flow pass with the mark set by the server. The thresholds of the RIO algorithm were configured in terms of the queue size (30 packets). The values of these and all the parameters are shown in Table 8.

Color	Thresholds				maxp	wq
	(as % of queue size)		In packets			
	Min	max	min	max		
Red	0.10 Q	0.25 Q	3	8	0.2	0.008
Yellow	0.20 Q	0.40 Q	6	12	0.1	0.008
Green	0.35 Q	0.65 Q	11	18	0.001	0.008

Table 8: RIO parameters

5.1.3.3 Subjective video quality assessment

The received streams for all six tests were recorded by the VLC client, so that they could be evaluated subjectively by a group of 10 human observers. This subjective assessment was based on the Double Stimulus Impairment Scale (DSIS) standard [18], in which the assessor is first presented with an unimpaired reference of the video sequence, then he/she is shown an impaired version. The assessor rates the impairment of the second sequence with respect to the reference using the 5 point grading scale shown in Table 9.

Grade	Description
5	Imperceptible
4	Perceptible, but not annoying
3	Slightly annoying
2	Annoying
1	Very annoying

Table 9: Impairment scale

5.1.3.4 Results

The 10 individual grades for each one of the six scenarios were averaged in order to get a MOS (Mean opinion Score). The results are shown in Table 10 and Figure 16. Clearly, our DiffServ-aware proposal yields a better perceived quality, as indicated by the higher MOS observed throughout the tests.

Network Load L	**MOS**	
	BE	**AF**
1.1 *C*	2.75	3.19
1.3 *C*	1.88	2.50
1.5 *C*	1.13	2.25

Table 10: MOS by test scenario

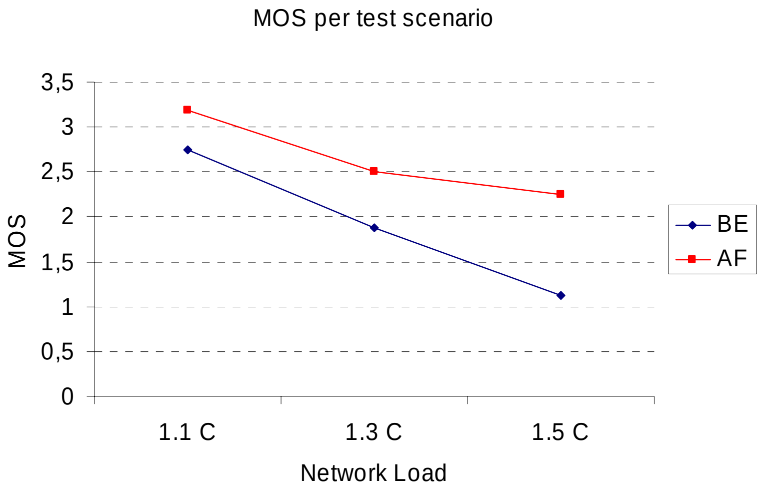

Figure 16: MOS by test scenario

The gains in perceived quality increase with the level of congestion. Table 11 shows the percentage gains in MOS obtained by AF over Best Effort for each network load, which go from 16% for the lowest load to almost 100% for the highest load.

Network Load L	**MOS gain (%)**
1.1 *C*	16,00
1.3 *C*	32,98
1.5 *C*	99,12

Table 11: MOS gains of DiffServ-aware streaming over Best Effort

6 IPv6 Flow Label Usage

The Flow Label is a field in the IPv6 header that has been planned to operate for per flow QoS treatment. General rules for the Flow Label field were proposed recently in RFC 3697 [3], but specific use cases have not been described yet. RFC 3595 [2] defines textual conventions to represent the Flow Label field. Actually the flow label tries to integrate the classic Diffserv operation where traffic is aggregated into classes with the flow establishment. Therefore RFC 3697 defines that the 20-bit Flow Label field is used by a source to label packets of a flow and the zero value is used to indicate packets that are not part of any flow. Packet classifiers use the triplet of Flow Label, Source Address, and Destination Address fields to identify which flow a particular packet belongs to. Packets are processed in a flow-specific manner by the nodes (routers) that have been set up with that flow-specific state. In all cases the Flow Label value set by the source must be delivered unchanged to the destination node. If an IPv6 node is not providing flow-specific treatment, it must ignore the field when receiving or forwarding a packet. Each established flow should expire when the flow is idle in order to increase router performance. Therefore, the RFC defines that nodes should not assume that packets arriving 120 seconds or more after the previous packet of a flow still belong to the same flow (unless a flow state establishment method defines a longer flow state lifetime or the flow state has been explicitly refreshed). Also, this flow expiration allows the re-use of Flow Label values. Flow Label values previously used with a specific pair of source and destination addresses must not be assigned to new flows with the same address pair within 120 seconds of the termination of the previous flow. Finally, to avoid accidental Flow Label value reuse, the source node should select new Flow Label values in a well-defined sequence (e.g., sequential or pseudo-random) and use an initial value that avoids reuse of recently used Flow Label values each time the system restarts.

In addition, a major issue at the deployment of QoS services that uses the Flow Label field is the security aspect. In particular, it is crucial to avoid theft of service attacks by unauthorized traffic. Such attacks are possible in two ways: by spoofing the Flow Label value (only on valid nodes that uses the correct source address) or by spoofing the whole 3-tuple (Flow Label, Source Address, Destination Address). The latter can be done in an intermediate router or in a host that is not subject to ingress filtering. Also, we should notice that IPsec protocol does not include the IPv6 header's Flow Label in any of its cryptographic calculations (in the case of tunnel mode, it is the outer IPv6 header's Flow Label that is not included). As a consequence IPsec does not provide any defense against an adversary's modification of the Flow Label. Finally it is recommended that applications with an appropriate privilege in a sending host should be entitled to set a non zero Flow Label. But this is an issue that depends on the host’s operating system or on the used policy and authorization mechanisms.

In the near future, the usage of the Flow Label in QoS services and also in related mechanisms is expected. A classic application that can take advantage from Flow Label field is a VoIP implementation that can succeed flow establishment. But in this case, accompanied mechanisms such as admission control and flow signaling protocols are necessary in order to ensure security and successful operation.

7 Conclusions

This document described the report from the second phase of QoS activity in 6NET project. During the first phase, as deliverable D4.4.2v1 described, most of the partners analysed the operation and interaction of QoS mechanisms in local IPv6 testbeds and validated the performance of QoS implementations for different platforms, such as commercial routers from different vendors. Also, a QoS model for 6NET's backbone network was proposed and described in the first deliverable.

The second phase continued the above work by defining the QoS model in detail, implementing and applying it in the 6NET backbone and finally performing large scale IPv6 QoS tests. In particular, the 6NET network became IPv6 QoS enabled and remained operational for about 8 months (until the 6NET network's decommission). A number of tests were defined that aimed to evaluate the performance of the QoS service in the 6NET network. These tests were performed successfully, as described in the previous sections. The tests verified several mechanisms in various network conditions and especially under heavy congestion. They also showed that applying QoS on the 6NET environment, actually gave no performance degradation on the core network's hardware (Cisco 12xxx series routers). The successful completion of these experiments clearly demonstrates that such a service (IPv6 QoS IP Premium) is mature enough in order to be deployed in a production IPv6 network.

This document also described the implementation and testing of an architecture for the DiffServ-aware streaming of MPEG video based on the mapping between the frame types of the video flow and the drop priorities of an AF-enabled IPv6 network. The goal of the proposed architecture is to reduce visual distortion brought about by packet losses in a congested network. The mapping proposal was implemented in the VideoLAN streaming application, tested in a local IPv6 network and evaluated by means of subjective video quality assessment. The results showed that the proposed DiffServ-aware streaming approach yields gains in perceived quality ranging from 16% to 99% over Best Effort streaming, depending on the level of congestion. In a more general scope, the results give evidence that DiffServ-awareness can be used at the application level to increase performance from the *user's perspective* in a DiffServ-enabled IPv6 network infrastructure. The benefits in performance depend on the mapping strategy defined to integrate the semantics of the information at the user level and the traffic classes in the network.

Finally, the IPv6 Flow Label is not yet used in the IPv6 world, but it is expected to be standardized and be more widely used in the near future.

8 References

[1] S. Vegesna, 'IP Quality of Service: the complete resource for understanding and deploying IP quality of service for Cisco networks', Cisco Press, 2001

[2] B. Wijnen, RFC 3595, "Textual Conventions for IPv6 Flow Label", Network Working Group, September 2003

[3] J. Rajahalme, A. Conta, B. Carpenter, S. Deering, RFC 3697, "IPv6 Flow Label Specification", Internet Engineering Task Force, March 2004

[4] http://www.cisco.com

[5] http://ouranos.ceid.upatras.gr/ipv6_qos/default.htm

[6] http://dast.nlanr.net/Projects/Iperf/

[7] http://mgen.pf.itd.nrl.navy.mil/

[8] Ethereal Network Analyzer, http://www.ethereal.com/

[9] C. Bouras, A. Gkamas, D. Primpas, K. Stamos, “Quality of Service aspects in an IPv6 domain”, 2004 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS’ 04), San Jose, California, USA, July 25 - 29 2004, pp. 238 – 245

[10] C. Bouras, A. Gkamas, D. Primpas, K. Stamos, "Performance Evaluation of the Impact of Quality of Service mechanisms in an IPv6 network for IPv6 - capable real time applications", Journal of Network and Systems Management, Kluwer Academic Publishers, Volume 12, Issue 4, December 2004, pp. 463 - 483

[11] http://www.csl.sony.co.jp/person/kjc/kjc/software.html

[12] Implementing QoS for IPv6, Documentation, December 2004, Cisco Systems Inc.

[13] M. A. El-Gendy and K. G. Shin. Assured Forwarding Fairness Using Equation-Based Packet Marking and Packet Separation. Computer Networks The International Journal of Computer and Telecommunications Networking. 41(4): 435-450. March 2003.

[14] O. Medina. Etude des algorithmes d'attribution de priorité dans un Internet à Différentiation de Services. Ph.D. Thesis, Université de Rennes I, Rennes, France, March 2001.

[15] http://www.cnaf.infn.it/~ferrari/tfngn/lbe/results/lbe-geant/

[16] IP Performance Metrics (IPPM) Working Group, http://www.ietf.org/html.charters/ippm-charter.html, IETF.

[17] Service Quality across Independently managed Networks, http://archive.dante.net/sequin/

[18] ITU-R. Methodology for the subjective assessment of the quality of television pictures. Recommendation BT.500.11.2002.

9 Appendix A: QoS configuration in 6NET routers

- class-map EF: used to match EF packets

```
class-map match-all EF
  match dscp 46
```

- class-map EF-FILTERED: used to authorise IP Premium traffic using the access-list nren-auth-ef. This list will contain a list of all IP addresses authorised by participating NRENs to use EF. The marking is done by the 6NET router in the access interface.

```
class-map match-all EF-FILTERED
  match access-group name nren-auth-ef
```

- class-map LBE: used to match LBE traffic

```
class-map match-all LBE
  match dscp 8
```

- class-map OTHER: used to match all traffic except EF and LBE

```
class-map match-all OTHER
  match not dscp 8
  match not dscp 46
```

policy-map NON-PARTICIPANT: used to remark all dscp to 0 from NREN connections who are not participating in QoS testing. This should be applied as an inbound service-policy to ALL non-participant NREN interfaces and all external/peering interfaces. Initially, this is all interfaces except DFN, GRnet, JANET, NORDUnet and Renater

```
policy-map NON-PARTICIPANT
  class class-default
   set dscp 0
```

policy-map TRUSTED-IN: used to match and police packets in the trusted (it is applied with the condition that NREN does marking) tests. It polices all the EF traffic to 5% of interface's bandwidth and the exceeded traffic is dropped. The policing is done in an aggregated basis to all the incoming marked traffic. Also, this policy map guarantees 1% of interface's bandwidth for LBE. This will initially be applied to access interfaces of DFN, GRnet, JANET, NORDUnet and Renater

```
policy-map TRUSTED-IN
  class EF
   police cir percent 5 conform-action transmit exceed-action drop
  class LBE
   police cir percent 1 conform-action transmit exceed-action transmit
  class class-default
   set dscp 0
```

policy-map FILTERED-IN: used to match and police packets where the 6net core verifies DSCP markings are valid (against a list supplied by an NREN in advance) and remarks anything not authorised to BE. LBE is treated the same as for trusted.

```
policy-map FILTERED-IN
  class EF-FILTERED
   police cir percent 5 conform-action transmit exceed-action drop
   set dscp 46
  class LBE
   police cir percent 1 conform-action transmit exceed-action transmit
  class class-default
   set dscp 0
```

policy-map my_policy: For the core links it will be required to create two queues, one priority and one other queues. This will need to be applied on the CORE facing interfaces on all 12Ks

```
policy-map my_policy
  class EF
    priority
  class class-default
    random-detect dscp-based
```

The Service policy will need to be appended to the access ingress-interface.

```
interface POSx/x
service-policy input <the selected policy>
```

Next the policy map my_policy must be added to all CORE POS interfaces:

```
interface POSx/x
service-policy output my_policy
```