MPLS/DiffServ Interworking: preliminary functional and performance tests for TANGO project

Gino Carrozzo, Nicola Ciulli, Giacomo Sergio

META Centre
Consorzio Pisa Ricerche
C.so Italia 116, 56125 Pisa, ITALY
Telephone: (+39) 050 915 811
Fax: (+39) 050 915 823
Email: {g.carrozzo, n.ciulli, g.sergio}@cpr.it

Stefano Giordano, Giodi Giorgi, Fabio Mustacchio, Gregorio Procissi

Dept. of Information Engineering
University of Pisa
via Caruso, 56122 Pisa, ITALY
Telephone: (+39) 050 2217 511
Fax: (+39) 050 2217 522
Email: {s.giordano, g.giorgi, fabio.mustacchio, g.procissi}@iet.unipi.it

***Abstract*— The DiffServ and the MPLS control and data plane functionalities might be both active at the same time in the same backbone IP network, but their full inter-working is still a matter of study for both standardization committees (e.g. IETF Traffic Engineering Working Group) and router manufacturers. This integration implies a complete implementation of the low-level network functionalities, in terms of QoS support and MPLS path management, in order to enable a framework for the deployment of an advanced management of the QoS-IP traffic connections. This paper aims at reporting the preliminary functional tests carried out on an experimental test-bed, made up of two inter-connected MAID domains of prototypal routers. The rationale for these tests is that the final automated MAID architecture will be based on the reliable and validated operation of these basic and well-known functionalities in a modular framework.**

I. Introduction

The current research and operative IP broadband networks are based on well-known and mature technologies (e.g. MPLS) and QoS architectures (e.g. IntServ, DiffServ).

The IETF DiffServ architecture is largely recognized as a flexible and scalable solution for the provisioning of QoS-IP network transport services and it is currently implemented in several commercial routers, prototypes and field trials derived from research projects. Moreover, different combined solutions (e.g. IntServ/DiffServ) have been proposed and tested, aimed at building end-to-end dynamic services which adapt the coarse grained QoS configured in backbone the to the finer grained user QoS requests.

Concerning the Multi Protocol Label Switching (MPLS) technology, its standardization has resulted in an enhanced control plane for IP networks made up of a set of tools (e.g. for traffic engineering and its survivability) and protocols aimed at enabling the paradigm of an intelligent circuit-oriented network in the context of the connection-less packet networks.

DiffServ and MPLS might be both active at the same time in the same network, but their full inter-working is still a matter of study for both standardization committees (e.g. IETF Traffic Engineering Working Group) and router manufacturers. Some general protocol extensions have been defined for DiffServ support in MPLS architecture [1], and they are going to be available on commercial routers. The main issues towards the integration of the DiffServ and the MPLS technologies inside a unique architecture rely on:

- managing DiffServ-aware traffic engineering mechanisms both in an intra-domain and in an inter-domain deployment scenario;
- extending the existing protocols for the Label Switched Paths (LSP) request, set-up/tear-down in order to support DiffServ QoS guarantees;
- defining and verifying the operation of an integrated multi-layer and multi-technologies IP network.

Realizing such an integration implies, at first, to deploy the complete implementation of the low-level network functionalities, in terms of QoS support (e.g. IP traffic control and conditioning) and MPLS path management (e.g. LSP setup and traffic injection). This results in the setup of a complete framework for the deployment of an advanced management of the QoS-IP traffic connections. As detailed in [2], the Multiple Access Inter-Domain (MAID) architecture is aimed at providing Network Operators with the robust and user-friendly mechanisms to set-up QoS-aware LSPs, hiding the underlying complexity of managing all the involved parameters.

This paper aims at reporting the preliminary functional tests carried out on an experimental test-bed, made up of two inter-connected MAID domains of prototypal routers based on IA32(PC) Linux OS platforms. The tested MAID functionalities are a subset of the overall designed ones [2], since the DiffServ management is completely automated and under the Bandwidth Broker control (also in the inter-domain case), but not completely integrated with the related LSP signalling phase yet. The rationale for this preliminary test phase is that the final automated MAID architecture will be based on the reliable and validated operation of these basic and well-known functionalities in a modular framework.

In the following sections, after a brief recall of the MAID architecture basics (ref. Sec. II), the experimental scenario is detailed (ref. Sec. III). The focus is on the test-bed topology and on the applications generating the IP traffic (both artifi-

cially generated and based on real-time multimedia streaming) for which a QoS-MPLS treatment is applied. In Sec. IV the collected performance is shown and discussed for the different test scenarios, while in Sec. V conclusions are derived with some hints to future upcoming work.

II. The MAID network architecture

The MAID network architecture (ref. Figure 1 and [2]) is based on the advanced functionalities of two network elements: the Multiple Access Border Router (MA-BR) and the Bandwidth Broker (BB). IP flows with QoS requirements are managed by the MA-BR, which provides the inter-working between the access network and the backbone. This network element merges the functionalities of an edge router (ER) and a border router (BR), providing the correct handling of QoS parameters contained in the different signaling protocol messages. The BB, instead, manages the backbone resources, in order to provide a fine-grain/tailored traffic engineering and to handle policy/admission procedures for IP flows accessing the backbone.

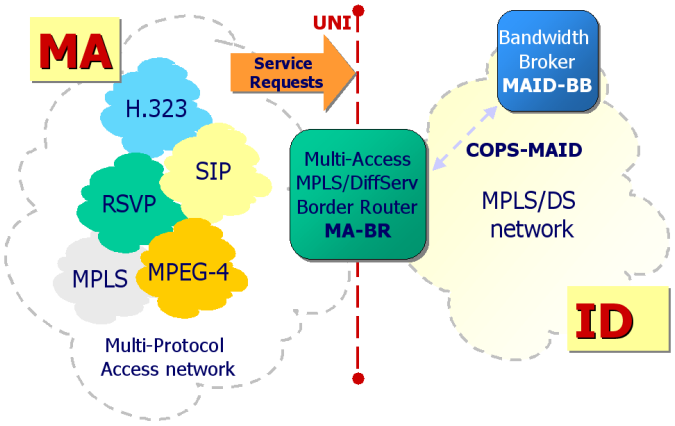

Fig. 1. Multiple Access Inter-Domain network scenario.

The modular composition of the network elements used in the MAID test-bed are shown in Figure 2, in Figure 3 and in Figure 4.

The MAID data plane provides the mapping and forwarding of the access network flows into the proper DiffServ PHBs/MPLS Label Switched Paths (LSP) and vice versa. On the other hand, the MAID control plane is responsible for Admission Control (AC) and policy decisions (taken on a per-flow or per-PHB basis) and for the service level agreement (SLA) maintenance.

MPLS LSPs with QoS-DiffServ guarantees are established both inside a unique MPLS/DiffServ domain and across multiple domains. The signaling protocol for setting up and tearing down LSPs inside a domain is RSVP-TE [1][3][4], while the signalling protocol for the communication between an access network and the MA-BR is application-dependent (e.g. RSVP for IntServ, H.323 or SIP for VoIP, etc.). The protocol used for the communication between the MA-BR and the BB is

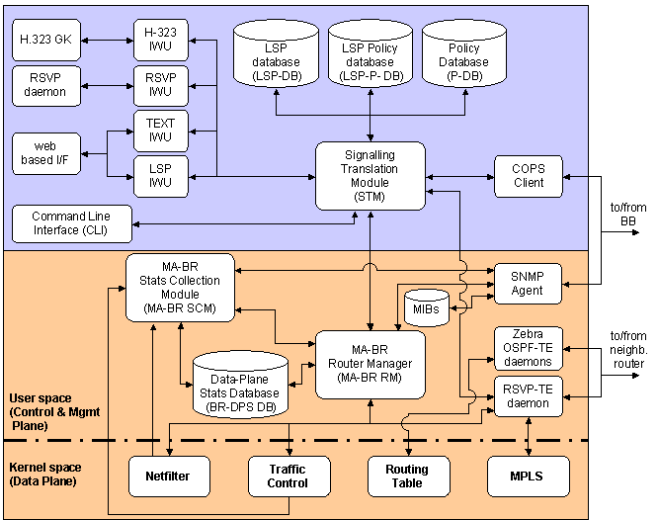

Fig. 2. MAID Border Router internal modules.

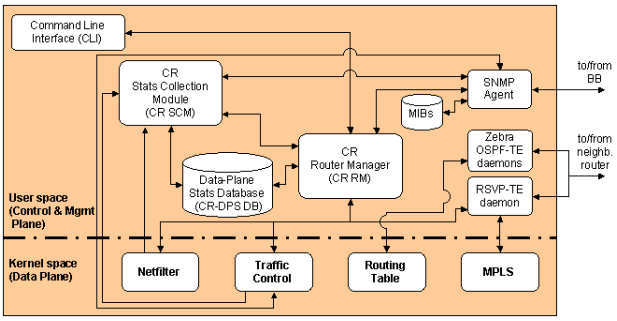

Fig. 3. MAID Core Router internal modules.

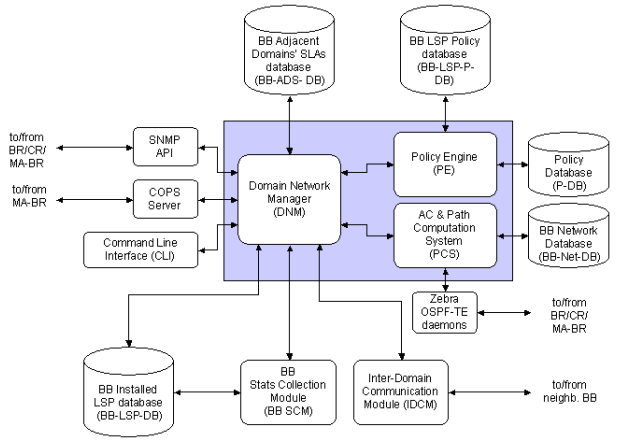

Fig. 4. MAID Bandwidth Broker internal modules.

the Common Open Policy Service protocol (COPS) [5][6] extended with a unified MAID semantic [7].

A detailed overview of the internal design issues is out of the scope of this paper and may be found in [2] and in [8].

III. The experimental scenario

The preliminary functional tests discussed in this paper have been carried out on a distributed test-bed made up of two

inter-connected MAID domains. These domains are located in the networking laboratories at the Department of Information Engineering of the University of Pisa and at the META Centre of the Consorzio Pisa Ricerche. The two domains are permanently interconnected through a Gigabit Ethernet optical fiber link. At the network layer, each domain is configured as an independent autonomous system with proper strategies and policies for QoS provisioning and Traffic Engineering. The routers in each domain are prototypal routers based on IA32(PC) Linux OS platforms, equipped with the common well-known modules for the management of MPLS resources and for traffic conditioning (i.e. Traffic Control, TC), and with most of the MAID-specific modules sketched in Sec. II.

A detailed overview of the configured topologies is shown in Figure 5. Each domain has its own Bandwidth Broker, which manages the dynamic configuration of the network resources under its scope (via COPS for MA-BRs and via SNMP for CRs and BRs), as well as the inter-domain communication by means of COPS-MAID protocol [2]. In both cases, BBs trigger their actions upon receiving requests for decisions through their own COPS Server.

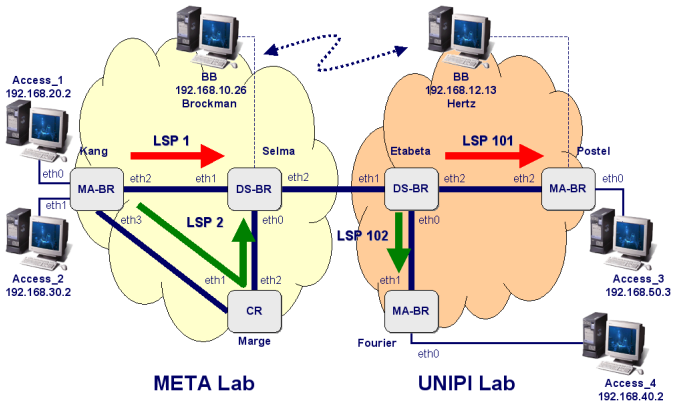

Fig. 5. Distributed MPLS/DiffServ test-bed topology.

The prototypal routers used in the test-bed have been configured in order to test all the possible functionalities of a MAID network:

- MA-BRs (i.e. Kang, Postel and Fourier), which play the role of MPLS/DiffServ LERs (Label Edge Router) by managing the traffic from/to the access clients and injecting it in the proper DiffServ LSPs; moreover, MA-BRs triggers the requests (via COPS-MAID) for policy/admission control decisions to the respective BB;
- CR (i.e. Marge), which plays the role of MPLS/DiffServ LSRs (Label Switching Router); it receives the DiffServ configuration of its TC resources via SNMP from its BB and the MPLS configuration of explicitly routed LSPs via RSVP-TE from the originating LSR;
- BRs (i.e. Selma and Etabeta), which play the role of MPLS/DiffServ LERs like the MA-BRs, but basically oriented to the inter-domain operation.

The scheduler used by the network elements is the implementation of the Hierarchical Token Bucket (HTB) available in the iproute2-2.4.7 package for Linux kernel 2.4.20. HTB is a kind of CBQ (Class Based Queuing) algorithm, approximating service discipline based on the class concept; this feature is fundamental when dealing with DiffServ PHBs.

The access networks/clients have been configured in order to play the role of source/destination of different kind of QoS-unaware IP traffic. Two types of traffic flows have been injected in the MAID test-bed:

- artificial traffic, generated by specialized applications;
- real-time traffic, generated by the delivery of multimedia contents.

A. *Artificial traffic*

Two applications have been used in the MAID test-bed to generate artificially traffic flows: RUDE v0.62 [10] and BRUTE v1.0Beta [11].

RUDE stands for Real-time UDP Data Emitter and it is a small and flexible application that generates UDP traffic in two modes: constant bit rate, which is the commonly used selection, and user-defined traffic traces. The operation and configuration of RUDE are similar to other traffic generator tools (e.g. MGEN), but, instead of using an approach entirely based on the system functionalities (e.g. for timers resolution), RUDE is conceived as a system-independent application.

BRUTE is the acronym of Brawny and Rough UDP Traffic Engine and it is another user space application designed at the Department of Information Engineering of the University of Pisa for generating high-load customizable traffic flows. BRUTE provides more traffic distributions with respect to RUDE (e.g. CBR, Poisson, Poisson Arrival of Burst [12]) as well as a more flexible configuration interface for defining customized traffic profiles. High performance is obtained both synchronizing the sending process with more accuracy and higher resolution and reducing transmitted packet latency adopting data-link socket to bypass the TCP-IP kernel stack.

B. *Real-Time traffic*

The real-time traffic injected in the MAID test-bed is a streaming of multimedia contents from a server to the requesting connected clients. The testing environment is based on the Helix DNA platform, which is an open-source standard-based software for streaming multimedia productions over IP networks. This platform is developed by the Helix Community and hosted by RealNetworks.

The server side of the platform is based on the Helix DNA Server engine [13]. It can support the real time packetization and network transmission of different media types (e.g. MP3 audio, RealAudio and RealVideo). Multimedia contents are streamed by means of RTSP/RTP sessions [14] via HTTP, TCP or UDP connections, both in unicast and multicast mode.

The streaming selection from the server archive can be performed via an HTTP connection that returns the RTSP session description to the client (e.g. green lines in the upper side of Figure 6). The control of the multimedia traffic is managed by the server on the basis of the RTSP messages

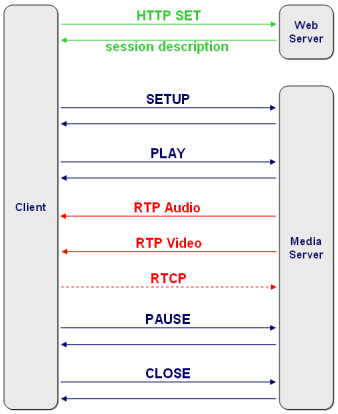

Fig. 6. RTSP operations.

(e.g. SETUP, PLAY, PAUSE, CLOSE) received by each client on a dedicated connection (e.g. blue lines in Figure 6). The multimedia traffic is streamed by two RTP sessions, one for the audio the other for the video contents (e.g. red line in Figure 6).

The optional session information collected on the server by the arrival of RTCP packets from clients (i.e. red dashed line in Figure 6) enables a connection monitoring.
If multiple versions of the same multimedia content are available on the server at different encoding bit-rates, the resulting statistics can trigger a dynamic adjustment of the output RTP streaming bit-rate.

The purpose of using such a multimedia platform in this work is limited to testing the performance of the MAID network when managing the real-time traffic generated by operative multimedia applications. Therefore, the client (i.e. a common Real One Player) and the server have been configured in order to start directly the RTSP streaming of Real Media contents (Audio plus Video) without any preliminary HTTP server wrapping. Moreover, the RTSP control traffic has been conveyed in a TCP connection with no QoS and no MPLS treatment (i.e. best effort behavior), while the RTP downstream contents have been injected in DiffServ LSPs in order to guarantee multimedia quality on the receiving client.

IV. Performance Studies

Different tests are carried out for assessing the performance of the MAID test-bed with respect to the different source applications and traffic profiles injected into the network. These tests highlight also the critical elements of the MAID data plane, responsible for an unexpected limitation in the overall performance. In all the tests traffic is sent after a configuration phase takes place. This phase is similar to the static resource provisioning provided by the Network Operator for those QoS-unaware access networks that can not use the dynamic MAID-UNI features. Configuration consists of the DiffServ LSPs establishment and of the mapping of the traffic flows into LSPs by means of a WEB interface. For each LSP, a QoS class and a reserved bandwidth are signaled.

A. Artificial traffic

Since two different applications for artificial traffic generation have been selected, different traffic flows have be injected into the test-bed. In order to collect significant results, all the test of this class have been carried out by generating two traffic flows from the same source client: the first to 192.168.50.3:6970 through a DiffServ LSP with an EF reservation of $1.5Mbps$, the latter to 192.168.40.2:7970 through another DiffServ LSP with an AF_{11} reservation of $512kbps$.

In the first test, the two traffic flows are generated by RUDE with a Constant Bit Rate of $1.5Mbps$ and $512kbps$, respectively. Figure 7 shows that some packets are dropped even if the reserved rate equals exactly the nominal traffic rate.

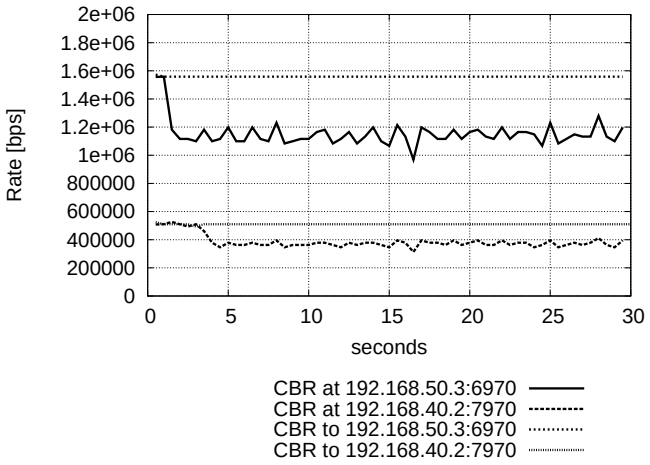

Fig. 7. RUDE: CBR traffic flows.

The same behaviour can be observed if the test is repeated by using the BRUTE traffic generator (ref. Figure 8). The policer located on the ingress MA-BR is the software element responsible for this packet dropping. Therefore, this element requires an accurate configuration/tuning of its parameters, in order to achieve the desired performance, above all when operating in quasi-saturation conditions.

Aiming at characterizing the policer performance, different traffic profiles, other than the CBR one, have been injected into the network. The BRUTE traffic generator enables this kind of tests by generating Poissonian and Poissonian Arrival of Burst (PAB) flows with the same mean bit-rate of the CBR one. In the former case (e.g. Poissonian traffic), a dropping behavior similar to that of the CBR case can be observed (ref. Figure 9). In the latter case (e.g. PAB traffic), instead, the higher burstiness of the traffic significantly increases the packet drop rate (i.e. from 25% up to 50% ca.), hence showing a further degradation of the policer performance.

To demonstrate the experimental repeatability of the above tests, they have been run for 10 times. In all cases the dropping performance has been very close to the values mentioned above as shown in Figure 11.

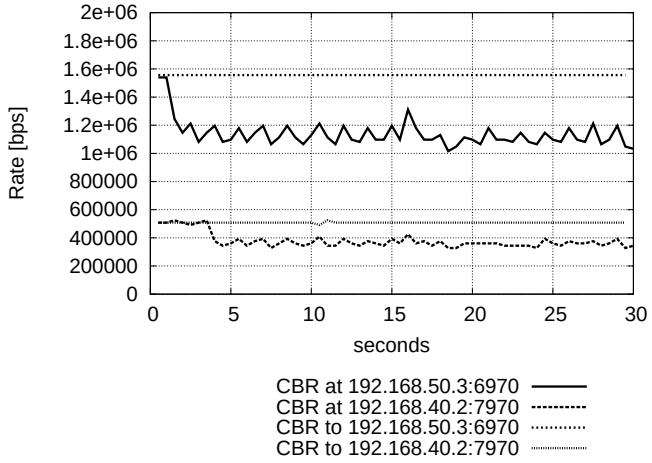

Fig. 8. BRUTE: CBR traffic flows.

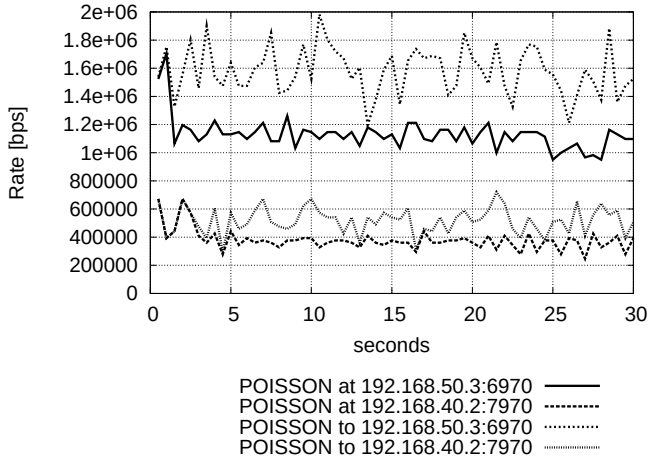

Fig. 9. BRUTE: Poisson traffic flows ($\lambda = 1\ s^{-1}$).

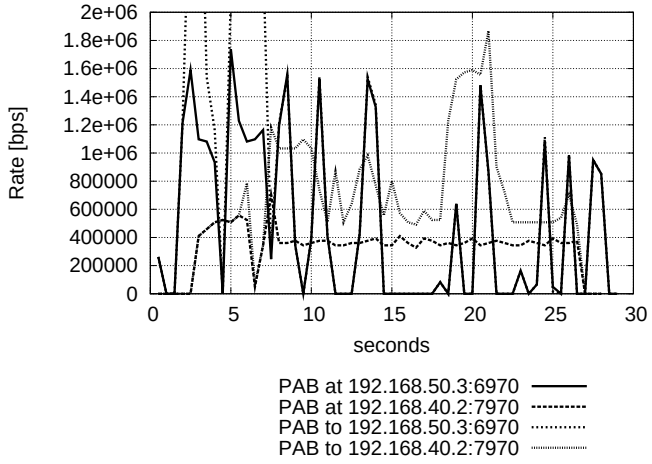

Fig. 10. BRUTE: Poisson arrival of Burst traffic flows ($\lambda = 1\ s^{-1},\ \alpha = 1.5,\ \theta = 0.5\ s$).

B. Real-Time streaming traffic

Two types of tests are performed in this testing scenario. The first one is characterized by:

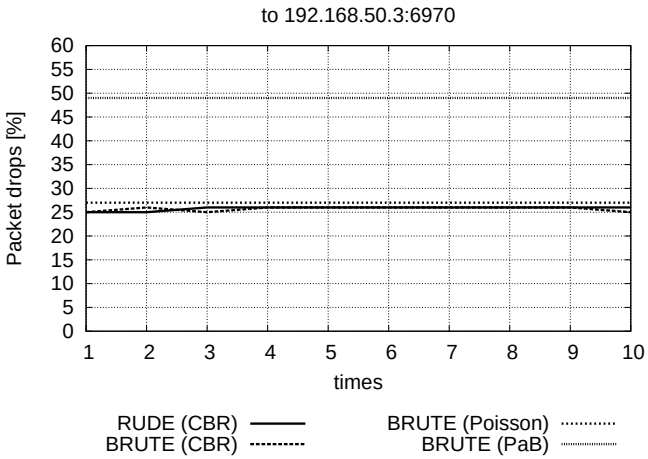

Fig. 11. Policer performance in terms of packet drops.

- a fixed amount of bandwidth reserved in the QoS-MPLS network;
- a single encoded version of the multimedia content, streamed at the encoding bit-rate of $768kbps$;
- a variable connection type configured on the destination client.

The video streaming is flowed to 192.168.50.3:6970 through a DiffServ LSP with an EF reservation of $1Mbps$ and to 192.168.40.2:7970 through another DiffServ LSP with an AF_{11} reservation of $512kbps$. Figure 12 shows the received bit-rate when both clients have been configured with a LAN connection speed (e.g. $10Mbps$).

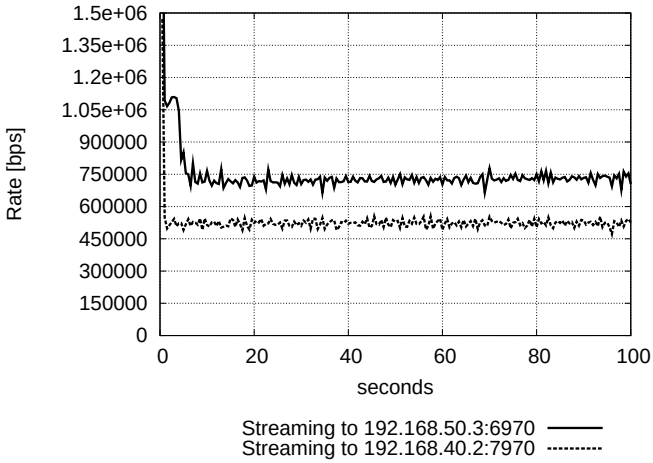

Fig. 12. RTSP streaming with LAN speed configuration on both clients (encoding @$768kbps$ only).

In this scenario, after a few seconds in which some packets are dropped on both connections, the client attached to the EF LSP perceives a good video and audio quality. Instead, the client attached to the AF_{11} LSP experiences a jerky reproduction because of the packet drops induced by a reserved bandwidth (e.g. $512kbps$) lower than the encoding rate (e.g. $768kbps$). The poor quality of the played contents is also

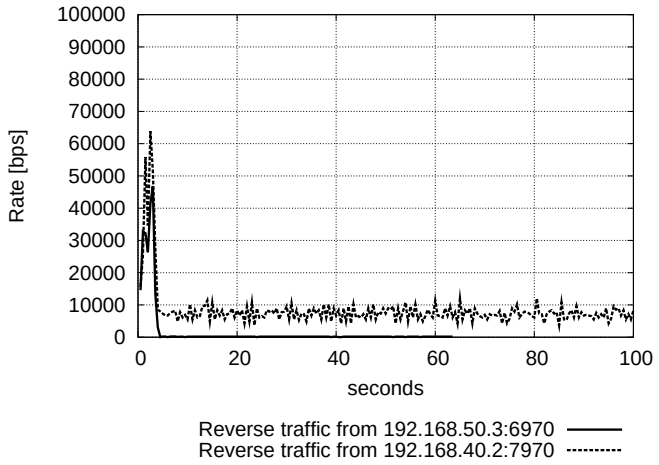

Fig. 13. Reverse RTCP traffic in case of LAN speed configuration for both clients (encoding @$768kbps$ only).

highlighted by the amount of reverse RTCP traffic (i.e. from the client to the server) that carries connection monitoring information (ref. Figure 13).

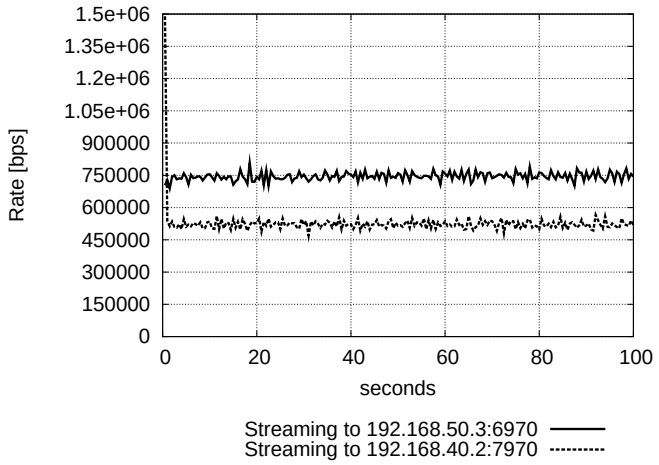

Fig. 14. RTSP streaming with speed configuration DSL for the client attached to EF LSP and LAN for the other (encoding @$768kbps$ only).

Packet drops in the initial phase are due to the server attempt to fill the buffer at the full client connection speed (e.g. $10Mbps$), as announced by the client itself in the setup phase. This problem can be solved by limiting the initial server "turbo-rate" to one of the possible slower connection types (e.g. T1 at $1.5Mbps$ or DSL at $768kbps$). Figure 14 shows the absence of packet drops for the traffic flow on the EF LSP, when the related client has been configured with a DSL connection. Since the client is configured so as its bandwidth fits the encoding rate, it does not need to buffer at a higher rate, resulting in a optimal perceived quality throughout the whole streaming. However, this performance is basically related to the amount of bandwidth reserved for the stream throughout the network. Indeed, when the client cannot succeed in filling up its buffer at an acceptable rate (e.g. in case the encoding bit-rate - $768kbps$ - is higher than the reserved bandwidth - $512kbps$), it triggers automatically a mechanism of PAUSE-PLAY repetitions (e.g. the trace towards 192.168.40.2:7970 in Figure 15), waiting for possibly better network conditions. Obviously, the resulting multimedia content played is of poor quality.

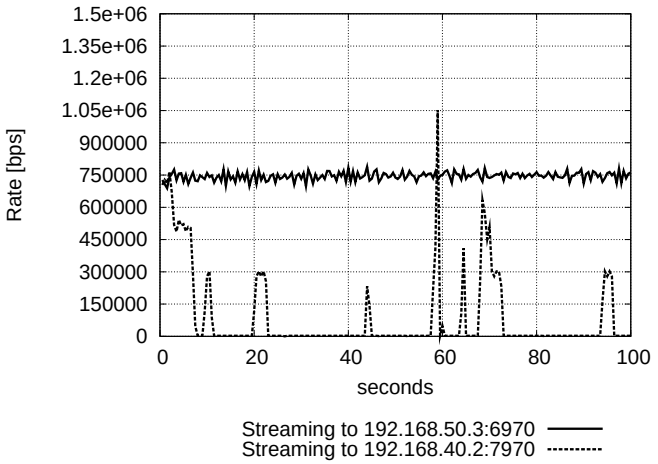

Fig. 15. RTSP streaming with DSL speed configuration on both clients (encoding @$768kbps$ only).

The latter type of tests is performed to evaluate performance when the same multimedia content is available at different encoding rates. In this case, the server chooses the best fitting encoding bit-rate on the basis of the information on the connection, collected in the initial phases of the streaming. These tests are characterized by:

- a variable amount of bandwidth reserved in the QoS-MPLS network;
- two different versions of the same multimedia content streamed at encoding bit-rates of $768kbps$ or $512kbps$;
- a DSL (e.g. $768kbps$) connection type configured on the clients.

In Figure 16, the streaming is flowed to 192.168.50.3:6970 through a DiffServ LSP with an EF reservation of $1Mbps$ and to 192.168.40.2:7970 through another DiffServ LSP with an AF_{11} reservation of $512kbps$. In this case, the two clients negotiate the proper rate with the server (e.g. $768kbps$ for the traffic through the EF LSP and $512kbps$ for the other). It can be noticed that the traffic flowing into the AF_{11} LSP is attempted to be buffered at a higher bit-rate, since the server supposes to deal with a connection at bandwidth higher (i.e. DSL) than the one actually reserved. Thus, during the initial phase, packet drops occur. Thereafter, no packet loss is experienced and the differences on the perceived playing quality on the two clients are due to the different encoding rates (i.e. better playing on the client attached to the EF LSP).

The same objective and perceptive QoS performance resulted from a network configuration in which the LSP to 192.168.50.3:6970 has an EF reservation of $768kbps$ (ref. Figure 17). This confirms that to eliminate the transient "turbo-

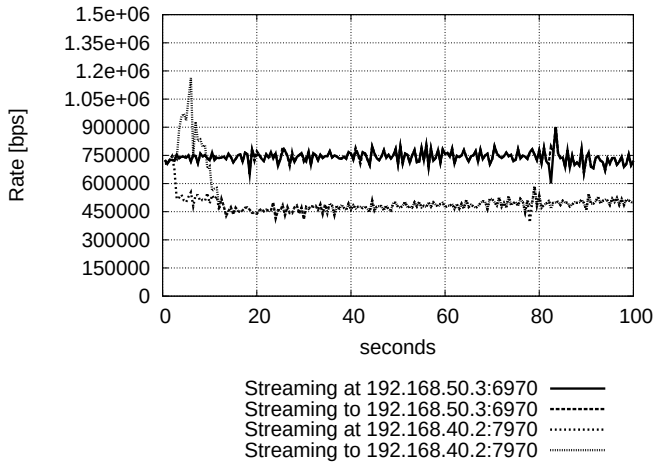

Fig. 16. RTSP streaming with DSL speed configuration on both clients and an over-provisioned reservation for the client attached to the EF LSP (encoding @$768kbps$ and @$512kbps$).

player" effect, a proper connection speed configuration in the client player is sufficient.

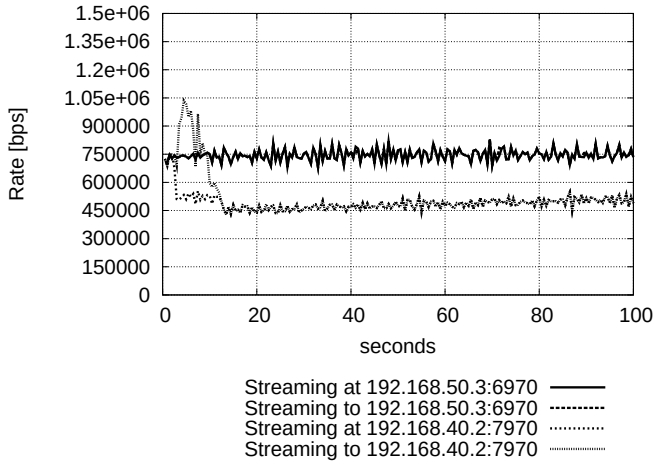

Fig. 17. RTSP streaming with DSL speed configuration on both clients and tailored reservation for the client attached to the EF LSP (encoding @$768kbps$ and @$512kbps$).

V. Concluding remarks and Future Work

This paper reports the preliminary functional tests of the interworking of the MPLS and DiffServ technologies in an experimental MAID test-bed.

The experimental scenario is based on a first release of the MAID network software deployed in two inter-connected remote domains of prototypal routers. The main purpose of the tests is to prove the low-level network functionalities for QoS support (e.g. IP traffic control and conditioning) and for MPLS path management (e.g. LSP setup and traffic injection) in a semi-automatic deployment scenario. The tested MAID functionalities are a subset of the overall designed ones, since the DiffServ management is completely automated and under the Bandwidth Broker control (also in the inter-domain case), but not completely integrated with the related LSP signalling phase yet.

The experimental test-bed is solicited with different kinds of IP traffic for which a QoS-MPLS treatment is applied. Traffic is both artificially generated (i.e. by means of specific applications) and based on real-time streaming of multimedia contents (e.g. audio + video).

The collected performance shows the effectiveness of the modular MAID architecture when managing QoS-MPLS services across different administrative domains, as well as some weak points of available prototypal implementations (e.g. the ingress policer performance when operating in quasi-saturation conditions).

The results of this test campaign spur on the deployment of a completely automatic QoS-MPLS service setup and on a full exploitation of the MAID-UNI/NNI capabilities, when extending the experimental scenario to those QoS-capable access networks (e.g. IntServ, H.323, SIP, etc.) from which the MAID service setup can be dynamically triggered by intercepting the call setup signalling.

Acknowledgements

This work has been supported in part by the Italian Ministry of Education, University and Research through the project TANGO (MIUR Protocol No. RBNE01BNL5).

References

[1] F. Le Faucheur (Editor) et al., *Multi-Protocol Label Switching (MPLS) Support of Differentiated Services*, IETF RFC3270, May 2002
[2] G. Carrozzo et al., *Multi Access Inter Domain architecture for QoS provisioning in MPLS/DiffServ networks*, submitted to the 2004 International Conference on Communications (ICC2004).
[3] D. Awduche et al., *RSVP-TE: Extensions to RSVP for LSP Tunnels*, IETF RFC 3209, December 2001.
[4] D.Black, S. Brim, B. Carpenter, F. Le Faucheur, *Per Hop Behavior Identification Code*, RFC 3140, June 2001.
[5] D. Durham, Ed., J. Boyle, R. Cohen, S. Herzog, R. Rajan, A. Sastry, *The COPS (Common Open Policy Service) Protocol*, IETF RFC 2748, January 2000.
[6] S. Herzdog, Ed., J. Boyle, R. Cohen, D. Durham, R. Rajan, A. Sastry, *COPS Usage for RSVP*, IETF RFC 2749, January 2000.
[7] G. Carrozzo (Editor), *COPS-MAID, COPS Usage for Multi-Access Inter-Domain MPLS-DiffServ Networks*, draft-cpr-rap-cops-maid-00.txt, Internet Draft, Work in progress, November 2003.
[8] *Traffic models and Algorithms for Next Generation IP networks Optimization (TANGO) Project*, http://tango.isti.cnr.it/.
[9] N. Ciulli et al., *An Architecture for Transparent End-to-End QoS-IP Transport Service Provisioning*, Proc. of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '03), Las Vegas (NV) USA, June 23-26 2003.
[10] *Real-time UDP Data Emitter (Rude)*, http://rude.sourceforge.net/.
[11] *Brawny and Rough Udp Traffic Engine (Brute)*, http://netgroup-serv.iet.unipi.it/brute/.
[12] W.Feller *An introduction to probability theory an its application*, Vol.2 New York Wiley 1966.
[13] *Helix DNA Server* 9.0 *(*9.4.4.25*)*, http://www.helixcommunity.org/.
[14] H. Schulzrinne et al., *Real Time Streaming Protocol (RTSP)*, draft-ietf-mmusic-rfc2326bis-05.txt, Internet Draft, Work in progress, Oct. 2003.