Available online at www.sciencedirect.com
SCIENCE DIRECT
Computer Communications 29 (2006) 1393–1401
computer communications
www.elsevier.com/locate/comcom

IPv6 deployment: Real time applications and QoS aspects☆

C. Bouras [a,b,*], A. Gkamas [a,b,1,3], D. Primpas [a,b,2,3], K. Stamos [a,b,2,3]

[a] Research Academic Computer Technology Institute, D. Maritsas Building, N. Kazantzaki str., University of Patras Campus, GR-26500 Rion, Greece
[b] Computer Engineering and Informatics Department, University of Patras, GR-26500 Patras, Greece

Received 21 August 2005; accepted 24 August 2005
Available online 12 October 2005

Abstract

This paper gives an overview of the issues related to the QoS mechanisms under IPv6 and the transition of applications to the new Internet protocol. We describe the implementation and testing of a QoS service on IPv6 networks, which is based on the DiffServ architecture (expedited forwarding) and provides strict priorities to packets that are produced from real time applications. The service was implemented and tested at the 6NET large-scale IPv6 network, and additional advanced testing was performed in a local testbed. We present and analyze the results from our tests under a number of different scenarios. In addition, we focus on issues regarding application transition to IPv6 and we briefly discuss as a case study the transition of OpenH323 protocol stack to support IPv6. We also discuss the usage and the expected impact of the new Flow Label field in the IPv6 header.

Keywords: IPv6; QoS; 6NET; Real-time applications; Dual-stack; Flow label

1. Introduction

In order to address the limited address space of IPv4 and other concerns regarding its age and its ability to support the future needs for the Internet, the Internet Engineering Task Force (IETF) has developed a suite of protocols and standards known as IP version 6 (IPv6) [1]. The design of IPv6 is intentionally targeted for minimal impact on upper and lower layer protocols by avoiding the random addition of new features. More than simply increasing the address space, IPv6 offers improvements like built-in security support, plug and play support, no checksum at the IP header and more flexibility and extensibility than IPv4. IPv6 also facilitates efficient renumbering of sites by explicitly supporting multiple addresses on an interface. The widespread adoption of the new Internet Protocol will fuel innovation and make possible the creation of many new networking applications. It will also allow the replacement of the NAT solutions that have been implemented today in order to work around the lack of IPv4 addresses. NAT introduces a number of problems to network applications that need knowledge of the IP address of the host machine or want to take advantage of Quality of Service mechanisms, like VoIP implementations.

Internet traffic consists of flows generated by different applications, which all typically receive the same treatment from the network, as the network can only provide best-effort service. This treatment causes many problems to real time applications, because they are sensitive on parameters as delay, packet loss or jitter. The implementation of QoS techniques is therefore necessary and their usage is widely experimented. During the last years, two architectures—IntServ and DiffServ—have been proposed in order to provide QoS and some services have already been deployed. Until now, significant research activity has been done on QoS in IPv4 networks, but QoS in IPv6 networks is still a relatively open issue. One of the next goals in this area, and the topic that we examine in this paper is the investigation of the deployment of a QoS service on an IPv6 domain. Since the usage of the IPv6 protocol increases and many domains have been IPv6 enabled or are expected to become so in the next few years, the investigation of the supported QoS mechanisms on IPv6 and the deployment of a QoS service should be a basic goal for

☆ This work was partially supported by the 6NET project funded by the IST program of the European Commission (IST contract No: 2001-32603).

* Corresponding author. Address: Computer Engineering and Informatics Department, University of Patras, GR-26500 Patras, Greece. Tel: +30 2610 960375; fax: +30 2610 969016.

E-mail addresses: bouras@cti.gr (C. Bouras), gkamas@cti.gr (A. Gkamas), primpas@cti.gr (D. Primpas), stamos@cti.gr (K. Stamos).

[1] Tel.: +30 2610 960465.
[2] Tel.: +30 2610 960316.
[3] Fax: +30 2610 960358.

0140-3664/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2005.08.014

existing networks as they migrate to tomorrow's IPv6 networks.

In order to evaluate the QoS mechanisms in IPv6, we implemented a QoS service at the large-scale 6NET network, and also at a local testbed especially setup for this purpose. For real-time traffic, we used the traffic generated by an H.323 videoconferencing application based on the library developed by the OpenH323 project. The OpenH323 project [2] develops an open source implementation of the H.323 standard in the form of a central library, the OpenH323 library, which is also based on another open source library called PWLib. The OpenH323 library can be used for the rapid development of applications that wish to use the H.323 protocol for multimedia communications over packet-based networks. The library and most of the applications it supports have now been ported to IPv6 [3,4]. Although the SIP protocol has recently gained wide adoption, H.323 is still very important as there is a very large installed base of H.323 products and it is still used widely (for example GnomeMeeting, an H.323 client based on OpenH323 is the most important videoconferencing application in the Linux environment).

Part of this work has been supported by the 6NET project [5], in the framework of which the native IPv6 network used for our experiments was installed, configured and maintained. 6NET is a 3-year European project to demonstrate that continued growth of the Internet can be met using new IPv6 technology. The project has built a native IPv6-based network connecting 16 countries in order to gain experience of IPv6 deployment and migration from existing IPv4-based networks. This has been used to extensively test a variety of new IPv6 services and applications, as well as interoperability with legacy applications. The 6NET network is Europe's biggest IPv6 network, and aims at gaining and developing an understanding of the issues involved in deploying IPv6 networks, in terms of physical infrastructure, address allocation, registries, routing, DNS operation, and network management among others.

The rest of the paper is organized as follows: Section 2 presents related work on IPv6, while Section 3 presents issues regarding real time applications and IPv6. In Section 4, we analyze QoS aspects in IPv6 by presenting some QoS experiments over IPv6 infrastructure. Finally, Section 5 describes the future work that we intend to do on this area and Section 6 presents our conclusions.

2. Related work

IPv6 has generated a lot of interest in the research community and as IPv6 deployment is steadily becoming more widespread, more and more affected areas of the network infrastructure are examined. There is very large literature on the general aspects of the new Internet Protocol and its benefits [6,7]. The topic of applications operating over IPv6 is also extensively covered and a number of approaches have also been proposed in order to port already existing network applications to IPv6 [8–11]. However, an area that relatively lacks sizeable literature is the issue of Quality of Service mechanisms and their operation over the IPv6 protocol, particularly in real-world scenarios. Of course the issue of Quality of Service in general is a very active research area [12]. In earlier publications [13] the implementation of QoS mechanisms using Cisco network devices has been extensively investigated using localized testbeds. This research is followed up and expanded by the experiments covered in the current paper. Although IPv6 DiffServ has been implemented in many commercial routers for some time now, literature is still lacking in terms of large-scale tests in native IPv6 networks, and this paper intends to cover this aspect.

Besides 6NET, a number of other research projects [14–18] in the European-Union and other parts of the world are also pushing the boundaries of IPv6 deployment and large-scale experience of the protocol. 6QM [17] is specifically devoted to research and development of measurement technologies for Quality of Service in IPv6 networks.

3. Applications for IPv6

Programming network applications for IPv6 [19] introduces a number of changes compared to IPv4. This means that most of the existing network applications cannot normally make use of an IPv6-enabled network. There have been several approaches to solve this problem, such as Bump-in-the-Stack [9], Bump-in-the-API [10], IPv6-to-IPv4 Transport Relay Translator [11] or Stateless IP/ICMP Translation Algorithm (SIIT) [8], which are generally useful for the transition period and cover the need for interoperation between nodes that are at different IPv4 to IPv6 transition stages.

However, it is ultimately inevitable to have full IPv6-enabled applications, in order to make full use of the new Internet Protocol and support its wide adoption. An application that makes use of IPv6 can immediately benefit from the largest obstacle that IPv4 presents, namely the scarcity of addresses and therefore the proliferation of complex mechanisms, like NAT, that attempt to alleviate this problem. There are a lot of applications, with VoIP applications being a notable example, which experience serious difficulties when tried to be setup in a NAT environment. By porting such applications to IPv6, the huge supply of fully-functional, routable addresses easily solves all the NAT-related problems.

Moreover, the IPv6-enabled application will be ready to utilize new IPv6 features such as the Flow Label field and the emerging extensions at the Internet Layer, such as IPsec and Quality of Service (QoS) support.

3.1. IPv4–IPv6 comparative application evaluation

In order to have a reference point for the rest of the experiments we initially compared the IPv4 and IPv6 stacks used by the OpenPhone application based on OpenH323 which has been IPv6-enabled, without any congestion or QoS mechanism.

The first experiment was to use the IPv4 version of the OpenPhone application at a Point-to-Point communication, sending video and audio between 2 PCs, and without any

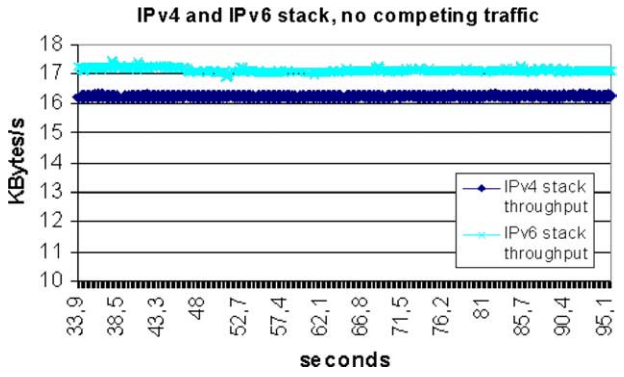

Fig. 1. IPv4—IPv6 comparison.

competing traffic at the intermediate link. As shown at Fig. 1, we obtained a steady transmit rate of 16 kb/s throughout the experiment. The quality of the video transmitted was relatively low, because of the characteristics of the H.261 codec. We then repeated the experiment using the IPv6 stack for the Point-to-Point communication between the two endpoints. Again, we were sending video and audio between 2 PCs, and without any competing traffic at the intermediate link.

Again we can see in Fig. 1 that in the absence of any competing traffic and with a link of high capacity, we obtain a steady transmission rate of around 17 kb/s, around 7% larger than the IPv4 transmission rate. This difference is due to the fact that the Data-Link layer was carrying 294-byte packets in the case of IPv4, and 314-byte packets in the case of IPv6. The standard IPv6 header is 20 bytes larger than the standard IPv4 header, which produces the 7% overhead. This is in fact an expected and known result, since the larger IPv6 header introduces some overhead, especially in relatively low-rate transmissions.

In both cases, we can observe that the choice of network layer stack is not an issue, since the application will consume the required bandwidth, given an uncongested link. We cannot however expect that this will always be the case. A transmission rate of around 140 kb/s, means that for low bandwidth links (for example modem or basic ISDN links) there will be significant congestion. Also for high bandwidth links that carry a lot of additional traffic, unwanted results can occur if the H.323 traffic is added to the competition.

4. QoS aspects in IPv6

Our purpose in this section is to describe the results of our experiments with QoS mechanisms based on the DiffServ architecture in IPv6 networks. Moreover, we examine the issue of the usage of the Flow Label field in the IPv6 header, and the ongoing work in this area.

4.1. IPv6 QoS testbed implementation

In this section, we describe the implementation and testing of a QoS service on IPv6 networks. The service is based on the DiffServ architecture (expedited forwarding) and provides strict priorities to packets that are produced from real time applications. This service was applied throughout the pan-European 6NET network backbone, which is displayed in Fig. 2. Also Fig. 3 displays in more detail the part of the testbed that was used for the experiments. The QoS service that was implemented aims at providing prioritization for traffic coming from real time applications. Its operation is to classify the packets that belong to this application and use a 'priority queue' for them. The rest of the traffic on the router will be treated as usual, with best-effort service.

The service has been implemented using the Modified Deficit Round Robin (MDRR) mechanism and follows the classic guidelines of the DiffServ architecture. For the experimental scenarios we have used the Iperf [20] traffic generator for artificial traffic. Background traffic is classified with DSCP value 0 (default) and is treated as best-effort. In addition, we inserted foreground traffic that simulates an aggregate of real time traffic. This traffic, depending on the specific test, was either TCP or UDP traffic, or a mix of artificially generated UDP traffic and RTP traffic [21] generated by an application based on the OpenH323 library, which has been ported to IPv6. On the network devices, a marking mechanism has been applied in order to mark the background and foreground traffic. In particular, we distinguish the background and foreground traffic with different access lists and mark them with different DSCP values: default (0) and the predefined expedited forwarding (46), respectively. Next, the output interfaces of the network devices have been configured in order to send the packets that have been marked with DSCP 46 with strict priority.

The various phases of the experiments described in this section can be categorized as follows:

- Various scenarios using a mix of artificially generated UDP and TCP traffic for background and either TCP or UDP artificially generated traffic for foreground, using the large-scale 6NET testbed.
- Experiments using a mix of artificially generated UDP and TCP traffic for background and a mix of artificially generated UDP and real-time traffic using the local CTI testbed.
- Experiments using the local CTI testbed where the network devices have been additionally configured in order to apply the Weighted Random Early Detection (WRED) mechanism for congestion avoidance on the background traffic. In this case, the goal is to measure if the existence of WRED makes any impact on the QoS guarantees that the foreground packets experience.

For the large-scale scenarios, the marking of the packets was performed at the access level and prioritized traffic was policed at 5% of the total available bandwidth, which corresponds to about 7.5 Mbps for the core links that have a total capacity of 155 Mbps at the physical level.

For the purposes of the experiments we used three PCs located at various points in the network. Premium traffic in the foreground was sourced from UK (JANET network), was routed through the Netherlands and Germany, and was

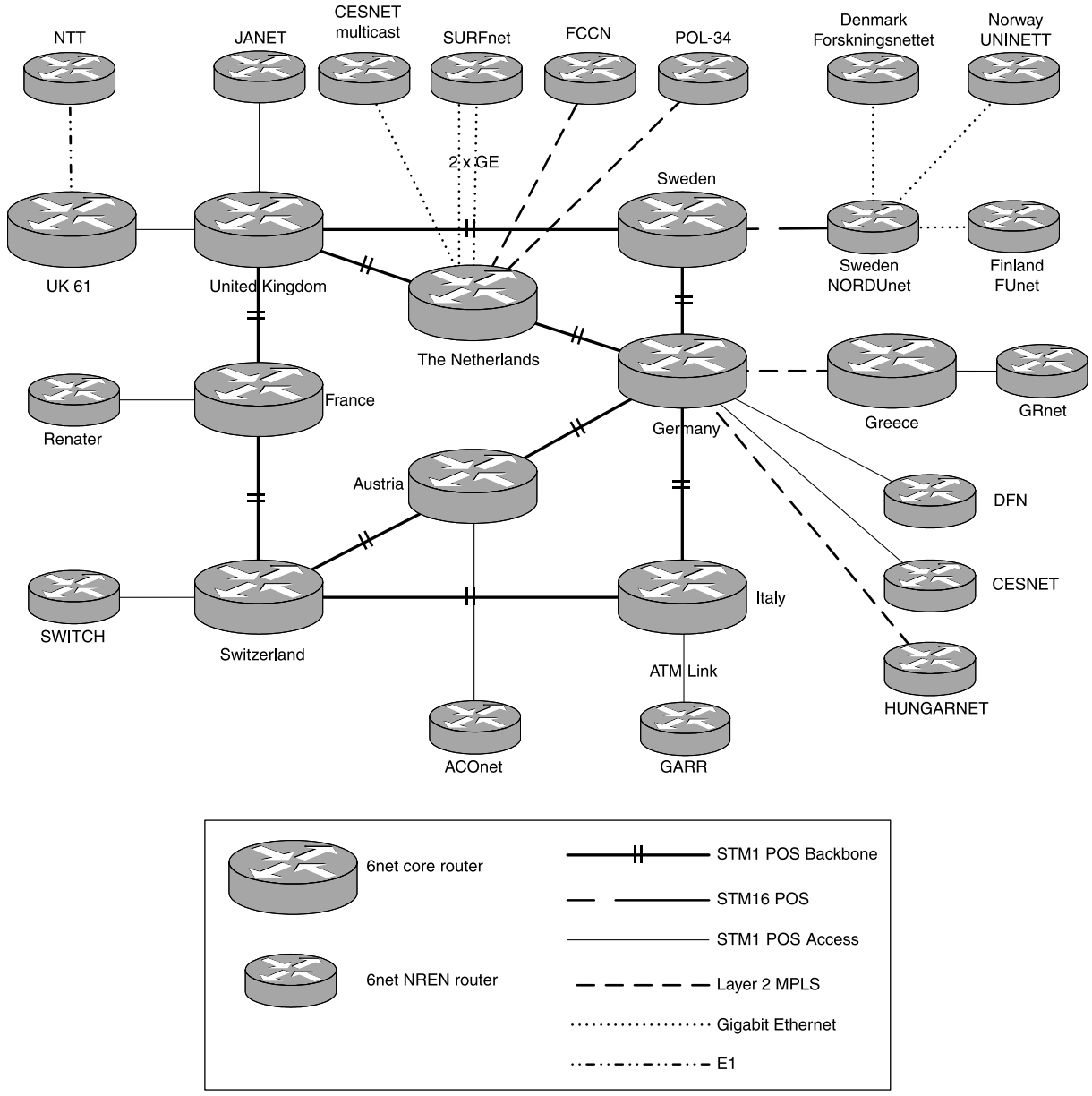

Fig. 2. 6NET core network.

received at Greece. Furthermore, in the experiments that we wanted to artificially create background traffic, a generating PC at the Netherlands was used which sent traffic through Germany to Greece.

The measurement of the network's metrics and the service's performance was done using specific tools. The first one was the statistics of the Iperf traffic generator for the traffic that it was generating. These 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:

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

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

$$D(i,j) = (R_j - R_i) - (S_j - S_i)$$

Another tool was the RTCP statistics that the videoconference tool was providing and finally, the results from the Ethereal Network Protocol Analyzer [22] that captured all the packets at the receiver and provided us graphic representations of their throughput.

Using this infrastructure, we performed many tests investigating the QoS mechanisms that are supported on IPv6. First of all, we tested the prioritization and classification mechanisms performing a large number of tests. Then, these tests became more complex and finally, we applied all the mechanisms simultaneously to test the whole QoS service with real time data.

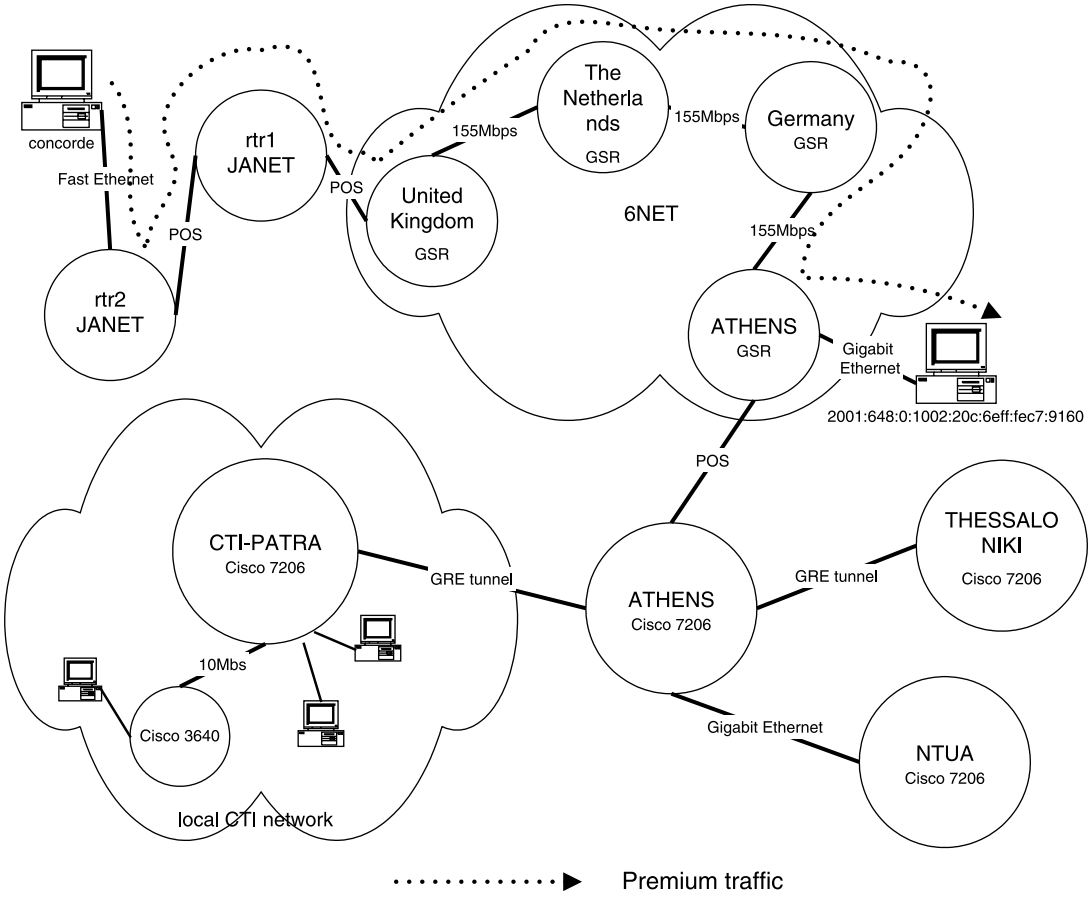

Fig. 3. Large-scale testbed.

4.2. Investigation of the prioritization mechanism

The first tests at the large-scale 6NET testbed aimed at investigating the operation of the classification and prioritization mechanisms. The classification mechanism was implemented using access lists and creating a policing class in the input interface of the router. According to the ipv6 access list that the packets belong to, the policy class assigns the DSCP values: ef (46) for the foreground traffic and default (0) for the background. Next, on the output interface of the router, we configured a second policy class that gives strict priority to the packets that have been marked with the DSCP value for EF.

At a first stage we verified the correct operation of the marking, policing and shaping mechanisms, and then we conducted a number of scenarios that included simultaneous UDP and TCP background traffic, while the foreground traffic was alternated between the two transport protocols. In order to verify that traffic marked with the DSCP value of 46 (IP Premium traffic) was indeed handled preferentially compared to the rest of the traffic, we artificially congested the network by generating 200 Mbps of background UDP traffic. As Table 1 demonstrates, while in general traffic had very large losses (about half of the transmitted background traffic packets were lost), premium marked traffic was able to comfortably traverse the congested links almost without any losses.

Table 1
Comparing premium to best-effort traffic under congestion

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

4.3. Large-scale scenarios

The setup for each scenario is displayed in Table 2. They have been designed so that we could investigate the effectiveness and characteristics of the implemented QoS mechanisms in order to provide a measurable improvement to various types of traffic that has been marked as premium traffic. In order to more closely simulate actual network conditions, scenarios include combinations of either UDP, TCP or both types of traffic at the background (best-effort traffic). We have selected a mix of 30% TCP and 70% UDP for the background traffic, in order to more closely simulate a real-world environment with a large number of bandwidth-consuming UDP applications, and a considerable amount of TCP applications (such as web browsing).

The results from scenarios 7 and 9 which are comprised of multiple tests, are also visualized in Figs. 4 and 5, respectively.

Table 2
Large-scale scenarios

Scenario	Background (30% TCP–70% UDP)	Foreground
1	50 Mbps	UDP traffic (1.5 Mbps)
2	50 Mbps	TCP traffic (1.5 Mbps)
3	80 Mbps	UDP traffic (1.5 Mbps)
4	80 Mbps	TCP traffic (1.5 Mbps)
5	120 Mbps	UDP traffic (1.5 Mbps)
6	120 Mbps	TCP traffic (1.5 Mbps)
7	80 Mbps	UDP traffic > 1.5 Mbps)
8	80 Mbps	TCP traffic > 1.5 Mbps)
9	120 Mbps	UDP traffic > 1.5 Mbps)
10	120 Mbps	TCP traffic > 1.5 Mbps)

Figs. 4 and 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 a 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).

Figs. 6 and 7 summarize the results for TCP foreground traffic for scenarios 8 and 10, respectively, where we were gradually increasing the TCP foreground transmission rate. As soon as the transmission rate approached the allocated

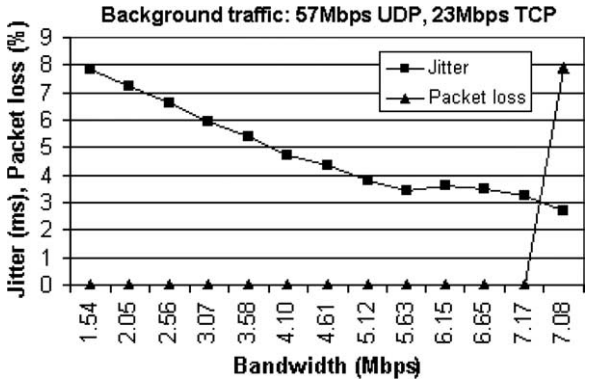

Fig. 4. UDP foreground with 80 Mbps background traffic.

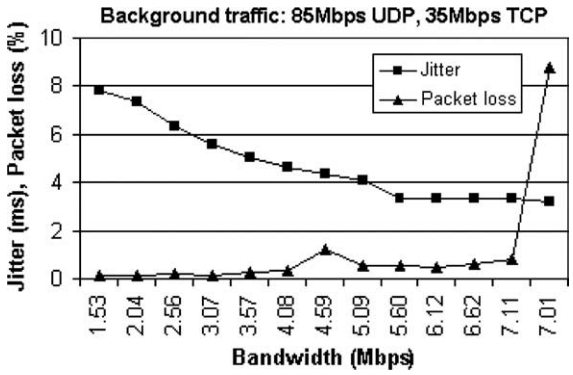

Fig. 5. UDP foreground with 120 Mbps background traffic.

threshold, the transmission rate could no longer be increased, since the policing mechanism was dropping excessive packets and the TCP congestion avoidance mechanism was using this information to reduce the transmission rate.

4.4. Experimental testing for real time applications

Additionally, extended tests in CTI's local testbed were conducted, aiming to evaluate the QoS mechanisms with real-time applications. This testbed uses infrastructure that has been deployed for IST 6NET project extended with additional CTI routers. The basic testbed consisted of two routers, one router of CISCO 7200 series (local 6NET router) and one router of CISCO 3600 series. This testbed is interconnected with CTI's production network and can be shown in Fig. 8 that presents it in detail. The software version that this testbed uses is the CISCO IOS 12.2(13) T [23].

The implemented QoS framework in the local testbed was similar to the one applied at the 6NET network, with the only difference being that the Class-based Weighted Fair Queuing mechanism was used instead of Modified Deficit Round Robin. This was done due to the fact that MDRR is supported in higher end devices which were not available at CTI local testbed. In any case, the Class-based WFQ mechanism can also provide strict packet priority as it implements the Low Latency Queue mechanism, which is a method to en-queue packets on a 'priority' queue in order to guarantee low latency and jitter.

The strict priority (which implements the Low Latency Queues) is extremely suitable for real-time applications that need low delay, packet loss and jitter. Therefore, we tried to simulate realistic conditions of traffic load and to measure

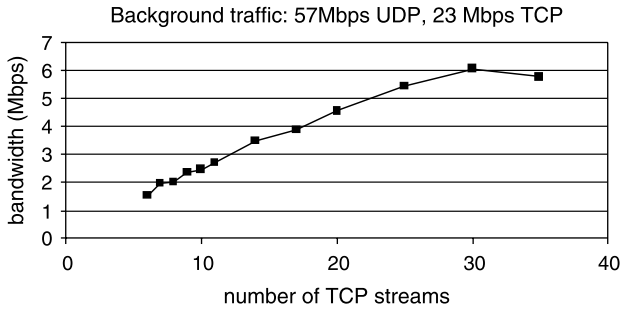

Fig. 6. TCP foreground with 80 Mbps background traffic.

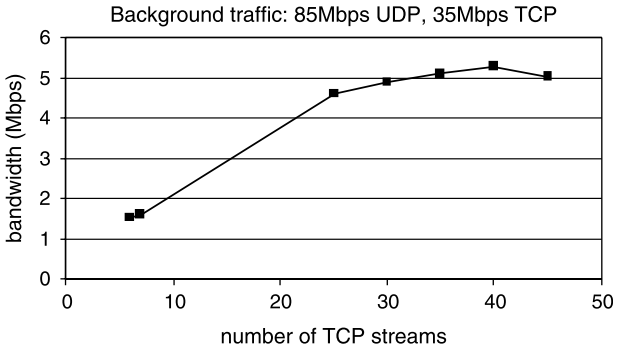

Fig. 7. TCP foreground with 120 Mbps background traffic.

the performance of real-time applications that experience the preferential treatment. For implementing the following testing scenarios we used an application based on the OpenH323 library.

Initially, the network was loaded with background traffic, generated by the Iperf traffic generator (the selected traffic is a mix of TCP and UDP traffic). At this point we should note that we have started loading the network before inserting the foreground traffic, in order for TCP to obtain a stable state. A while later, the foreground traffic that was generated by a videoconference (using the OpenPhone application based on the OpenH323 library) was inserted.

This test was performed for more than 5 minutes and we recorded the packets that were exchanged. The background traffic was 5 Mbps UDP traffic and also TCP traffic that tried to occupy as much bandwidth as it could. Finally, the results showed that the UDP background traffic had only a few packets dropped. Similarly, the foreground traffic (OpenH323) had zero packet loss and excellent quality, which proves that the QoS mechanisms achieved their goal. In addition, the TCP background traffic was straggled by the strict priority mechanism. But on the other hand, the foreground traffic has an average throughput of almost 300 Kbps and very good video quality.

Similarly, a second test with the same characteristics and traffic load was performed. The only difference was that we also added at the foreground traffic extra UDP traffic (300 Kbps), created by the Iperf traffic generator. The results were the same, as the foreground traffic had almost zero packet loss (both UDP and RTP). In addition, the RTP traffic (OpenH323) had excellent video quality taking advantage of the operation of the strict priority mechanism (low latency queue). The foreground traffic displayed a steadier rate, which happens because of the inclusion of artificially generated UDP traffic. Artificially generated traffic has a much lower variation in its transmission rate than the actual VoIP traffic generated by the OpenH323 application.

4.5. Investigation of WRED mechanism

Our next goal was to investigate the operation of the WRED mechanism. The WRED mechanism is a popular mechanism for congestion avoidance that has been tested extensively in IPv4. During our tests we tried this mechanism on our IPv6 domain. The tests aimed to prove the correct operation of this mechanism on IPv6 and secondly to investigate its impact on the performance of the foreground traffic. At this stage, 2 separate testing scenarios were performed. In the first one, the WRED mechanism was set up in order to be applied on the background traffic using 30 and 50 packets in the queue as thresholds. In addition, the maximum queue size was 75 and the drop possibility was 10%. A test that we had also performed

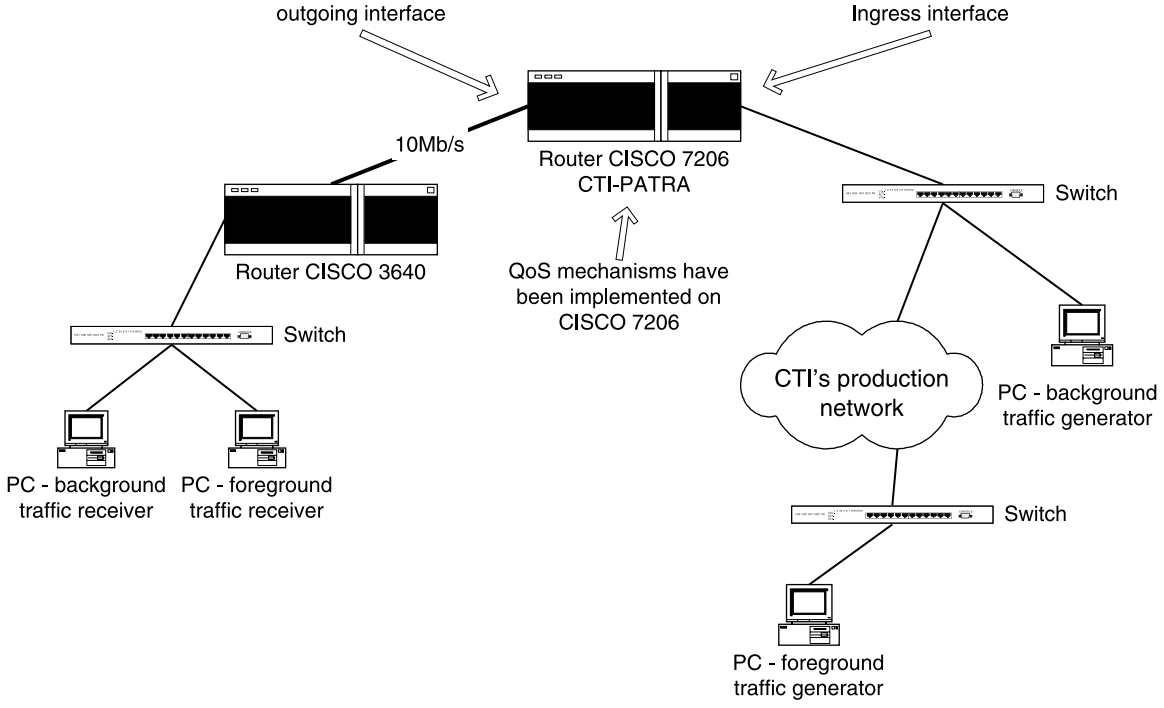

Fig. 8. The local CTI testbed.

earlier was repeated with this new configuration, in order to compare the results. So, we inserted background traffic that was a mix of TCP and UDP (5 Mbps) and foreground traffic a mix of UDP (700 Kbps) and RTP (OpenH323-based application). The result was that the foreground traffic still only had a few packet losses and very good quality of video. On the other hand, the background traffic had several drops that were caused by the WRED mechanism (UDP background traffic had almost 2% packet loss). So, the foreground traffic does not seem to receive any impact from the operation of the WRED mechanism. The strict priority mechanism seems to work transparently. On the other hand, the background traffic has many packet losses, especially, if we compare the result (2% losses of UDP) with the same experiment in Section 4.4 without the WRED mechanism, where the result was less than 0.5%. So, the WRED mechanism worked according to its specification and reduced the background traffic. The most significant observation arises when we look at the TCP throughput of the background traffic and compare it with the corresponding throughput on Section 4.4. It is obvious that the throughput in this case is lower and that the WRED caused this reduction of TCP's rate. This can be explained if we consider that the WRED mechanism 'created' packet losses earlier, which led TCP to believe that the network was congested and reduced its rate.

The second scenario was identical to the first but this time we changed the thresholds of the WRED mechanism. We tried to approach the max queue size and configured the min and max thresholds to be 55 and 75 packets, respectively. The drop possibility was also 10%.

We observed similar results regarding the foreground traffic, as the packet losses were almost zero and the video quality was very good. This time the background traffic had better behavior, as only 0.92% of UDP traffic packets were lost. In addition, the TCP traffic had a bigger average throughput (1.36 Mbps). So, at this experiment the queues were allowed to be more filled and achieved a better performance for the background traffic. But, regarding the foreground traffic (for the QoS service that is tested) the existence of the WRED mechanism does not have any significant impact.

4.6. Flow label usage

Flow label is a field in 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 [24], but specific use cases have not been described yet. RFC 3595 [25] 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, the 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 flow-specific state and in any case 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 routers' performance. Therefore, the RFC defines that the 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. The flow expiration also allows the Flow Label values reusability. 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 security. In particular, it is crucial to avoid theft of service attacks by unauthorized traffic. Such attacks are possible with the two following ways: by spoofing the Flow Label value (only on valid nodes that use 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, in this point we should note that the 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 example that can take advantage from the flow label field is a VoIP implementation that can achieve 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.

5. Future work

All the tests that were performed also indicated some points that need further research and investigation. The most interesting and open issue for research is the investigation of the way that the policy profile should be selected and configured for an aggregate of flows of real time data in order to follow the SLAs. We are also interested on testing QoS services in dual stack networks, where the QoS mechanisms serve IPv4 and IPv6. The latter is very important in order to

measure possible performance problems between IPv4 and IPv6 flows.

6. Conclusions

In this paper, we presented a QoS service that was implemented on a large-scale native IPv6 network and examined its performance. The QoS mechanisms that were used were tested widely in order to make sure that they work well and additionally to investigate their performance. The tests used simulated background traffic (but as close to reality as possible) to overload the network, and the QoS service aimed to handle traffic that belongs to real-time applications. Examining all the above experiments we come to the conclusion that the QoS service we tried to implement and test, using the above mechanisms, worked efficiently as it reduced the packet loss, the delay and the jitter of the real-time data, consequently increasing the receiving quality for the applications that produced these data.

Additionally, the results from the experiments on the WRED mechanism that had been applied on the background traffic indicate that there was no significant impact on the foreground traffic from the existence of the WRED. On the other hand, the background traffic has been affected from the WRED mechanism and the impact is proportional to the values that the thresholds of the WRED mechanism had been configured at.

The overall conclusion is that the QoS service, with the use of the specific mechanisms that were tested, can provide prioritization on an IPv6 domain to the specified traffic and therefore the real-time application (that produces the traffic) can operate efficiently and with high quality. The experience we gained from our involvement with IPv6 also shows that having ported useful applications to the new Internet Protocol will play a crucial role to its further adoption.

Acknowledgements

The authors would like to thank the 6NET project partners for their valuable cooperation and contribution to the experiments. In particular, we would like to thank Lancaster University, Cisco Systems, Greek Research and Technology Network (GRNET), United Kingdom Education and Research Networking Association (UKERNA) and the 6NET project as a whole, which is funded by the IST program of the European Commission (IST Contract No: 2001-32603).

References

[1] Deering S., Hinden R., RFC 2460, 'Internet Protocol, Version 6 (IPv6) Specification', Internet Engineering Task Force, December 1998
[2] OpenH323 project, http://www.openh323.org
[3] S. Josset, C. Bouras, A. Gkamas, K. Stamos, Adding IPv6 support to H323: gnomemeeting/openH323 port, 11th International Conference on Software Telecommunications and Computer Networks (SoftCOM 2003) Croatia, Italy, October 7-10 2003 pp. 458–462.
[4] C. Bouras, A. Gkamas, D. Primpas, K. Stamos, Performance Evaluation of an IPv6-capable H.323 Application. The 18th International Conference on Advanced Networking and Applications (AINA 2004) Fukuoka, Japan, March 29-31 2004 pp. 470–475.
[5] 6 NET project, http://www.sixnet.org.
[6] The IPv6 Forum, http://www.ipv6forum.com/
[7] HS247, http://hs247.com/
[8] Nordmark E., RFC 2765, 'Stateless IP/ICMP Translation Algorithm (SIIT)', Internet Engineering Task Force, February 2000
[9] Tsuchiya K., Higuchi H. , Atarashi Y., RFC 2767, 'Dual Stack Hosts using the 'Bump-In-the-Stack' Technique (BIS)', Internet Engineering Task Force, February 2000
[10] Lee S., Shin M-K., Kim Y-J., Nordmark E., Durand A., RFC 3338, 'Dual Stack Hosts Using 'Bump-in-the-API' (BIA)' Internet Engineering Task Force, October 2002
[11] Hagino J., Yamamoto K., RFC 3142, 'An IPv6-to-IPv4 Transport Relay Translator', Internet Engineering Task Force, June 2001
[12] S. Vegesna, IP Quality of Service: The Complete Resource for Understanding and Deploying IP Quality of Service for Cisco Networks, Cisco Press, 2001.
[13] 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.
[14] Euro6IX project, http://www.euro6ix.org/
[15] 6 WINIT project, http://www.6winit.org/
[16] 6 POWER project, http://www.6power.org/
[17] 6 QM project, http://www.6qm.org/
[18] SATIP6 project, http://satip6.tilab.com/
[19] Gilligan R., Thomson S., Bound J., Stevens W., RFC 2553, 'Basic Socket Interface Extensions for IPv6', Internet Engineering Task Force, March 1999
[20] Iperf, The TCP/UDP Bandwidth Measurement Tool, http://dast.nlanr.net/Projects/Iperf/
[21] Casner S., Frederick R., Jacobson V. and Schulzrinne H., RFC 3550, 'RTP: A Transport Protocol for Real-Time Applications', Internet Engineering Task Force, July 2003
[22] Ethereal Network Analyzer, http://www.ethereal.com/
[23] Cisco Systems, Inc. home page, http://www.cisco.com
[24] Rajahalme J., Conta A., Carpenter B., Deering S., RFC 3697, 'IPv6 Flow Label Specification', Internet Engineering Task Force, March 2004
[25] Wijnen B., RFC 3595, 'Textual Conventions for IPv6 Flow Label', Internet Engineering Task Force, September 2003