Educational improvements applying an MPLS network simulator: a technical approach

M. Domínguez-Dorado, F. J. Rodríguez-Pérez, J. Carmona-Murillo, J. L. González-Sánchez

***Abstract* — Nowadays, multiple services are being offered over the Internet next generation infrastructure. Most of these services are causing important social changes: online collaboration, e-commerce, e-vote, knowledge sharing, etc. Future technical leaders should learn about the impact of their activities on the society and not only about the usual economic point of view. These principles must be instilled into future engineers by their educators, who should have a suitable set of teaching tools which allows them to improve their educational methods. As MPLS is one of the most important technologies that support the integration of the aforementioned Internet services, in this work, we present an MPLS teaching tool and an evaluation of its influence when educating future engineers in an academic environment.**

***Index terms* — simulator, MPLS, teaching experience, engineering education, dynamical interaction, networking tool.**

I. Introduction

MPLS (Multiprotocol Label Switching) is a connection-oriented technology that arises to palliate the problems that current networks have related to speed, scalability and traffic engineering [1]. Simultaneously, it offers end-to-end QoS (Quality of Service) [2] by means of flows differentiation and resources reservation. Moreover, it eliminates the problem of managing different control planes that take place in IP/ATM networks, providing mechanisms to achieve the convergence of both technologies.

MPLS acts as link between network protocols and the corresponding link-level protocol. To do it, the MPLS header is located after the network header and before the link header [3] in the frame structure. In fact, MPLS packets forwarding is based on labels and not in the analysis of encapsulated data from upper levels. It is a multiprotocol technology that supports any network protocol as well as any technology in lower layers (link or physical). In this way, an attractive mechanism has been provided to take advantage of present infrastructure in backbone environments, making easy the migration between technologies; however, the efforts performed for years to develop innovative mechanisms that offer support to IP over ATM, have not been lost. Most of the developed techniques are still valid to have IP over MPLS and MPLS over ATM.

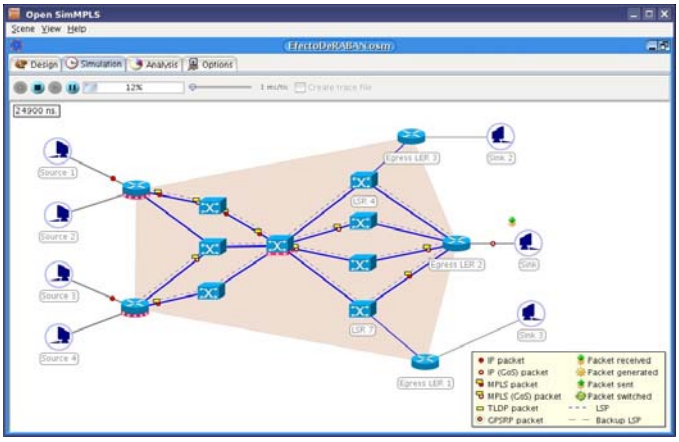

Fig. 1. General view of a scene window where flows with different GoS levels coexist.

In this work we show up an MPLS network simulator called OpenSimMPLS. It is a functional and visual tool (Fig. 1) that can be used in network and/or communications teaching. It considers the main operational and setting aspects of an MPLS domain [4]; at the same time, it has been improved to be compatible with GoS-supporting (Guarantee of Service) domains [5]. An MPLS domain with GoS capability could be understood as an environment able to carry out local recoveries of discarded MPLS packets, together with the possibility of re-establishing LSPs (Label Switched Paths) in a local way [6], [7]. This feature can be offered to some specific and privileged flows, that needs a fast and reliable service. GoS also allows these flows to be prioritized with regard to those not characterized as GoS traffic.

In the following section a comparative between some common teaching simulators is made. In third section a brief description of the simulator visual environment is shown, as well as some other functional aspects. In fourth section we explain some important technical issues about its implementation. In fifth point, the applications of OpenSimMPLS in academics environments or preliminary research works are highlighted. In section six, we show recent results obtained after using the simulator to teach future

M. Domínguez-Dorado belongs to DISIT at Universidad de Extremadura. Avenida de la Universidad s/n. CP: 10071. Tlf: +34 607 417 860. Fax: +34 927 257 202. e-mail: mdomdor@unex.es.

F. J. Rodríguez-Pérez belongs to DISIT at Universidad de Extremadura. Avenida de la Universidad s/n. CP: 10071. Tlf: +34 927 257 195. Fax: +34 927 257 202. e-mail: fjrodri@unex.es.

J. Carmona-Murillo belongs to DISIT at Universidad de Extremadura. Avenida de la Universidad s/n. CP: 10071. Tlf: +34 625 943 308. Fax: +34 927 257 202. e-mail: jcarmur@unex.es.

J. L. González-Sánchez belongs to DISIT at Universidad de Extremadura. Avenida de la Universidad s/n. CP: 10071. Tlf: +34 927 257 195. Fax: +34 927 257 202. e-mail: jlgs@unex.es.

engineers. Finally, the article concludes summarizing the contributions of the simulator and the future work.

II. Related work

OpenSimMPLS is not the first effort carried out to develop an educational MPLS simulator. Some researchers have already made proposals in this way. For example, in [8] we can find an MPLS simulator that allows designing and setting MPLS domains and their components. One can run a simulation and perform statistical analysis of its results, everything from an educational point of view. Their main features are: it is a teaching simulator that allows an elementary statistical analysis of the network traffic; it has a visual editor to design scenes; it is a multiplatform software and, although of facto it is free-of-charge and easy to install (it is an runnable applet located at project's homepage), neither its source code is available nor it is a downloadable executable for a local execution. On the other hand, it doesn't allow interacting with the simulation at runtime or changing its language. Also, due to its educational approach, it has not been designed to work in real MPLS networks using current manufacturers' components.

MNS (MPLS Network Simulator) [9] is an NS2 (Network Simulator 2) extension to allow MPLS networks simulations. Therefore, their features are similar to those of NS2; that is: it is a teaching tool, allows a full statistical analysis of all the events happened during the simulation, it is multiplatform software, it is distributed under the terms of a free software license and it is free of charge. Some of its weaknesses are: it doesn't allow the user to interact at simulation-time to modify its behavior; that is, the simulation will operate as previously specified in the configuration file. In teaching tasks, this static behavior prevents the student to exploit the discovery learning mechanism. The installation of MNS+NS2 is not easy (it requires to compile and to install some libraries). The generation of simulation scenes is difficult due to the necessity of knowing TCL (Tool Command Language) for the definition of scenes and all their components. This last, confers to MNS+NS2 the property of being a complete tool to validate research results, but in teaching tasks, it originates a learning curve that makes difficult the design of practical sessions in a short time. This is a great problem, taking into account the difficulty of fitting these practical sessions into a teaching calendar at universities. As the previous simulator, it is not capable to work in real MPLS networks. Although it is broadly used in teaching, it is focused on networks research works.

In the case of OpenSimMPLS [10], in order to make easy its future use in different academic organizations, it has been developed as multiplatform software, licensed under the terms of GPL (General Public License). It incorporates the advantages of [8] and [9], but it also allows students to dynamically interact with the simulation; it is translated to other languages; allows visual designing of scenes and GoS technology simulation; it doesn't require installation. A comparison between the features of these three simulators is presented in Table 1.

In order to do more specific works (not educational), there are more complete tools; for instance, Totem [11] or OPNET [12] both directed to work in real environments.

TABLE 1
SIMULATORS COMPARISON

	MNS /NS2	MPLS simulator	OpenSimMPLS
Teaching tool	•	•	•
Interactive simulation			•
Statistics data	•	•	•
Multiplatform	•	•	•
Multilanguage			•
Visual design		•	•
Free software	•		•
Free of charge	•	•	•
GoS simulation			•
Real environments applicability			
Installation and execution easiness		•	•

III. OpenSimMPLS simulator

Simplicity is the main feature of the simulator's graphics user interface. It has tree differentiated areas: workspace, main menu and scenes windows. The first one is the main area where simulation of different MPLS scenes takes place. Main menu is located in the upper-left corner. It includes options related to file management (creating, saving and loading scenes to/from disk), windows layout and help.

Finally, scenes windows allow designing and analyzing particular MPLS scenes. Their structure is formed by several tabs, which will be described in following sections.

A. *Topology designing area*

The first tab is the designing area, in which the parameters related to the topology and configuration of the MPLS domain are set. The toolbar is composed of several icons which show the elements that can be inserted in an MPLS domain (LERs, LSRs, links...).

The first icon makes reference to the Sender, which is the node that generates network traffic in the simulator. The second one is the Receiver, which is the sink of this data flow. The third one represents an LER (Label Edge Router), used to assign labels to IP or MPLS packets. It also classifies them, establishes a path towards the destination host through the MPLS domain and, finally, allows labeled packets to ingress in the MPLS domain. The fourth icon is an LERA (Label Edge Active Router) that operates as an LER, but it also analyses the IP header to know if packets have GoS requirements, codifying in this case those requirements in the MPLS header [5].

A GoS marked IP traffic can only keeps those GoS parameters inside the MPLS domain if this flow has accessed through an LERA node. Next icon represents an LSR, which switches the traffic in the domain. It is a fast component, because it only checks packets label set by the ingress LER/LERA. An LSR node has not capabilities to be an MPLS

network edge router. The sixth icon makes reference to an LSRA (Label Switch Active Router) that switches MPLS traffic inside the domain. It also has the capability to perform packets retransmissions and LSP fails recoveries both in a local environment. It also has the capability of temporary packets storing. In this way, it allows satisfying local retransmission requests from another LSRA in the domain. Last icon is the Link which makes possible the connection between two nodes. All components in a simulation scene have to be connected using links. Traffic will flow through those links. Anyway, OpenSimMPLS incorporates errors control that allows the student to generate free-of-errors topologies.

B. Scenes simulation area

Once scene's topology design has been finished, we should use the Simulation Area. There, one can analyze its behavior in a visual way: traffic generation, congestions, link failures, etc. (Fig. 2). The simulation topology will be the corresponding to the scene specified in the designing area. The only difference in the structure of both areas (designing and simulation) is the set of icons showed in the toolbar. Now we can see icons to manage the simulation operation. Finally, we can start the simulation by clicking the first icon that shows a gear.

When a simulation is running, a progress bar indicates the percentage of the simulation in course. There is also a small counter at upper-left corner that shows the number of elapsed simulation nanoseconds. On the other hand, it is also possible to slow the simulation, what makes feasible to analyze simulation events in a more detailed way without the necessity of simulation stopping and restarting. To do it, one must use the slider in the toolbar.

Fig. 2. Different types of traffic packets being discarded.

Real-time simulation shown in the simulation area consists of the graphical representation of the internal values generated by the elements that compose the scene. Most of times, using both visual representation of the simulation events and statistical charts of the elements (links and nodes), is enough to understand the different events happened during the simulation; however, sometimes we need to have a numerical point of view to understand some complex situations. To achieve it, OpenSimMPLS is able to generate a human-comprehensible trace file in text format, containing all events that have taken place during the simulation; for instance, affected components, consequences, etc.

This way, a functional method which makes possible to check the simulation is provided. In order to generate this trace file, a click must be done on the checkbox called 'Create trace file'.

During the simulation, different scene elements automatically modify their visual appearance as needed. For example, LER and LSR nodes will change their color depending on their congestion level. This is a measurement of the amount of packets accumulated in the node buffer. Graphical representation of packets will allow knowing the kind of on-fly data traffic in the simulation window (classified according to their priority). It also informs about the quantity and types of traffics, flows speed, when and how the signaling takes place, established paths, etc.

Normally, packets are forwarded through the network, but they also can be discarded in congested nodes. In this case packets will be shown, graphically, falling down these nodes (Fig. 2).

During simulation, packets representation belonging to different types of traffic, as well as data flows forwarding, can be analyzed with the aid of the legend that is shown (optionally) at the lower-right corner (Fig. 3). The legend has demonstrated to be a great help to the students when they use OpenSimMPLS; they are able to analyze what is happening during simulations.

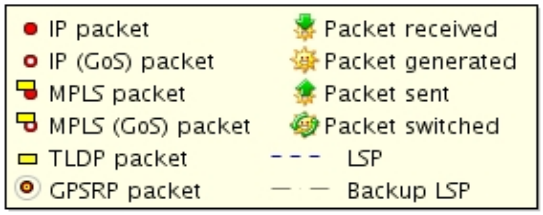

Fig. 3.Visual help to be used by the pupil during the simulation.

At this moment we have explained several visual signs that have to be interpreted to understand the events that take place during a simulation. However, simulation is an interactive environment and lots of actions can be issued. For instance, node congestion can be caused by clicking on the node. After that, the node will suffer great packets saturation (Fig. 4). If it continues receiving incoming traffic surely will begin to discard packets in a short time. This function is very useful to cause packets losses and consequent packets retransmissions without having to wait the natural node congestion.

Fig. 4. Artificial congestion of nodes.

In real conditions, a link has failure possibilities. Repairs, electric discharges, human mistakes, etc, can make a link to fail and then its traffic will be discarded. However, traffic overload does not cause a link failure. OpenSimMPLS can simulate that fact because it supports link failures anytime. However, it is not an event that happens during the simulation in a natural way, so it must be caused by hand. We will be able to simulate a link failure in a simulation with a click on the desired link. The link will change its appearance showing

itself as a dashed red line and will cause all on-fly packets to be lost.

C. Analysis of results area

We can go on to use the Analysis Area (Fig. 5) if we have set scene's components to generate statistical data; then we will see the charts that they are generating (or that they have already generated, if simulation has finished).

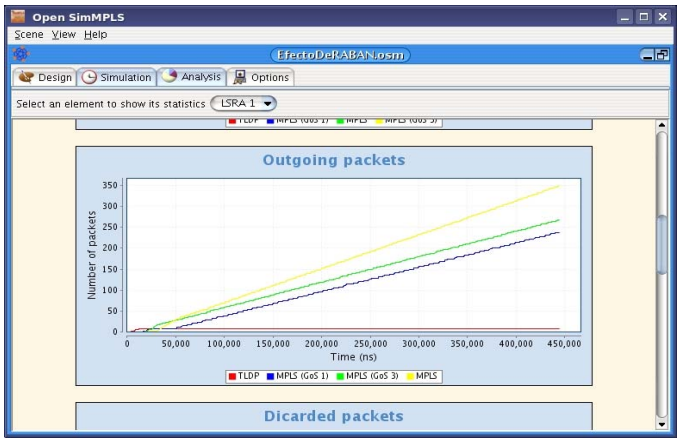

Fig. 5. Chart example. Analysis area.

This area is split up in two parts: a toolbar where we can choose the node or link to be analyzed and a bigger area where the statistical charts will be shown.

During a simulation, charts will have a dynamic behavior, changing according to the simulation advances. If it has already finished, charts show final results. Furthermore, charts generated by OpenSimMPLS are not static images, but they operate as interactive objects. One can obtain a pop-up menu just clicking on a chart using the secondary mouse button. This way, one will have access to different options, for instance disk saving of the image, interesting chart enlargement or printing, etc.

IV. Implementation Details

OpenSimMPLS is a standalone JAR application. Installation, therefore, does not require any difficult step, and we just only need to invoke OpenSimMPLS execution through the SUN's Java Virtual Machine, that should have been previously installed.

One of the OpenSimMPLS advantages is its portability, since it works independently of computer architectures or operating systems. That is why Java language has been used. Java has also allowed the implementation of the simulator as a multitask application (it is able to simulate more than a scene simultaneously), what is gotten by means of multithreading coding.

Moreover, Java is an object-oriented language. The simulator's main class, called *openSimMPLS*, starts the simulator execution. The *main()* method, belonging to that class, creates a *TDispensadorDeImagenes* object that will load all necessary images for the application. Then it will be passed as a parameter to the constructor of every interface elements, improving the performance. Later, a *JSimulador* object is created. It is the main application window. At this moment, simulation execution gives up being a sequential process. From that moment, everything is controlled by listening to the user events generated by the application interface: mouse orders, menu options, etc.

During the simulation, a *clock* sends signals to the scene elements (links and nodes) as temporization events (Fig. 6) or ticks.

The *clock* component is configured using two values: first, the simulation length (total number of ticks) and second, the ticks duration. The *clock*, which is executed into its own thread, will advance from the beginning until it reaches the maximum number of ticks defined for the overall simulation. Every tick will be sent to all scene elements, which are synchronized, so when it reaches the end, the thread will be stopped and the simulation will finish.

When the different topology components receive a tick, they also get its duration (number of nanoseconds). Then, every component activates its own execution thread, so it is generating concurrent processes. Each thread will carry out a task, depending on the device type; for instance, switching, packets forwarding, etc. Each component execution thread will stop when the time conferred on it by the received tick is used up.

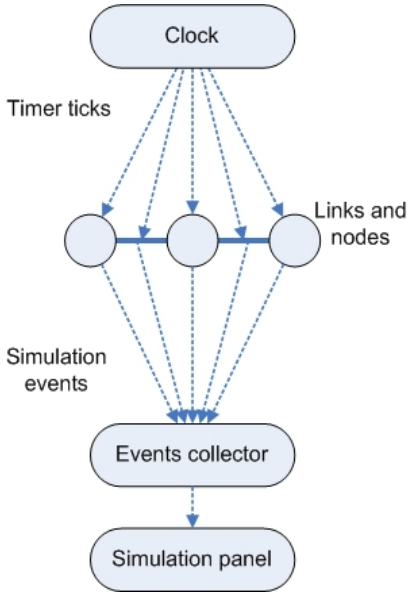

Fig. 6. General operation of a scene simulation.

When every component has used their tick up, the clock detects it and generates the next tick to repeat again the process. The class that implements the clock is denominated *TReloj*, and contains an internal list of elements. Every elements belonging to that list will get temporization events by the clock.

At runtime, lots of events happen. They must be collected in order to shown them in a visual form. This acquirement process is carried out by a scene global collector. All topology elements, will notify the collector of the task they are performing during their operation time. Although the tick generation is a discrete process, through its repetition, we get a fluid simulation. Therefore, the elements of the topology

(*TNodoEmisor*, *TNodoReceptor*, *TNodoLER*, *TNodoLSR*, *TNodoLERA*, *TNodoLSRA*, *TEnlaceInterno*, *TEnlaceExterno*), will encapsulate methods to carry out these notification tasks.

The global events collector is implemented by *TRecolectorSimulacion* class. This class has a method called *capturarEventoSimulacion()*, that allows the topology elements to send the simulation events generated during their threads execution time. However, the collector only gathers events but it doesn't show them. To show them graphically, the services of a simulation panel component must be used. Thus, the ability to isolate generation tasks and events gathering from its visual representation is achieved.

The simulation panel is implemented in *JPanelSimulacion* class that carries out tasks like screen refreshing, visual simulating, etc. That is, the events that arrive at the collector are interpreted and shown in the screen, making easy the user understanding. The method used to send the events to the screen is called *ponerEvento()*.

A. Scene's topology

The topology is an object needed to store all the components in the scene and to manage the links graph between the nodes and the links. It also establishes the associations between either the clock and the elements or the collector and the elements. The overall functionality is implemented in *TTopologia* class. To carry out these tasks, each element has to use a unique identification, which is assigned by the identifier generator that belongs to the topology.

B. Simulation scene

The scene is implemented as a *TEscenario* class that contains all components of a simulation environment; for instance: clock, collector, topology, identifiers generator, links, nodes, etc. This means that the overall components in a scene are stored in a single object.

The most important method in the scene is called *generarSimulacion()*. It starts the simulation after the topology clock is turned on. In fact, everything needed to operate is contained in *TEscenario* object, even the user interface doesn't exist. This means that one can create an event interpreter easily, where users with visual limitations could get an adapted interface. Even it is possible to separate events generation from their visualization and execute them in different hosts.

V. TEACHING AND RESEARCHING APPLICATIONS OF OPENSIMMPLS

The main goal of OpenSimMPLS is to be useful to university professors as a tool for teaching innovation [13]. It also allows analyzing the operation of MPLS networks via a multilingual and multiplatform system.

OpenSimMPLS will be shown in Spanish language, if that one is the operating system configuration, otherwise, the default simulator language will be English. OpenSimMPLS uses the Java internationalization system, so it is easy to translate the tool to another language. It also has an easy-to-use graphical interface. Each element is coded following the object oriented paradigm, and the generated processes operate by means of independent threads, in a concurrent way. The simulation is composed by three stages: MPLS domain design and configuration, visual representation of events, and generation of simulation statistical results.

Students of networking and communications subjects increase their learning thanks to practical examples. The simulator permits the students to observe the network behavior in particular situations; for instance, when the kind of used traffic is multimedia. It also allows comparing and contrasting the results thanks to the reconfiguration of domain elements. In this way, students can improve MPLS scenes, proposing new features to be added. It is also possible to detect either pernicious or beneficial effects on the traffic.

A discovery learning based method is performed in OpenSimMPLS, so students can solve their own problems and situations; learn technology procedures and understand events features. They also learn how to control these events and what to do in particular cases, thanks to the dynamical interactivity during the simulation.

OpenSimMPLS can be used to help students to reinforce their knowledge about the technology, for instance, when the students think in a hypothetical network situation, they can test it through the simulator results. In this way, the simulator is a test bed that returns feedback to the students (Fig. 7). This returned information must be analyzed to know what happens inside the MPSL domain, and how it is operating.

In summary, there is a double feedback process when the simulator is used in an academic centre: In one hand, the simulator execution gives the student the possibility of analyzing the MPLS scene behavior; in the other hand, after examining statistical results, several conclusions can be inferred by the student, who will can change the settings and restart the simulation. This repetitive process motivates the students to develop their own mental strategy about MPLS operation.

Hereafter we show the educational OpenSimMPLS capabilities, as well as guidelines for using it in a lab practical session. First of all, the teacher explains the theoretical concepts about MPLS technology, as well as their possible problems (nodes congestion, links failures, packets loss, etc.).

Next, the student can immediately start the practical session. Wasting time on explaining the simulator operation is not necessary due to the useful user manual that it enclose. In practical sessions, several activities will be proposed to the students:

- Well designed and optimum MPLS scenes.
- Scenes where exist common problematic situations.
- Scenes where is necessary the student interaction, causing links failures or node congestions.

In the first case, scenes will make the students to reinforce their theoretical concepts. In the second point the students will analyze situations where, although there is not network failures, it performance can be improved by way of reconfiguring topology components or changing the profile of generated data flows. In the last case, the student will have the

opportunity to detect and to analyze the consequences of sporadic network failures. The student will be able to check how the introduced variations affect to the final system performance [14].

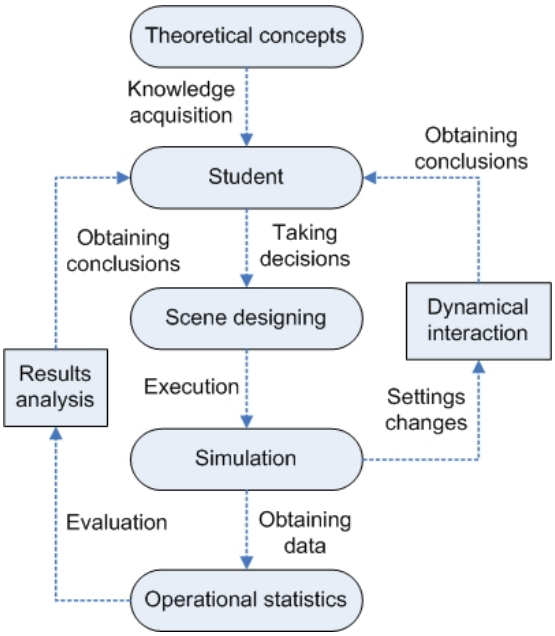

Fig. 7. Double-feedback process followed by the pupil when using OpenSimMPLS simulator.

OpenSimMPLS is an interactive application what means that the student becomes an active part during the simulation (modifying properties of the scene in a dynamic manner), developing this way the experimental knowledge and the discovery learning [15], [16].

Our tool is a solution to make the operation of an MPLS domain familiar to students of networking subjects. The student can research into communications issues and size up MPLS network resources, avoiding unnecessary risks of modifying the configuration of real devices in the lab local network. On the other hand, and as collateral goal, the statistical results obtained will be useful to the student to develop new methodologies in its future work as next generation network planner. It allows acquiring a very methodical discipline: planning of a MPLS network, obtaining and analyzing results and successive refinements toward a final optimal architecture (Fig. 7).

The use of OpenSimMPLS reinforces the theoretical concepts of MPLS technology and offers the students the motivation they need to understand the interaction between different components of a scene.

To sum up, using the simulator as a teaching support to explain MPLS operation contributes several advantages:

- It is easy to install in a lab (it doesn't require database; it admits multiple architectures and operating systems).
- The use of the simulator as a teaching or a learning validation tool, will always suppose a cheaper solution than deploying a real MPLS domain in an academic laboratory.
- OpenSimMPLS allows modifying the settings of the scenes' components; after that, it let us to analyze the consequences of these changes and to learn of them. In a real MPLS network, at an academic lab, won't always be allowed to carry out configuration changes.
- Simulation allows the student to obtain detailed statistical data that can be used to examine particular behaviors of the MPLS domain.
- It is open source software. That means that it allows the educators to teach not only networking subjects, but also coding subjects, proposing to their students some changes to be implemented to the simulator in order to improve it or add new capabilities.

VI. Real OpenSimMPLS evaluation in the university

In our case, the simulator is used to gain teaching innovation in subjects belonging to the Telematics Engineering Area at Polytechnic School of Cáceres, such as 'Broadband communications' or 'Network planning, specification, designing and evaluation'. It is also used in subjects focused to train Ph.D. candidates, e.g. 'Multimedia, multiprotocol and heterogeneous networks integration with QoS' and 'Broadband, multimedia and multiprotocol communications with QoS and security' at the University of Extremadura. We have carried out a comparative study of the OpenSimMPLS use in 'Broadband communications'. The study compares the qualifications obtained by the students in 2005/2006 academic year (teaching without the use of OpenSimMPLS) and 2006/2007 academic year, when we used the simulator as a supporting tool. In Fig. 8 it can be observed that for 2005/2006 year the rate of passed exams was 58.82%, and 41.18% of failed exams. In the following year, coinciding with the incorporation of OpenSimMPLS, the percentage of passed exams grows to 66.67%, diminishing the rate of failed exams, therefore, to 33.33%.

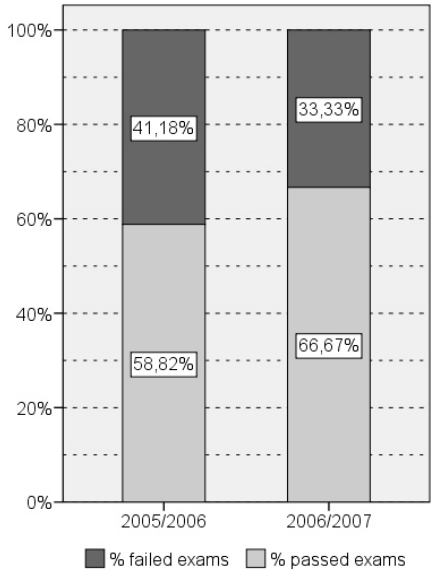

Fig. 8. Comparative between pupils who have passed the exam and those who have failed it. Years 2005/2006 and 2006/2007.

In Fig. 9 are also observed that the groups' average mark in 2005/2006 year was 4.67 on 10 and, in 2006/2007 year, that value ascended to 6.17 on 10. At the same time, standard deviation diminishes from 3.19 points in 2005/2006 to 2.71 points in 2006/2007, showing therefore a better homogenization of students' qualifications.

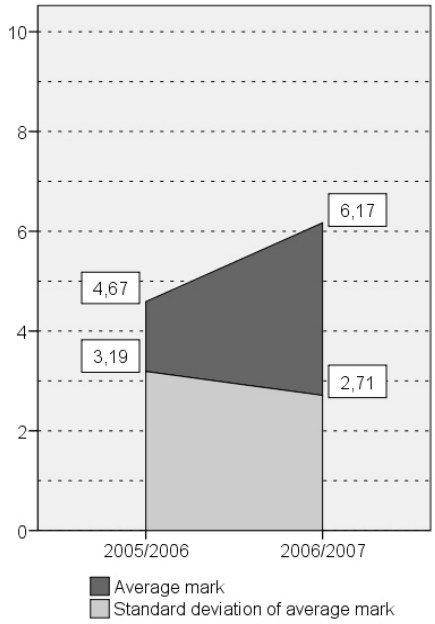

Fig. 9. Average grade variation and marks standard deviation for years 2005/2006 and 2006/2007.

Summarizing, in Fig. 10 the improvement obtained in 2006/2007 year when using OpenSimMPLS is quantified: on one hand, the increment of the average mark has been 32.07% and on the other hand, the decrease of standard deviation is 24.28%.

Fig. 10. Increment of Average grade variation and decrement of marks standard deviation between years 2005/2006 and 2006/2007.

VII. Conclusions and Future Work

The use of OpenSimMPLS to carry out professional works in real network environments is not recommended due to its teaching purpose (it doesn't incorporate real features of current manufacturers' components). However, the present work proposes the use of OpenSimMPLS as an educational tool to be innovative when teaching subjects belonging to Telematics Engineering Area, which is justified by the growing interest that the MPLS technology is waking up.

Also, the simulator is a supporting tool to those research projects related to MPLS (as a particular case, we have implemented the GoS support on it), as well as in teaching subjects related to this technology. Particularly we have proved the improvement of the results concerning to 'Broadband communications' group. We observed an increase in the students' motivation and interest, with an improvement of the average mark and also obtaining more homogeneous qualifications, as shown by the decrease of the standard deviation.

Its multiplatform philosophy and its free software license make easy its own evolution, since it can incorporate the received feedback from other users by means of the project's homepage or via the simulator 'Contact the authors' option.

Among its future possibilities, we are thinking about carrying out coding practices so that the students of subjects related to Telematics Engineering Area can develop modules, algorithms and additional network technologies on the simulator. On the other hand, OpenSimMPLS will be improved by adding IPv6 (Internet Protocol version 6) and RSVP (Resource Reservation Protocol) support and with the addition of new visual features intended to allow obtaining a more detailed and richer information.

Acknowledgements

This research work is sponsored in part by the regional Education, Science and Technology ministry belonging to the Extremadura Regional Government, by means of AGILA project numbered as 2PR03A090.

References

[1] M. Kodialam. T. V. Lakshman. *Restorable Dynamic QoS Routing*. IEEE Communications Magazine, Vol 40, Issue 6, June 2002, pp 72-81.

[2] J. Gozdecki, A. Jajszczyk, R. Stankiewicz. *Quality of Service Terminology in IP Networks*. IEEE Communications Magazine, Vol 41, Issue 3, Mar 2003, pp 153-159.

[3] E. Rosen et al. *Multiprotocol Label Switching Architecture*. RFC 3031, January 2001.

[4] M. Domínguez-Dorado, F. J. Rodríguez-Pérez, J. L. González-Sánchez, A. Gazo. *Multiplatform and Opensource GoS/MPLS Simulator*. II European Modelling and Simulation Symposium (EMSS2006). International Mediterranean Modelling Multiconference (I3M2006). Barcelona, 2006, pp 529-537.

[5] F. J. Rodríguez Pérez, J. L. González Sánchez, A. Gazo Cervero. *RSVP-TE extensions to provide guarantee of service to MPLS*. IFIP Networking 2007. Atlanta (USA) May 2007.

[6] J. L. Marzo, E. Calle, C. Scoglio, T. Anjali. *QoS Online Routing and MPLS Multilevel Protection: A Survey*. IEEE Communications Magazine, vol 41, Issue 10, Oct 2003, pp 126-132.

[7] G. Ahn, W. Chun. *Simulator for MPLS Path Restoration and Performance Evaluation*. Chungnam National University, Korea, 2001, pp 32-36.

[8] MPLS Simulator: http://www-entel.upc.es/xavierh/mpls (07/05/2007).
[9] G. Ahn, W. Chun *Design and Implementation of MPLS Network Simulator*. 15th International Conference on Information Networking, 2001, pp 694.
[10] OpenSimMPLS: http://gitaca.unex.es/opensimmpls (07/05/2007).
[11] http://totem.info.ucl.ac.be/index.html (07/05/2007).
[12] http://www.opnet.com (07/05/2007).
[13] S. H. Thomke. *Simulation, learning and R&D performance: Evidence from automotive development*. Research Policy, Volume 27, Issue 1, May 1998, pp. 55-74.
[14] T. J. Overbye, P. W. Sauer, C. M. Marzinzik, G. Gross. *A user-friendly simulation program for teaching power system operations*. IEEE Transactions on Power Systems, Vol 10, Issue 4, Nov 1995, pp 1725-1733.
[15] T. de Jong, W. R. van Joolingen. *Scientific Discovery Learning with Computer Simulations of Conceptual Domains*. Review of Educational Research, Vol. 68, Issue 2 (summer, 1998), pp. 179-201.
[16] A. Parush, H. Hamm, A. Shtub. Learning histories in simulation-based *teaching: the effects on self-learning and transfer*. Computers and Education, Vol. 39, Issue 4, Dec 2002, pp. 319-332.

Manuel Domínguez-Dorado, Zafra (Badajoz). Spain, 1977. He got his BS and MS degree in Computer Science at University of Extremadura, 2004. Now, he is a Ph.D. candidate and FPI grant holder at Telematics Engineering Area, Computing Systems and Telematics Engineering Department (UEx). His areas of interest are MPLS-TE, QoS routing and inter-domain routing.

Fco. Javier Rodríguez-Pérez, Huelva. Spain, 1976. He got his MS degree in Computer Science at University of Extremadura, 2000. Now, he is professor at Telematics Engineering Area, Computing Systems and Telematics Engineering Department (UEx). His areas of interest are QoS routing and Guarantee of Service over MPLS-TE.

Javier Carmona-Murillo. Badajoz. Spain, 1982. He got his BS and MS degree in Computer Science at University of Extremadura, 2005. Now, he is a Ph.D. candidate and a researcher at Telematics Engineering Area, Computing Systems and Telematics Engineering Department (UEx). His areas of interest are QoS and IPv6 mobility support.

José Luis González Sánchez, He got his Ph.D. in Computer Science at Technical University of Catalonia. He is Lecturer Professor and Telematics Engineering Area coordinator at Computing Systems and Telematics Engineering Department (UEx). He is the main researcher of GITACA research group. His areas of interest are QoS, MPLS-TE and security in communications.