MULTIPLATFORM AND OPENSOURCE GoS/MPLS SIMULATOR

Manuel Domínguez-Dorado, Francisco J. Rodríguez-Pérez, José L. González-Sánchez, Alfonso Gazo-Cervero
Computer Science Department, University of Extremadura
Avda. de la Universidad s/n, CP: 10071 Cáceres, SPAIN
E-mail: {mdomdor, fjrodri, jlgs, agazo}@unex.es

KEYWORDS

MPLS, path recovery, MPLS simulator, network prototype, performance evaluation.

ABSTRACT

MPLS (Multiprotocol Label Switching) technology provides interesting elements to integrate network technologies like ATM (Asynchronous Transfer Mode) and IP (Internet Protocol) with QoS (Quality of Service) and traffic engineering. It is an advanced proposal which presents great interest for researchers into networks and communications. In this paper we present a simulator called OpenSimMPLS; it is a tool for planning, designing and evaluating MPLS networks. It allows the same operations to be performed over GoS (Guarantee of Service)/MPLS networks, an improvement that incorporate into MPLS the capability of recovering discarded packets and re-establishing broken LSP's (Label Switched Path), both locally. The simulator makes possible that researchers can configure, interact and analyze the operation of a MPLS domain in an easy and efficient way. On the other hand, due to its freeware and open source license, it might be used by researchers as protocol engineering platform, to carry out analysis, and as a tool to validate results. Its source code is opened so everybody can extend the simulator by adding support for new protocols or techniques.

INTRODUCTION AND RELATED WORK

MPLS (Rosen et al, 2001) is a technology that arises to palliate the problems that raise the present networks as far as speed, scalability and traffic engineering. At the same time, it offers end to end QoS (González-Valenzuela and Leung, 2002; Kodialam and Lakshman, 2003) by using flows differentiation and resource reservation. In addition, it eliminates the problem of manage several control planes that take place in IP/ATM networks, providing mechanisms to easily obtain the convergence between both technologies.

MPLS acts as nexus between link and network layers. To do it, in the structure of a frame the MPLS header is located after the network header and before the link header. Currently, the MPLS packets forwarding is based on this header, called MPLS label, and not in the analysis of the data that it encapsulates (Rosen et al, 2001b).

It is a multiprotocol technology that can simultaneously carry any network protocol and support any technology in lower layers (link or physic). Therefore, MPLS involves an attractive mechanism to take advantage of the current backbone infrastructure, facilitating therefore the migration of technologies. In addition, the efforts made during years in the development of innovative proposals for IP over ATM could be re-used, since most of the researched techniques are valid to accommodate IP over MPLS and MPLS over ATM.

Due to the increase and the relevance of this technology, tools that facilitate its adoption are needed; for example simulators (Ahn and Chung, 2001b), control and management tools, security tools, and so on. In this work an MPLS network simulator, called OpenSimMPLS, is presented. It is a functional and visual tool that considers the fundamental configuration and operation aspects of a MPLS domain (Cavendish, Ohta and Rakotoranto, 2004). In the same way, the simulator has been extended to support MPLS domains with GoS (Domínguez-Dorado, Rodríguez-Pérez, González-Sánchez, Marzo and Gazo, 2005). Among others features, It support path protection (Huang and Sharma, 2002) and lost packets recovery.

In the following section, the interface of the simulator as well as the most important functional aspects that make it interesting, are described. In the third section, general details concerning with its implementation and its extension possibilities are commented. The article concludes emphasizing two aspects: on the one hand, the contributions of OpenSimMPLS as a platform to validate research results, and on the other hand describing new features that are being built in the simulator and those that are being planned to incorporate soon.

OPENSIMMPLS OVERVIEW

OpenSimMPLS has been planned as a tool to allow MPLS and GoS/MPLS networks simulation (Ahn and Chung, 2001). This characteristic allows using the simulator in two different ways: on the one hand, it allows testing and evaluating MPLS networks with its basic characteristics, for example the label signalling or

the LSP establishment; and on the other hand it allows to experiment with advanced concepts like, traffic load balancing, local packets recovery or LSP re-establishment (Marzo, Calle, Scoglio and Anjali, 2003). The operation of the simulator is governed by three stages, which are implemented as well in three separated zones in the application: GoS/MPLS domain design and configuration, interactive simulation and analysis of results. Also it is needed to consider a pre-stage in which the length of the simulation, the frequency of sampling, and so on, will be set up.

Simulator Zones

In order to make the use of the simulator easy, three zones have been designed (Figure 1): first, the menu bar, in the upper margin of the application window, which contains the required options to manage the open or file-stored scenes and to arrange child windows that are opened. Second, the working zone, which is the rest of the main window, where every MPLS scenes simultaneously opened, will be located. Third and last, scene windows which contain all they need to completely designing, simulating, configuring and analyzing GoS/MPLS domains. It is possible to work simultaneously with more than a scene, which allows visualizing and contrasting its parallel evolutions.

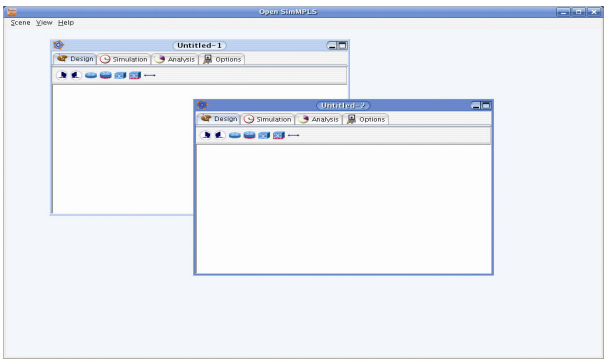

Figure 1. General View of Simulator Zones

Once a new scene is created, a scene window appears in the simulator working zone. From that moment, every heavy configuration work, design, evaluation or simulation of the new scene is made from that window. Scene windows contain four tabs (Figure 2) that correspond to the design, simulation, analysis and timing options, previously explained.

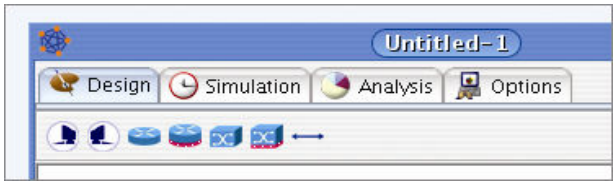

Figure 2. Tabs of the Scene Window

GoS/MPLS Domain Design

'Design' tab allows inserting GoS/MPLS domain elements so that, element by element, the domain on which it is desired to make the simulation and analysis could be created. To it, this tab has a set of icons that represents each one of the elements to insert in the topology in creation (Figure 3).

Figure 3. Icons of GoS/MPLS Elements

From left to right, these icons represent a source node, a sink node, a LER (Label Edge Router) node, a LER node with GoS support, a LSR (Label Switch Router) node, a LSR node with GoS support and a link, used to physically join two nodes.

For all the available elements, the work method is the same one: first of all the element must be added to the scene or domain to be simulated; at this moment it is necessary to enter some minimum data, e.g. the name of the element, so that the simulator can operate; After the element insertion, if it is not desired to use the default configuration, the element's detailed options should be configured.

In order to insert each one of the elements in the scene, a click must be performed over its icon. The simulator verifies that the current scene allows adding the selected element; if not, OpenSimMPLS will show the corresponding warning, whereas if there is no problem, it will show the element's general configuration window (Figure 4).

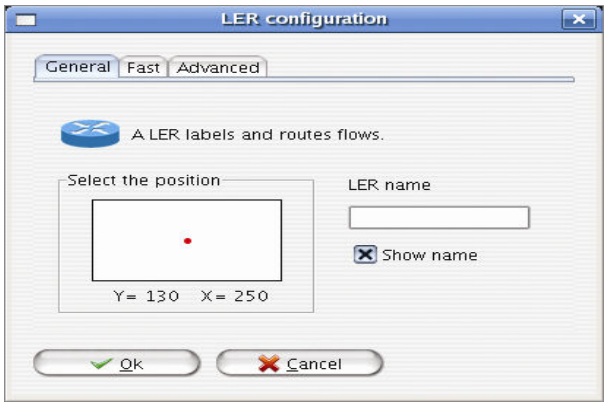

Figure 4. Example of Initial Element Configuration

After accepting, the element will be inserted in the scene, which can be shown in the white canvas of the scene window. Elements already inserted support the

drag and drop paradigm if the mouse left button is used. That allows easily modifying the element position. In the same way, a contextual menu of each inserted element (which contains options such as the advanced properties) can be obtained by clicking on the element; to it, the mouse right button must be used. For example, one of the properties that can be activated for each element is the desire of generating statistical data or not; this means that during the analysis phase the possibility of examining the statistical data of only elements configured for it, exists, making easy the analysis and saving memory.

Besides, a menu with options can be obtained anywhere on the scene by clicking with the mouse right button on a blank area of the domain representation that does not contain elements.

Following this procedure, a GoS/MPLS domain can be completely built in a simple way, free of errors (Figure 5). At this moment it is possible to finish the scene design and configuration and follow setting up the simulation timing parameters.

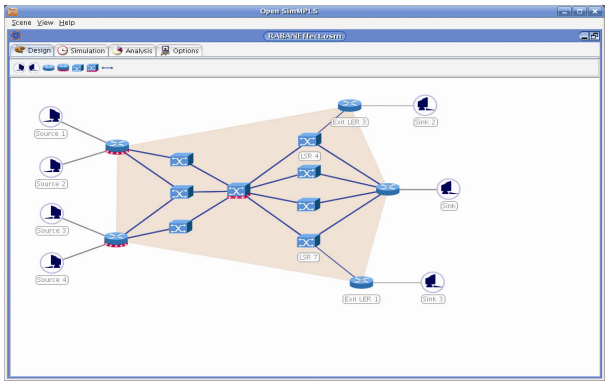

Figure 5. Complete GoS/MPLS Domain Design

Timing Parameters Configuration

In the 'Options' tab of the scene window, general characteristics can be configured on the simulation scene that is being developed: information about the scene like author's name, description and, what is most important, the length (in nanoseconds) and the sampling interval of the simulation (Figure 6).

Figure 6. Timing Parameters Configuration

The maximum value for the sampling interval is automatically established depending on the configured link delays of the domain and the length of the simulation. Making this interval longer, the granularity in the capture of happened events during the simulation is increased, which causes that the visual simulation is coarser and the charts generated in the analysis process contain less intervals; but at the same time less memory resources are needed, reason why this configuration is recommendable when running OpenSimMPLS in low performance hosts. For an optimal simulation this parameter should be tuned to a low value.

After tuning the timings parameters of the simulation, the scene is ready so that the simulation can start.

Interactive Simulation

The 'Simulation' tab contains everything to observe and manipulate the simulation of the scene. To it, it has two zones: the options zone and the simulation zone. The options zone (Figure 7) has a set of icons and components that allows, from left to right, the following operations: starting the simulation, finalizing the simulation, resuming the simulation, pausing the simulation, seeing the running progress of the simulation, slowing down the simulation and creating an analyzable trace file, that stores what is happening in the course of the simulation.

Figure 7. Simulation Options

The simulation zone is the white box that can be found at the centre of the scene window. The same topology that has been designed for the GoS/MPLS domain in the design stage will be shown on it. In order to execute the simulation one must click on the first icon of the option zone. Automatically the simulation will start. During the simulation many details are possible to be observed (Figure 8), e. g. on-fly traffic, the node congestion (indicated by different colours), the packets discards, and so on.

The simulator has been planned to be intuitive, reason why some details have been taken care specially. For example, the established LSP are painted with dashed lines, different kinds of traffic show different aspects, greater thickness are used to draw links having lower delay, another kind of dashed lines are used to the backup LSP, etc.

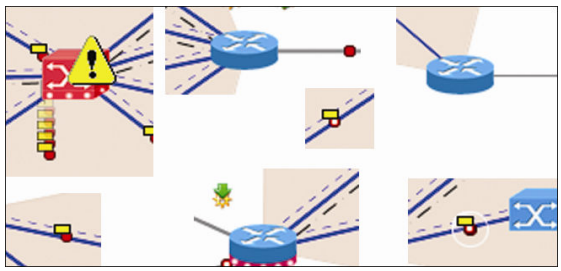

Figure 8. Simulation Details

If desired, the simulation runs on a continuous and fluid form without user intervention. However, the simulation can be interactive. On the one hand, it is possible to cause that nodes acquire a congestion level near 100% to evaluate the answer of the design that has been created to this kind of problems. Also, it is possible to cause a link to fail in order to observe how the LSP are re-established or how the backup LSP are activated. Both operations are made by means of a mouse click on the node or the wished link, respectively. In order to undo the caused effect, it is solely necessary to repeat the operation.

At any time, the user can cause an informative legend to appear, by clicking on some blank area of the simulation zone of the designed domain where there are not elements (Figure 9).

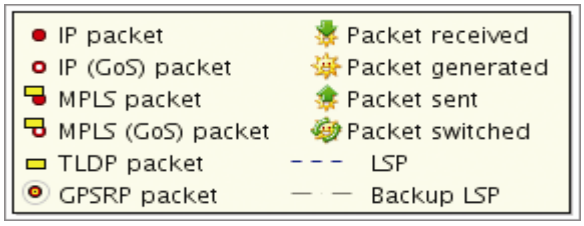

Figure 9. Legend

Finally, 'Create trace file', from the options bar, allows enabling or disabling the dump to disk of the simulation data. If it is activated, at the same time that the scene is being simulated, all the events taking place, which elements of the design and in what moment, are stored in a file, to be able to make a more detailed study of the simulation after finishing.

Analysis of Results

The last step consists of contrasting analytically that the results that have been observed in the visual simulation correspond to the statistical data gathered by the simulator. The 'Analysis' tab offers some statistical charts for each element. Charts will be available only for those elements configured for it, as commented in previous sections. Near the upper margin of this tab is the node selector. This selector allows choosing the node whose charts are required (Figure 10).

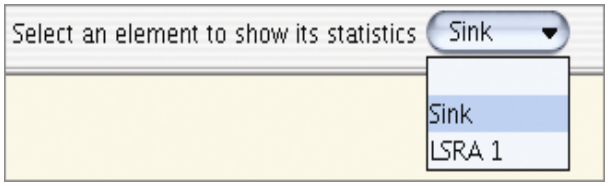

Figure 10. Node Selector

Some of the data shown in the charts are, depending on the kind of selected node: number of generated, discarded and received packets, retransmission requests, effective packet retransmission, and so on.

Once selected the wished node, charts associated to it are shown in the lower area of the tab (Figure 11). It is necessary to emphasize that charts are interactive and allow the user to zoom them, export them as a PNG (Portable Network Graphics) image or print them, among other options.

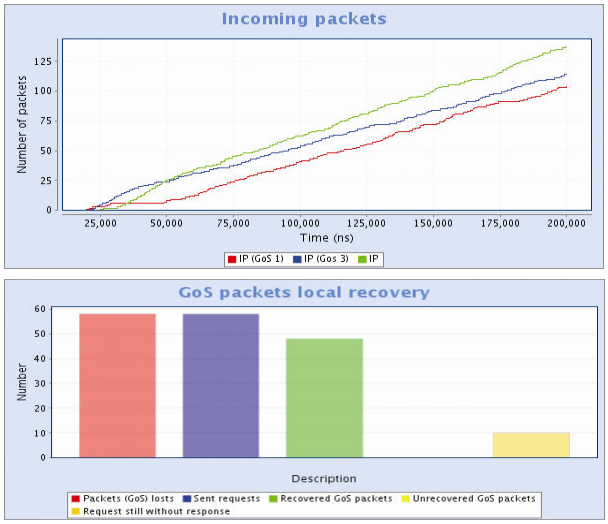

Figure 11: Example of Statistic Charts

Charts provided by OpenSimMPLS are generated by JFreeChart (Goh, 2006); opensource Java ™ software that has been included in the simulator and that allows generating advanced and robust charts. The JFreeChart API (Application Programming Interface) collects the needed data for charts generation from the events created by the simulator during the simulation of a scene. The generated charts are highly configurable, reason why it is relatively simple to adapt the form in which the data are shown. It is only needed to modify a few source code lines.

Information shown in the charts generated by the simulator is concrete and allows the analysis of the GoS/MPLS domain behaviour, e.g. a node shows the packets that arrive to it and the packets that it relays to another node during the simulation, in a graph of type XY. Each kind of traffic is represented as different series. Another data type, like total number of discarded GoS packets, is represented in a bar chart. If this information is joined to the provided one by the trace file and the visual simulation, a deep level of knowledge of what can have happened, how, when and why, should be obtained.

IMPLEMENTATION DETAILS

OpenSimMPLS has been implemented following the MVC model (Model View Controller) with Java ™. Whenever it is possible, the user interface is separated

of the application logic. In addition, its implementation is very modular so its maintenance and its improvements are easy. The simulator architecture, at very abstract level, follows a three modules scheme (Figure 12).

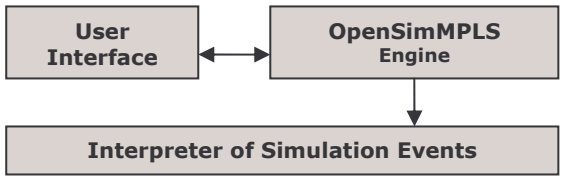

Figure 12. Simulator Abstract Architecture

To achieve this architecture, a wide class hierarchy has been created. An easy-to-use graphic environment has been created to allow the configuration of every parameter the simulator engine needs and the inspection of values they have in each moment. Once configured, the engine takes charge of generating the simulation. Really, with only a few changes, a complete simulation could be realised without need of graphics interface because all the application logic is implemented in this module. Finishing, in order to make the simulation process more friendly, a module that interprets every values generated by OpenSimMPLS engine, has been added. It shows these values on the screen in a visual and attractive form. At a lower abstraction level (Figure 13), the three modules are composed by some intercommunicated classes.

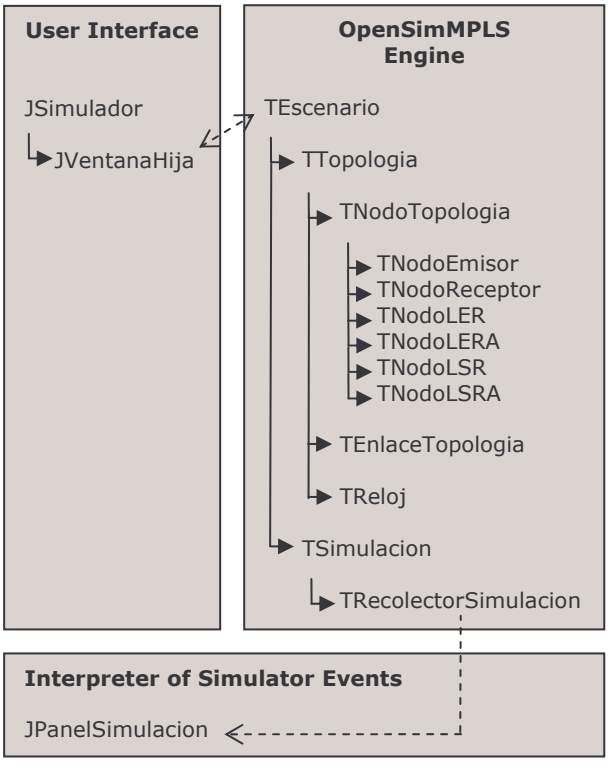

Figure 13. Simulator at Medium Level Architecture

These classes are joined to others, that are needed, but of less relevance so, due to space problems, they will not be detailed in this work.

User Interface

This module encloses everything that is related to the simulator graphic environment: windows, mouse and keyboard events, help, and so on. Its main class is *JSimulador*, which starts the application and creates a new simulator that will allow doing tasks related to the simulation. A *JSimulador* instance shows an empty main window and allows working with scenes, stored in disk or created at the moment. When a scene is created or opened from disk, it is loaded into a new child window of type *JVentanaHija*. This class, *JVentanaHija*, allows creating, modifying, configuring and executing scene simulations and will be just a communicator between the class containing the scene and the OpenSimMPLS user. Information managed by *JVentanaHija* instances is bidirectional since on the one hand the user must give instructions to the simulator but at the same time the graphic user interface must inform the user about everything happening, in a feedback process.

OpenSimMPLS Engine

Each *JVentanaHija* instance has a reference to an object of type *TEscenario* as attribute, containing the needed logic for the simulation. At every moment, information contained in *TEscenario* will be shown to the user in the window who will be able to modify its values by using the widgets provided. At a time, *TEscenario* contains attributes from different kinds. The most important among them is the one which stores the topology of the scene whished to be simulated. This information is contained in a *TTopologia* object. Apart from the topology, the scene includes needed information about the simulation configuration such as its length, sample interval, etc. This data is stored in an attribute of type *TSimulacion*. With these two objects we have the basis to start simulating because we know the GoS/MPLS network configuration and the timing parameters to be applied to the network in order to generate the simulation results.

TSimulacion

In addition to the simulation configuration, the *TSimulacion* object has a reference to an object of type *TREcolectorSimulacion*. This object will receive the data generated by the scene elements (nodes and links) throughout the simulation. It acts as OpenSimMPLS engine exit point and functions as nexus with the simulation events interpreter.

TTopologia

Instances of this class contain the whole network specification. They have a graph structure that allows a formal specification of the network connections to be simulated. This structure consists of a binary tree that stores elements of type *TNodoTopologia* (network nodes) and another binary tree that stores elements of type *TEnlaceTopologia* (links between nodes). Each of the elements, nodes and links, has its own features that will be explained afterwards. *TTopologia* counts on another attribute of type *TReloj* which is essential for the simulator to work properly. When simulation starts, *TReloj* sends periodically signals to every network nodes and links, giving them a span of time (defined in *TSimulacion*) in which they must make their necessary tasks. During this span, they must notify everything they do, to the topology *TRecolectorSimulacion* object. On finishing each span of time, all the elements are halted until the timer gives them other period of time to be able to continue. In this way, after successive spans, OpenSimMPLS records partial shots of the state of all the network elements that, once presented as an events sequence, allows observing the behaviour of the simulated network.

TElementoTopologia

This is an abstract class. All the elements likely to be inserted in a network with OpenSimMPLS are inherited from it (Figure 14).

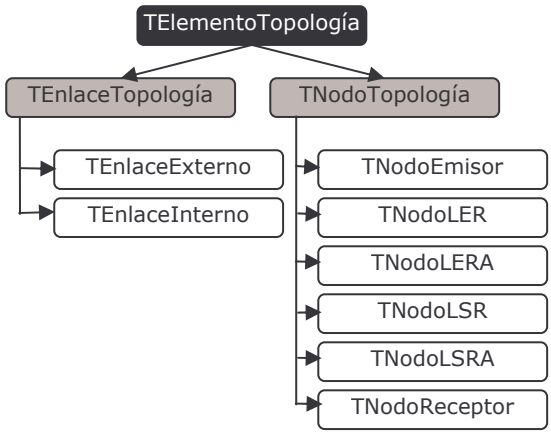

Figure 14. Elements Hierarchy

TEnlaceTopologia represents a link from the simulated network. Among its attributes we find the link delay and two references to the nodes it joins. There are two link varieties depending on if it joins two internal GoS/MPLS domain nodes or if one of these nodes is an external one. This differentiation is made exclusively with a visual representation purpose. When a link receive a span of time from the simulator timer, it starts to run the task that is configured for, which consists on propagating the traffic which is crossing it, slowly or fast depending on the link delay, from the source node to the sink one.

TNodoTopologia is the super class for all kinds of nodes that can be placed in an OpenSimMPLS scene. There are six varieties of this type:

TNodoemisor, is a class that implements all the traffic generator node functions. It is designed to generate fixed size packets or variable size packets. In the latter case, traffic is generated by using the real average distribution announced in the Internet2 Netflow Weekly Report (Table 1).

Table 1. Distribution of packets sizes

Traffic %	Size range (octets)
47%	0 – 100
24%	101 – 1400
28%	1401 – 1500
1%	1501 – 65535

TNodoLER, is a class that implements all the LER node functions, in charge of pushing or popping MPLS labels to the incoming or the outgoing traffic in a MPLS domain. This type of nodes incorporates the capability of carrying out end to end LSP establishment.

TNodoLERA, is a class that implements all the LERA (LER-Active) node. This type of node has the same capabilities that a LER node but, besides, it supports working with information flows that require GoS. In this class, multitude of features not present in basic LER nodes, are implemented, such as traffic prioritizing with PRR (Prioritized Round Robin), packets recovery with EPCD + GPSRP (Early Packet Catch and Discard + GoS PDU Store and Retransmit Protocol), fast rerouting in case of a main LSP failure with RLPRP (Resilient Local Path Recovery Protocol), load balancing with RABAN (Routing Algorithm for Balanced Active Networks), and so on.

TNodoLSR, instances of this class allows simulating a LSR node that switch the traffic fast along the MPLS domain.

TNodoLSRA, instances of this class implements all the basic LSR node features. It represents a LSRA (LSR-Active) node; moreover, objects of this type have the same advanced features that LERA nodes, that is to say, PRR, EPCC GPSRP, RLPRP and RABAN features are implemented on them.

TNodoReceptor. This class implements a sink node. Te structure of this node is simple since it is solely thought for the traffic to be directed to it. Its main features are that it is never congested and it requires only little effort to configure it.

Any element of the topology, nodes or links, follows the operation method showed below (Figure 15).

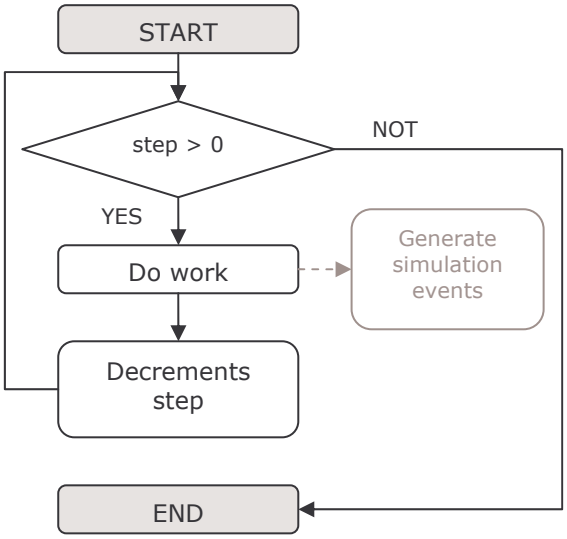

Figure 15. Flowchart of Elements Operation

The simulator is thread programming. This enables each topology element to work independently and concurrently during the simulation taking advantage of the processor capacity and the resources of the host where the application is running.

Interpreter of Simulation Events

The last component of OpenSimMPLS is the simulation events interpreter. Nowadays the simulator relies on a unique interpreter whose mission is to show graphical and visually the simulation result. Every *TEscenario* element has a reference to an object of type *JPanelSimulacion*. The latter is the graphic element which receives simulations events from the events collector (of type *TRecolectorSimulacion*) coming from the elements themselves during their operation. *JPanelSimulacion* allows showing different types of events: on-fly packets, discarded packets, broken links, established LSP, congested nodes, packets switching, LSP signalling, backup LSP establishment, etc.

OpenSimMPLS has been designed so that in the future new simulation events interpreters can be developed in order to show the simulation in a very different ways with minimum changes in the application, e.g. inclusion of a sound in each event for people with visual difficulty, separation of the simulation generation from its visual display over the network, and so on.

Features Summary

The OpenSimMPLS characteristics are plentiful and can help the researchers and network designers who wish to make network prototypes; in order to make easy their work, all the technologies and protocols supported by the simulator are briefly described next:

- TCP (Transmission Control Protocol) as IPv4 packets payload. Although the simulator works at layers 3, 2+ and 2 of OSI (Open Systems Interconnection) model, the existence of TCP traffic is simulated so that overhead introduced by this type of traffic, is reflected in the statistics. Marking the traffic at layer 4 is allowed in order to support the proposals of GoS over MPLS.
- IPv4 over MPLS. The Simulator allows the IPv4 traffic to be labelled with a MPLS header.
- IPv4 flows. Source nodes can generate IPv4 traffic with real properties of real IPv4 traffic.
- MPLS flows. Source nodes can generate MPLS traffic with real properties of real MPLS traffic.
- GoS marked IPv4 flows. This feature allows that the simulator supports the traffic prioritizing and the proposals to support GoS over MPLS. Traffic is allowed to be marked at layer 3.
- GoS marked MPLS flows. Permite que el simulador soporte la priorización de tráfico y las propuestas para soportar GoS sobre MPLS. Se permite marcar el tráfico a nivel 2+.
- Constant traffic. The traffic sources can generate packets with a constant cadence and a predefined fixed size.
- Variable traffic, with a packet size distribution from real Internet. Following the distribution average of the weekly traffic statistics of the Abilene network (Internet2), the simulator allows to generate similar variable traffic to the real traffic of Internet.
- Downstream on demand label distribution. The simulator allows the request of labels for the LSP establishment, but not the advertisement of predefined labels.
- TLDP (Tiny Label Distribution Protocol) support. The IETF (Internet Engineering Task Force) has recommended leaving the research on LDP (Label Distribution Protocol) so that, while the simulator is equipped of RSVP (Resource Reservation Protocol) label distribution, it incorporates a reduced LDP subgroup, at functional level, to be able to requests and allocates MPLS labels. This protocol has been called TLDP.
- GPSRP support. This protocol comprises of the proposal of supporting GoS over MPLS. It allows making local retransmissions of privileged packets when they are discarded in buffers nodes. In the simulator, this feature is only available for active nodes.
- RLPRP support. This protocol comprises of the proposal of supporting GoS over MPLS. It allows to the establishment of a backup LSP and its activation when the main LSP fails. In the simulator, this feature is only available for active nodes.
- RABAN support. This algorithm comprises of the proposal of supporting GoS over MPLS. It is a routing algorithm that allows the load balancing in network congested networks. In the simulator, this feature is only available for active nodes.

- DMGP (Dynamic Memory for GoS PDU) implementation. This component belongs to the proposal of supporting GoS over MPLS. DMGP is a node internal memory that allows the temporary storage of the privileged packages that are switched by the node. This makes possible their later retransmission if necessary. In the simulator, this feature is only available for active nodes.
- Floyd algorithm support. Traditional routing algorithm.
- Prioritized Round Robin buffers management method. This method belongs to the proposal of supporting GoS over MPLS. It is a queue policy that allows prioritizing the traffic according to its type and its GoS requirements. In the simulator, this feature is only available for active nodes.
- EPCD support. This technique comprises of the proposal of supporting GoS over MPLS. It allows the early detection of packet discards. This makes possible to detect that a privileged packet has been discarded and then, request its retransmission. In the simulator, this feature is only available for active nodes.

In addition, in order to be more useful as prototyping tool, the simulator considers visual and analytically the following situations:

- Local packets recovery.
- Local LSP's recovery.
- Complete scene simulation.
- LER, active LER, LSR, active LSR, source and sink simulation.
- Broken links simulation.
- Congestion cases simulation.
- Complete nodes statistics.
- LSP's and backup LSP's establishment simulation.
- Discarded packets simulation.
- Statistics charts printing.
- Statistic chart export to PGN images.
- Links delay simulation.
- Differentiated simulation for each type of traffic.

Finally, in order to assure that the OpenSimMPLS simulator acceptably fulfilled quality requirements, the bug detection software Findbugs, from the University of Maryland (Hovemeyer and Pugh, 2004), have been used during the simulator development process.

CONCLUSIONS AND FUTURE WORK

The present work proposes the use of the OpenSimMPLS (http://gitaca.unex.es/opensimmpls) simulator to the analysis, planning and assessment of GoS/MPLS networks in different areas such as research, teaching or commercial networks solutions deployment. This use is justified on the great interest the MPLS technology has arisen and on the difficulty of deployment real prototypes, either by it expensive economic cost or by the impossibility of adding new features to the nodes switch fabric existent in the present industry.

It has been confirmed in the practice that the simulator has a positive response over different platforms and allows obtaining satisfactory results, adjusted to the real MPLS networks operation, both in a visual form and in analytical one.

To sum up, it has been verified that the use of the simulator offers the following advantages:

- Simplicity of implantation and use. The simulator does not require data base and admits multiple architectures and operating systems. It has been probed over Intel and SPARC architectures, with Linux, Windows and Mac OS X operating systems.
- With OpenSimMPLS the different components of a MPLS domain can be reconfigured so that, the consequences of these changes can be analyzed. On a real MPLS network these modifications will not always be allowed.
- The simulation allows obtaining detailed trace files and statistics with which concrete behaviours of a MPLS domain can be analyzed.
- The use of the simulator will always suppose a more economic validation and prototyping solution than a real MPLS domain deployment.
- Its opensource license allows adding improvements in the application or adding new features, before starting to design new real switch fabric.
- The use of the simulator allows to make demonstrations to clients before deploying a real network solution to them, so that they can observe (and analytically verify) the advantages that they will obtain.

Currently, the simulator is being extended with the incorporation of new features among which the most important are the signalling and the resource reservation using RSVP-TE (Resource Reservation Protocol – Traffic Engineering) and CSPF (Constrained Shortest Path First). And, in a near future, we are working so that OpenSimMPLS allows simulating GoS/MPLS networks with IPv6 (Internet Protocol version 6) traffic and supporting more than a single MPLS domain. This will make possible the creation of interdomain environment prototypes supporting QoS and/or GoS to build e.g. MPLS virtual private networks (VPN-MPLS).

REFERENCES

Ahn G.; Chun W. "Design and Implementation of MPLS Network Simulator". *15th International Conference on Information Networking*, February 2001.

Ahn G.; Chun W. "Simulator for MPLS Path Restoration and Performance Evaluation". *Joint 4th IEEE International Conference on ATM (ICATM 2001) and High Speed Intelligent Internet Symposium*, April 2001.

Cavendish, Dirceu; Ohta, Hiroshi; Rakotoranto, Hari. "Operation, Administration, and Maintenance in MPLS Networks". *IEEE Communications Magazine*, Oct. 2004.

Domínguez-Dorado M.; Rodríguez-Pérez F. J.; González-Sánchez J. L.; Marzo J. L.; Gazo A. "An Architecture to provide Guarantee of Service (GoS) to MPLS". *IV Workshop in G/MPLS Networks*, April 2005.

Goh, Walter. "Review: JFreeChart". *NewsForge – The Online Newspaper for Linux and Opensource,* January 12th 2006. Available at http://software.newsforge.com

Hovemeyer, David; Pugh, William. "Finding Bugs is Easy". *SIGPLAN Notices*. December, 2004.

Huang, Changcheng; Sharma, Vishal; Owens, Ken; Makam, Srinivas. "Building Reliable MPLS Networks Using a Path Protection Mechanism". *IEEE Communications Magazine*, March 2002.

Internet2 Netflow. http://netflow.internet2.edu/weekly.

Kodialam M.; Lakshman T. V. "Restorable Dynamic QoS Routing". *IEEE Communications Magazine*, June 2002.

Marzo J.L.; Calle E.; Scoglio C; Anjali T. "QoS Online Routing and MPLS Multilevel Protection: A Survey". *IEEE Communications Magazine*, October 2003.

OpenSimMPLS: Multiprotocol Label Switching simulator. http://gitaca.unex.es/opensimmpls

Rosen E. et al. "Multiprotocol Label Switching Architecture", *IETF RFC 3031*, January 2001.

Rosen E. et al., "MPLS Label Stack Encoding", *IETF RFC 3032,* Jan. 2001.

González-Valenzuela, Sergio; Leung, Victor C. M. "QoS Routing for MPLS Networks Employing Mobile Agents". *IEEE Networks*, May – June, 2002.

AUTHOR BIOGRAPHIES

MANUEL DOMÍNGUEZ-DORADO was born in Zafra, Extremadura, Spain and went to the Polytechnical School of Cáceres, University of Extremadura where he studied Computer Science Engineering and obtained his Ms. D. in 2004. He worked for SADIEL, S.A. before moving again to the University of Extremadura where he is now a Ph. D. candidate. He researchs into multiprotocol technologies and interdomain routing while belongs to the Applied Telematic Engineering and Advanced Communications Research Group. His email address is `mdomdor@unex.es` and his web page can be found at `http://gitaca.unex.es/manolodd`.

FCO. JAVIER RODRIGUEZ-PEREZ was born in Huelva, Spain and went to the Polytechnical School of Cáceres, University of Extremadura where he studied Computer Science Engineering, obtaining his degree in 2000. He worked for RGD Solutions S.L. for three years and now he is a Ph. D. candidate, researching into QoS and traffic engineering techniques over MPLS. His email address is `fjrodri@unex.es`.

JOSÉ LUIS GONZÁLEZ-SÁNCHEZ received the Engineering degree in Computer Science at the Computer Science Faculty of Barcelona (Polytechnic University of Cataluña) followed by a Ph. D. in Computer Science (Polytechnic University of Cataluña). Since 1995, he has been with the Department of Computer Science (University of Extremadura) as assistant professor. He is the main researcher of the Applied Telematic Engineering and Advanced Communications Research Group (GITACA). Their areas of interest are: Quality of Service, Communications Protocols, Traffic Engeeniering, MPLS and Security. His e-mail address is `jlgs@unex.es` and his web page can be found at `http://gitaca.unex.es/jlgs`.

ALFONSO GAZO-CERVERO received his Ms. D. in computer science from the University of Extremadura (UEx), Spain, in 1999. Since then, he has been a member of the research and teaching staff in the Applied Telematic Engineering and Advanced Communications Research Group (GITACA) of UEx, where he develops his research toward a Ph. D. in QoS provision over heterogeneous networks. His email address is `agazo@unex.es` and his web page can be found at `http://gitaca.unex.es/agazo`.

ACKNOWLEDGEMENTS

This work is sponsored, in part, by the Regional Government of Extremadura (Education, Science and Technology Council) under Grant No. 2PR03A090. AGILA project. http://gitaca.unex.es/agila.