A Virtual High-Speed Network Laboratory[1]

Rafael Timóteo de Sousa Jr., Alexandre Carlos Vilarinho de Oliveira,
Regina Tsujiguchi, Vinícius Maia Pacheco, Cláudia Jacy Barenco Abbas, Ricardo Staciarini Puttini

Universidade de Brasília, Departamento de Engenharia Elétrica,
Laboratório de Engenharia de Redes de Comunicação (UnB/ENE/LabRedes),
Caixa Postal 4386, 70919-970 Brasília – DF, Brazil
desousa@unb.br, alexandre@redes.unb.br,
Regina.Tsujiguchi@stj.gov.br, Vinicius.Pacheco@stj.gov.br
barenco@redes.unb.br, puttini@unb.br

Abstract. This paper describes a laboratory environment and its WWW access and management application that together constitute a virtual laboratory developed to allow distance teaching/learning of QoS through the testing and experimentation of DiffServ/MPLS QoS configurations. The laboratory is composed by personal computers with Intel processors running the RSVP-TE daemon for DiffServ over MPLS under the Linux operating system. This paper describes the construction, configuration and optimization of this Virtual Laboratory as well as some theoretical concepts and real experiments implemented during the project. Motivations for the creation of this laboratory include its low cost, which is an important factor in a teaching environment, and the access to the environment from Internet web browsers allowing users to remotely learn about QoS.

1. Introduction

Due to the growing demand for resources like bandwidth for services that are delay sensitive, like videoconference and VoIP, Quality of Service (QoS) is a way to guarantee the perfect functioning of those services that are becoming more common in our daily lives.
QoS refers to the overall performance perceived by an end user or an application across a network such as the Internet. QoS is thus related the network capability of providing a better service to selected network traffic over various technologies, including Frame Relay, ATM, Ethernet, specially in IP-routed networks that may use any or all of these underlying technologies. This way, the primary goal of QoS is to provide traffic priority, including dedicated bandwidth, controlled jitter and latency, and improve traffic loss characteristics. Also important is making sure that providing priority for one or more flows does not make others to fail.
Learning QoS is now a very important aspect for networking professionals. Configuration, test and traffic verification of different QoS configurations are key points in a networking teaching environment. The Virtual High-Speed Network Lab presented hereafter supports these teaching/learning activities. This laboratory environment is built on a GNU/Linux environment, because is a free software so its implementation costs can be reduced, making it accessible to any institution that has at least two computers. The QoS tools that made possible the creation of this lab are based on a project developed on the Department of Information Technology of the University of Gent [1].
Section 2 of this paper presents the laboratory infrastructure and software architecture. In section 3, are presented some of the experiments supported by the environment. Section 4 presents the WWW application that allows distance learning based on the virtual lab. Section 5 presents the conclusion and future works.

2. Building a Virtual High-Speed Network Laboratory

Some personal computers (hosts) are necessary to build the virtual high-speed network laboratory, including network nodes and network clients.
Each network node host needs three network cards such as Ethernet or another layer 2 technology being used. The basic host configuration is a 300 Mhz PC, with 128 MB of RAM and 6GB of HD, under the Linux operating system.

[1] This work is sponsored by CNPq, the Brazilian government agency for scientific and technological development.

Besides the four hosts needed for implementing a QoS controlled network, there are at least two clients hosts under the Windows operating system, to allow videoconferencing experiments.

2.1 Physical Installation

An experimental backbone is needed to interconnect all the hosts, according to the reference topology shown in figure 1.

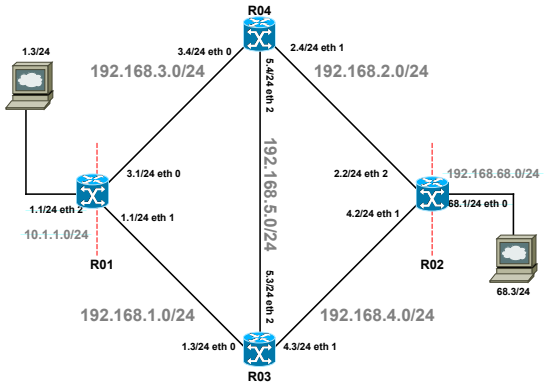


Fig 1. The Virtual High-Speed Network Laboratory Reference Topology

2.2 Software Installation

In each host, the installation of the operational system GNU/Linux Red Hat 7.2 with the 2.4.19 kernel is organized in order to maintain the maximum of the system's resources available to the virtual laboratory so the kernel can enable the DiffServ and the MPLS. To fulfill this approach, only a minimum set of packages is installed in the hosts and all the other unnecessary services are disabled [1].
Besides the operating system, the following programs and packages are needed:

- the zebra package [2], so the dynamic routing is enabled;
- the iproute2 [3], so the DiffServ can be implemented;
- the RSVP-TE deamon for DiffServ over MPLS under Linux [1], so the DiffServ can be implemented using MPLS shim header;
- the Ethereal package [4], to confirm that the MPLS shim header are added between the layers two and three headers;
- the MGEN [5], so that a congestion can be simulated and the packets which cause the congestion can be analyzed;
- the TRPR [6], so that the MGEN file, which contain all the data to analyze the congestion, can be prepared for gnuplot[7].

All the necessary programs are freely available on the Internet and can be easily obtained. To boot the DiffServ-MPLS kernel, a specific configuration file for the operating system initialization has been produced. To complete the software installation, the apache web server [8] is installed, allowing the execution of the virtual laboratory environment application, which was developed as a new web based user interface for the Virtual Laboratory.

3 Testing and Experimenting the DiffServ Backbone

The DiffServ over MPLS Backbone allows the various experiments that have been performed and verified. Initially, a MPLS tunnel was created for verifying the MPLS header, the shim header. Then the quality of service applied to the queues (using the TC tool) were tested by mapping a video stream into a tunnel with no guarantee (Best Effort PHB) and then mapping it into an Expedited Forwarding tunnel to verify if the queues were guaranteeing the differentiated service.
For mapping the traffic into some PHB, it was used the E-LSP, so the EXP field of the shim header defined the treatment to be taken for each flow. The experiment is performed using the following software:

- MGEN: to generate UDP flows at different destination ports.
- NetMeeting: for generating the video stream and testing the videoconference quality according to the PHB applied.

One tunnel was created and six UDP flows were mapped into the same tunnel, but each one in different PHBs. Table 1 shows the tunnel mapping.

Table 1. Tunnel Mapping

PHB	Destination Port	EXP
EF	5001	1
AF11	5002	2
AF12	5003	3
AF21	5004	4
AF22	5005	5
AF31	5006	6
AF32	5007	7

No MGEN traffic needs to be generated at the BE PHB, because the videoconference stream is already mapped into this PHB (EXP=0). After the MGEN traffic is initiated, the video observed at the clients freeze, due to the high packet loss. Then this experiment confirms that the BE PHB isn't recommended for transmitting this kind of traffic. The Figure 2 presents the result observed at the NetMeeting user interface.

The same test can be performed but this time the video stream is mapped into an EF PHB (EXP=1). After the MGEN traffic is initiated, the video quality remains excellent with no images freezing or distorting. Figure 3 presents the result obtained at NetMeeting with the EF PHB. Then, this experiment shows that this PHB (EF) is the ideal for carrying the videoconference traffic, because it guarantees the required bandwidth with low packet loss.

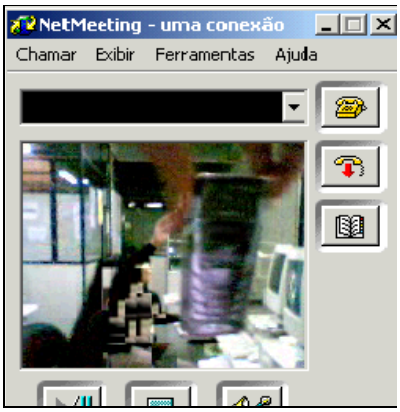


Figure 2. Videoconference mapped into the BE PHB

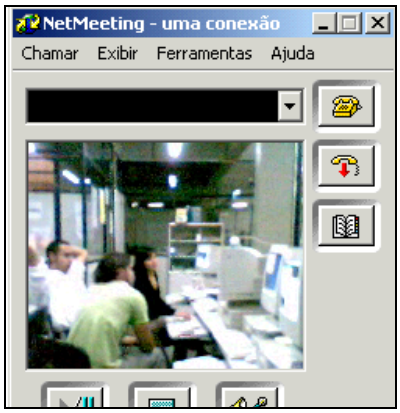


Figure 3. Videoconference mapped into EF PHB

After these experiments, it is possible to observe that the queues created by the TC Linux tool were applying the correct priorities for each configured PHB. Also, it can be verified that the MPLS technology is effectively being used for forwarding the packets through the Backbone.

Other similar experiments can be performed with the environment, including experiments using the WWW application as described bellow.

4 The WWW Graphics Interface

With the implementation of the described environment, experience with managing and manipulation of the experiments showed this was a very complex activity. It became clear, then, that a WWW graphic interface would considerably ease the tasks mentioned and would also provide means to offer a distance education laboratory to teach the techniques involved in configuring and managing QoS parameters in high speed networks.

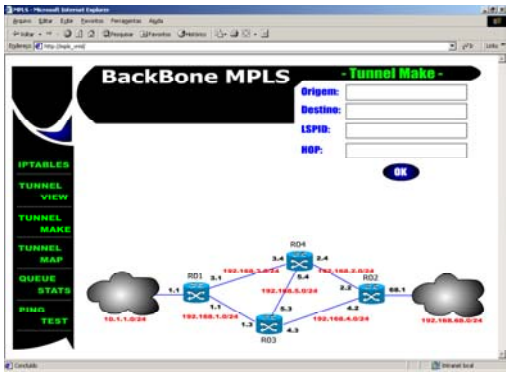
For the creation of such graphic interface, the following softwares were chosen: Macromedia Flash 5.0 for the graphics and Perl language for the programming involved. This choice was made because the two applications work well together and due to that they offer simple solutions to complex tasks.

In a machine directly connected to the backbone shown Fig 1., a Windows 2000 Server was installed and Flash 5.0 and Perl were configured to publish the web site through an Apache web server. This web site implements the graphic interface which structure can be seen in the Figure 4.

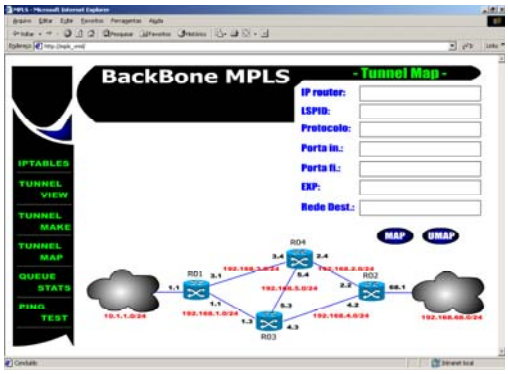
As can be observed in the Figure 4, the graphics interface offers multiple options on its menu. Each option provides a different functionality that enables the user to interact with the backbone. The options are:

- IPTABLES: using this option it is possible to verify how the iptables tables are in each router, showing how the rules are influencing the switching of packets that are being routed and how these packets are being marked with the QoS parameters.
- TUNNEL VIEW: through this option, the MPLS tunnels created and their EXP mappings can be visualized in each router. The transmission statistics of each tunnel can also be viewed here.
- TUNNEL MAKE: this option allows the creation of MPLS tunnels, specifying the origin, destination, id of the tunnel (LSPID) and also the specific path of the tunnel, the hops that will be used.
- TUNNEL MAP: with the MPLS tunnels created, the necessity of mapping the packets through them appears. This way, a chosen protocol can be mapped to the tunnel, enabling a type of PHB to be associated with it so that its packets are forwarded respecting a differentiated treatment in relation to the discard in the high traffic occasions.
- QUEUE STATS: this option was created in order to observe how the queues are behaving in the routers. Here, in the high traffic occasions, it is possible to view if the loss priority of the packets in the tunnels is being respected.
- PING TEST: to allow a transmission test of packets, this option was idealized. Here the user can specify where the ping will be originated and which router will be its destination. This way, it is possible to test if the tunnel created with the ICMP protocol mapped is having its packets forwarded correctly.

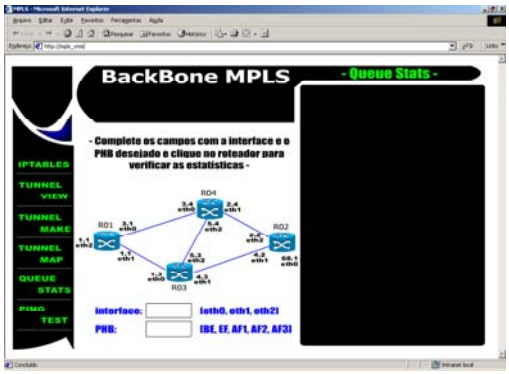
All options described above use the Flash to capture user options at the user interface. Perl modules are used to execute the corresponding tasks of consulting the routers and configuring them.



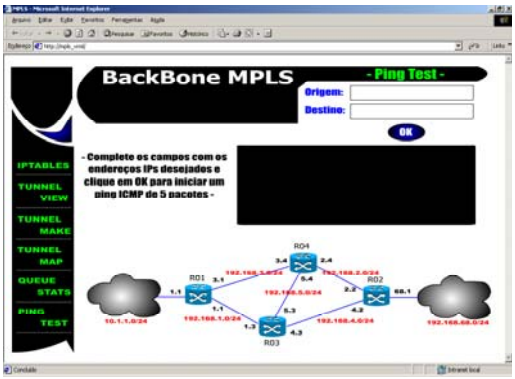
4-a: Defining a tunnel



4-b: Mapping a tunnel to a service



4-c: Verifying tunnel traffic



4-d: Verifying tunnel end points

Figure 4. Structure of the WWW graphics interface

5 Conclusion and future works

With the implementation of the Virtual Laboratory, several teaching institutions and universities can provide courses about QoS environment and its applicability, allowing institutions with only one computer and access to the Internet to use this lab to teach about QoS.
The virtual laboratory also stimulates teaching and research activities in the field of QoS environment, providing a cheaper solution to access an expensive technological tool. As such, the implemented virtual laboratory is now being used in the Network Engineering Laboratory at the University of Brasília - Brazil.
A research project presently under execution deals with the automatic discovery of QoS nodes in the network environment so as to allow the use of different and dynamic topologies in the QoS lab. Also, the graphics interface is being extended to allow the management of a heterogeneous QoS controlled network.
These developments in a free software framework are very interesting from the point of view of costs. High performance commercial systems are very expensive, including not only purchase costs, but also support and maintenance costs. The Virtual Lab presented here has a low cost hardware maintenance and this cost can be shared by a group of institutions that use the environment.
There are yet very few tools that can be freely used in the area of quality of service and the available tools lack good technical support. There is a demand for qualified people that are able to create these tools and to provide efficient support. A virtual laboratory for teaching the techniques of quality of service can be a helpful tool and provides an adequate cost model to fulfill this demand.

References

[1] http://dsmpls.atlantis.rug.ac.be/files/installation_rsvpd-0.70-rc2.txt. RSVP-TE daemon for DiffServ over MPLS under Linux Internet site accessed on September 9, 2002
[2] http://www.zebra.org/. GNU Zebra Internet site accessed on September 9, 2002
[3] http://diffserv.sourceforge.net/. Differentiated Services on Linux Internet site accessed on September 2, 2002
[4] http://www.ethereal.com/. The Ethereal Network Analyzer Internet site accessed on September 12, 2002
[5] http://manimac.itd.nrl.navy.mil/MGEN/. MGEN - The Multi-Generator Toolset Internet site accessed November 12, 2002
[6] http://proteantools.pf.itd.nrl.navy.mil/trpr.html. TRPR - *trpr* 1.9b8 User's Guide Internet site accessed November 12, 2002
[7] http://www.gnuplot.info/. Gnuplot Central Internet site accessed November 20, 2002
[8] http://www.apache.org. Apache Software Foundation Internet site accessed January 30, 2003