SOPORTE DE GARANTÍA DE SERVICIO (GOS) SOBRE MPLS MEDIANTE TÉCNICAS ACTIVAS

Ingeniería Informática
Escuela Politécnica de Cáceres
Universidad de Extremadura
ESPAÑA

ISBN10 84-15927-19-3
ISBN13 978-84-15927-19-8

M. Domínguez Dorado
ingeniero@ManoloDominguez.com
http://www.ManoloDominguez.com

J. L. González Sánchez
jlgs@unex.es
http://patanegra.unex.es/jlgs

Título

Soporte de Garantía de Servicio (GoS) sobre MPLS mediante Técnicas Activas

Autores

Manuel Domínguez-Dorado
José Luis González-Sánchez

Edita

Centro Nacional de Referencia de aplicación de las TIC basadas en fuentes abiertas (CENATIC)

ISBN10 84-15927-19-3
ISBN13 978-84-15927-19-8

Primera edición
Septiembre de 2013

Si desea ponerse en contacto con el autor principal de la obra,
puede hacerlo a las siguientes direcciones electrónicas

ingeniero@manolodominguez.com
http://www.manolodominguez.com

Resumen

La tecnología MPLS (Multiprotocol Label Switching) aporta potentes mecanismos para integrar tecnologías de redes como ATM e IP con calidad de servicio (QoS). Aunque esta tecnología comienza ya a estar madura, quedan por resolver algunos aspectos como el ofrecer servicios garantizados a fuentes privilegiadas que pueden requerir GoS. Este proyecto investiga y aporta diversas técnicas activas que aportan esa Garantía de Servicio. Para ello sigue dos líneas principales:

- Estudio sobre la implementación de un mecanismo de recuperación local de paquetes con requerimientos de Garantía de Servicio. Este mecanismo permite recuperar información perdida, en un entorno punto a punto en lugar de extremo a extremo, evitando que los protocolos de nivel superior sean los que tomen la iniciativa en una posible retransmisión, para casos puntuales donde la congestión de los nodos provoque el descarte de paquetes privilegiados.
- Estudio sobre la implementación de un mecanismo de recuperación local de LSP. Este mecanismo permite que ante la caída accidental de un enlace de la topología que afecte a un LSP por el que circula un flujo privilegiado, pueda establecerse de una forma adecuada un camino alternativo cuyas propiedades sean similares y así el tráfico privilegiado pueda ser reconducido.

A partir de estos planteamientos, y con la premisa de que el resultado debe respetar el estándar MPLS definido por el IETF, se han diseñado todo un conjunto de técnicas y protocolos que ayuden a conseguir estos objetivos. Entre ellos cabe destacar:

- **RABAN**: Un algoritmo de encaminamiento para redes activas balanceadas. Su propósito es el de encaminar de forma distinta aquellos flujos de información que estén marcados para obtener Garantía de Servicio. Los nodos de la red usarán este encaminamiento cuando sea necesario y cuando los paquetes no requieran GoS, serán encaminados mediante el algoritmo de Floyd estándar, donde los pesos de los enlaces vienen dados por el retardo de los mismos. Con el algoritmo RABAN el encaminamiento busca un compromiso entre el retardo de los enlaces, la saturación de alguna zona de la red y el número de nodos activos a atravesar, de tal forma que

los flujos privilegiados fluyan por zonas con poco retardo, con poco tráfico y atravesando preferiblemente nodos activos.

- **GPSRP**: Protocolo de almacenamiento y retransmisión de paquetes privilegiados. Es un mecanismo por el cual los paquetes que estén marcados con algún nivel de Garantía de Servicio son almacenados temporalmente en una memoria temporal DMGP para su posible retransmisión local en caso de pérdida. Este protocolo sirve para solicitar la retransmisión de paquetes privilegiados, que están identificados inequívocamente en todo el dominio MPLS a aquellos nodos que podrían tenerlos almacenados. También permite la confirmación o denegación de la retransmisión.
- **RLPRP**: Protocolo de recuperación flexible de caminos locales. Es un mecanismo que obliga a preestablecer LSP de seguridad para aquellos flujos que lo requieran para que, en el caso de caída el LSP principal, el continuar transmitiendo sea instantáneo. Este mismo mecanismo es el que activa el camino de reserva o no en función de cómo se encuentre el enlace principal.
- **DMGP**: Memoria dinámica para PDU con GoS. Es la memoria donde se almacenan los paquetes privilegiados que así lo requieran. Está indexada por la clave primaria que todo paquete privilegiado tiene, formada por la IP del origen más un identificador único que éste coloca en cada paquete privilegiado.
- **TLDP**: Subconjunto reducido del protocolo LDP a nivel funcional. Es orientado a la conexión y confirmado. Se establece entre dos nodos del dominio y permite el establecimiento de LSP, la eliminación del mismo, el establecimiento de LSP de respaldo y el desarme del mismo.
- **EPCD**: Algoritmo captura y desechado anticipado de paquetes. Maneja el buffer de entrada de los puertos en los nodos activos. Mientras que en los no activos se sigue una técnica Round Robin, aquí se trata de un Round Robin con prioridad; en condiciones de igualdad es un Round Robin y cuando hay paquetes privilegiados se les otorga preferencia. Además, indica a GPSRP que un paquete con garantía de servicio ha caído y por tanto debe desencadenarse una petición de retransmisión.

Palabras clave

MPLS, RABAN, GPSRP, RLPRP, DMGP, TLDP, EPCD, GoS

Abstract

MPLS (Multiprotocol Label Switching) technology provides powerful mechanisms to integrate network technologies like ATM and IP with Quality of Service. Although this technology is becoming mature, there are some aspects to be solved, such as offering guaranteed services to privileged sources that can require GoS. The project researches and provides techniques that allow support of this Guarantee of Service (GoS). To do that, it follows two main lines:

- Research about the implementation of a mechanism to recover packets with Guarantee of Service locally. This mechanism allows recovering lost information, on a point to point environment instead of extreme to extreme one, avoiding higher level protocols to take the initiative in a possible retransmission in the exceptional cases where nodes congestion causes privileged packets to be discarded.
- Research about the implementation of a mechanism to recover broken LSP locally. This mechanism allows that, when a link of the topology become accidentally down and affects a LSP supporting a privileged flow, an alternative path with similar features can be established in an adequate form and so, privileged flow can be correctly redirected.

Taking into account these approaches, and under the premise that the result must respect the MPLS standard defined by IETF, A set of techniques and protocols has been designed in order to help to the achievement of these objectives. Among them must be highlighted:

- **RABAN**: Routing Algorithm for Balanced Active Networks. Its purpose is that of routing Guarantee of Service marked flows in a different way. Network nodes will use this routing algorithm when necessary. If packets do not require GoS, they will be routed by means of the standard Floyd algorithm, where links weight depends mainly on their delay. With RABAN algorithm, routing seeks to an agreement between links delay, congestion in some network zones and the number of active networks to be gone through, in such a way that privileged flows go across network areas with low delay, low traffic, and passing preferably through active nodes.

- **GPSRP**: GoS PDU Store and Retransmit Protocol. It is a mechanism by which the packets having some Guarantee of Service level are stored temporarily in a memory called DMGP to be locally retransmitted in case of lost. This protocol is used for ask for the retransmission of privileged packets, which are unmistakably identified in the whole MPLS domain, to active nodes than could keep them stored. It also allows the confirmation or deny of the retransmission.
- **RLPRP**: Resilient Local Path Recovery Protocol. It is a mechanism that obliges to pre-establish a backup LSP for those flows that require it, so, in case of principal LSP becomes down, the transmission can continue instantaneously. The same mechanism actives the backup link, or not, according to the principal link state.
- **DMGP**: Dynamic Memory for GoS PDU. It is the memory where privileged packets that require it, are stored. It is indexed by the primary key that every privileged packet posses, and that is composed by the source IP and a unique identifier which is assigned by the source node to every privileged packet that it sends.
- **TLDP**: Tiny Label Distribution Protocol. It is a functional level reduced subset of the LDP protocol. It is connexion-oriented and provides acknowledged services. It works between two domain nodes and it allows the LSP establishment, its removal, backup LSP establishment and its elimination.
- **EPCD**: Early Packets Catch and Discard. It is an algorithm that manages the buffer of the incoming ports in the active nodes. While in non-active nodes the technique used is Round Robin, EPCD is a prioritized Round Robin; in equality condition, it functions as traditional Round Robin, whereas when a privileged packet arrives, it is given priority. Moreover, it indicates to GPSRP that a GoS packet has been discarded and so a retransmission request must be done.

Key words

MPLS, RABAN, GPSRP, RLPRP, DMGP, TLDP, EPCD, GoS

Este Proyecto Final de Carrera está cofinanciado en parte por el **Excmo. Ayuntamiento de Zafra** y por la **Universidad de Extremadura** con fondos provenientes de la "*Convocatoria 2003 de Ayudas para la realización de Memoria de Licenciatura o Proyecto Fin de Carrera y de mejores expedientes*".

Excmo. Ayuntamiento de Zafra
http://www.ayto-zafra.com

Universidad de Extremadura
http://www.unex.es

Índice simplificado

Índice completo

1. Presentación del proyecto

1.1. Introducción

Actualmente las *redes de comunicaciones de datos* están tomando un auge muy importante; el aumento del número de usuarios y el aumento de las aplicaciones para las que el intercambio de datos es necesario, así como la migración de la telefonía tradicional hacia la telefonía IP, el vídeo bajo demanda, etcétera, están repercutiendo drásticamente en los proveedores de infraestructuras tecnológicas de red que se ven avocados a realizar una dura transformación para poder *dar respuesta a las exigencias de la sociedad moderna*.

Simultáneamente, la aparición de las *tecnologías de conmutación óptica*, capaces de gestionar grandes volúmenes de información, requieren el diseño de *nuevos métodos de señalización y nuevos protocolos de comunicaciones* que permitan aprovechar toda su capacidad y que transformen la *red* en una *entidad inteligente*; una red flexible que permita no sólo el mero transporte pasivo de información sino el control, la gestión de recursos y fiabilidad sobre la información y sobre la propia infraestructura de red. Una red parametrizable que además cumpla el objetivo de reducir los costes de los proveedores de servicios de red y unifique el maregmánum de tecnologías actualmente implantadas, cuyo mantenimiento es un problema tanto económico como por la imposibilidad de ofrecer servicios de banda ancha con una calidad de servicio aceptable.

En este escenario surgen o reaparecen nuevas tecnologías:

Dense Wavelength Division Multiplexing (DWDM) como tecnología innovadora de conmutación para medios ópticos basada, como su nombre indica, en la multiplexación por división de la longitud de onda (LUZ). Utiliza las leyes de las ondas luminosas (reflexión, difracción, refracción, etcétera) para convertir los haces luminosos en portadoras de información muy rápidas.

Generalized Multiprotocol Lambda Switching (GMPLS), como evolución natural de la conmutación de etiquetas multiprotocolo (MPLS) hacia el etiquetado implícito en medios

ópticos basados en la longitud de onda. Esta tecnología permite entre otras muchas cosas interpretar los distintos haces luminosos sobre medios ópticos como etiquetas o marcas en base a las cuales encaminar la información desde el origen hasta el destino.

ActiveNets o redes activas. Esta arquitectura de red lleva tiempo en estudio pero ahora se presenta como una interesante opción para añadir a una potente red formada por el binomio GMPLS/DWDM la inteligencia y flexibilidad que comentábamos. Con esta arquitectura a cada paquete de datos acompaña información sobre cómo se ha de tratar dicho paquete. Los elementos de activos de la red gestionarán en base a esta información los recursos, el ancho de banda, el funcionamiento de la propia red, etcétera, en tiempo real.

Sistemas Multiagente (SMA). Esta arquitectura de red persigue una filosofía parecida a la de las redes activas, aunque desde un punto de vista mucho más software y haciendo uso del concepto de agente inteligente estudiado por la inteligencia artificial y la inteligencia artificial distribuida. Con esta arquitectura, diversos agentes software se propagan por la red colaborando entre si para conseguir adaptar el funcionamiento de la misma y así acoplarse a las necesidades del tráfico que está transportando, en tiempo real.

1.2. Objetivo del proyecto

Utilizando como punto de partida las tecnologías citadas y tras un análisis previo de los requerimientos y los problemas existentes, nos proponemos realizar el estudio y prototipo de una infraestructura de red basada en MPLS que mediante el uso de técnicas activas acerque las prestaciones de MPLS a las propuestas que se persiguen con GMPLS sobre medios ópticos conmutados vía DWDM. El prototipo dará solución a algunos de los problemas que aparecen al implantar un sistema con estas características en cuanto a la gestión del tráfico y los recursos, calidad de servicio, disponibilidad de la red, recuperación ante errores, etcétera.

El fin último de este proyecto es sentar una base que permita diseñar e implantar redes de área metropolitana (MAN) o de área extensa (WAN), en definitiva, redes de cierta envergadura, redes troncales, proporcionando garantía de servicio (GoS) a flujos privilegiados de información y permitiendo la recuperación de tramas descartadas y de

caminos en un entorno local, evitando así en la medida de los posible las retransmisiones extremo a extremo solicitadas por el nivel de transporte.

1.3. Metodología utilizada

El desarrollo del proyecto se regirá por las metodologías propias de la ingeniería informática, por tanto se dividirá a muy grandes rasgos en los siguientes apartados:

Estudio del problema existente: donde profundizaremos en la situación actual de las redes de comunicaciones de datos y en las tecnologías que están surgiendo, con el objetivo de fijar los problemas a los que hemos de dar solución.

Estudio de viabilidad: en este apartado analizaremos si hay una solución factible para los problemas, a tenor de las tecnologías existentes, del coste necesario para esas soluciones o de cualquier otro factor. Pondremos las cotas al el desarrollo del proyecto.

Diseño de la solución: donde profundizaremos detalladamente en los problemas encontrados, modelando una solución óptima para ellos. Al terminar esta fase la solución se encontrará totalmente planificada y diseñada acorde a los requerimientos.

Desarrollo: crearemos el prototipo final diseñado en la fase anterior de tal forma que sea portable, robusto, seguro y útil. Plasmará en la realidad lo que hayamos trabajado durante las fases anteriores.

Documentación: para que el proyecto cumpla el objetivo de ser la base para la implantación de las tecnologías de las que hemos hablado, debe estar bien documentado. Aquí nos dedicaremos en detalle a dejar documentadas todas las fases, el desarrollo del proyecto y los manuales necesarios para que pueda ser reutilizado en el futuro como base para más proyectos, para ampliación del mismo o para su uso.

1.4. Aplicaciones de la solución

Hay que destacar que la tecnología que vamos a tratar en este proyecto, MPLS es actualmente una de las *áreas más innovadoras* de investigación (en el mundo de las redes de comunicaciones) en el que todos los organismos están trabajando y en el que están invirtiendo la totalidad de empresas de comunicaciones. Combinada con las mejoras que proponemos, al añadir técnicas activas al funcionamiento de la red, permitirá ofrecer *servicios avanzados de comunicación* de calidad y con garantía de servicio.

Concretamente el mejor ámbito de aplicación para este proyecto es el de las redes metropolitanas y de área extensa. Introduciendo esta tecnología se conseguirá *potenciar el comercio electrónico* y el uso las nuevas tecnologías en las mejores condiciones existentes, con fluidez y seguridad y permitirá la colaboración telemática eficaz entre empresas, expositores, *vendedores y compradores*.

Las redes metropolitanas se podrán expandir para dar *acceso a los ciudadanos a los centros neurálgicos* como hospitales, centros de salud, ayuntamientos, colegios, institutos, etcétera, desde sus casas. *Y posteriormente* se podrán ofrecer estos servicios, mediante redes de área extensa.

2. Situación tecnológica actual

2.1. Multiprotocol Label Switching (MPLS)

Actualmente y desde hace varios años estamos sufriendo una expansión exponencial de las redes de comunicaciones a nivel mundial. Posiblemente las redes de telecomunicaciones sean en los países modernos, junto con red de carreteras y de ferrocarriles, una de las infraestructuras más importantes y un indicador del estado de desarrollo y del producto interior de los países.

Internet, el principal eje sin duda en cuanto a redes de comunicaciones, también se ve afectada por este crecimiento exponencial debido a lo cual, están surgiendo una serie de problemas que están invalidando los principios técnicos que hasta ahora se habían mostrado eficaces para dar soporte a esta infraestructura: los protocolos de comunicaciones y la tecnología hardware. Estos problemas surgen cuando se tiene que repartir los recursos entre los distintos usuarios pero se ven a todas luces insuficientes como para que se pueda asegurar una mínima calidad; entonces ¿Cómo se tiene que hacer este reparto?

Los ingenieros de todo el mundo llevan años trabajando para solucionar este problema, principalmente desde dos puntos de vista distintos, permitir que cualquier usuario pueda hacer uso de las redes aunque esto implique un pésimo rendimiento, o limitar el número de usuarios simultáneos de tal forma que aquellos que en un momento dado acceden a la red lo hagan con una mínima calidad. En torno a estos puntos de vista han surgido muchas alternativas pero no todas han tenido la misma aceptación: hay trámites burocráticos, problemas de implantación, incompatibilidades, problemas políticos… Parafraseando a Andrews S. Tanembaum: *"en las nuevas tecnologías los cambios revolucionarios, por buenos que sean, fracasan; se deben adoptar cambios evolucionarios"*.

Por tanto, el actual problema y punto en el que se encuentran la mayoría de investigaciones, consiste en hallar la fórmula menos drástica para realizar la adopción de las modernas propuestas técnicas ideadas por los ingenieros: ATM, IPv6, Gigabit Ethernet, etcétera, admitiendo a su vez la compatibilidad con las técnicas de siempre. Y es de esta

situación de donde surgen muchas de las ideas que giran en torno a la banda ancha y entre las cuales se encuentran conceptos como la calidad de servicio, la integración de servicios, UMTS, XDSL y, lo que nos ocupará más espacio en este trabajo, MPLS.

Todas las propuestas surgidas están teniendo su lugar de aplicación, sus características, etcétera; y para cada problema existe una solución mejor que otra. El ámbito de actuación de MPLS, en el que nos centraremos, será las troncales de redes, generalmente WAN, pero eso lo veremos más adelante.

2.1.1. Banda ancha y calidad de servicio

Existen diferencias a la hora de considerar qué es banda ancha y qué no. Ciertos sectores defienden que se puede considerar como de banda ancha aquellas redes que ofrecen una gran productividad, por encima de un nivel concreto, mediante el incremento de la tasa de transferencia; vulgarmente llamaremos al concepto de incrementar la tasa de transferencia como incrementar el ancho de banda. En este sentido, por tanto, incrementar el ancho de banda desembocaría en una red de banda ancha, un gran cauce por el que mandar gigantescas cantidades de información. Pero… ¿Se podría asegurar un ancho de banda concreto para una clase de tráfico concreta? ¿Alguien se dejaría operar remotamente vía una red de "banda ancha" con este sistema?

En la otra parte se encuentran aquellas personas que piensan que el incrementar el ancho de banda (realmente la tasa de transferencia) es un único paso o parámetro de los que realmente definen una red de banda ancha. En este caso, se introduce el término *calidad de servicio* como propiedad de una red de comunicaciones que permite ajustar varios parámetros en la transmisión relacionados con la calidad de las comunicaciones más que con la cantidad de información transmitida. Los parámetros más importantes de entre ellos son los siguientes:

DELAY: permite variar a la hora de realizar una conexión, el máximo retardo que se considera aceptable para una transmisión.

THROUGHPUT: permite especificar el ancho de banda requerido o, mejor dicho, la tasa de transferencia que se desea tener garantizada.

JITTER: es la variabilidad en el retardo que se desea como máximo. Hay veces que no importa en gran medida que exista retardo sino que se retardo se pueda acotar entre unos márgenes para que sea más o menos constante.

RELIABILITY: permite especificar la tasa de errores máxima que consideramos aceptable para la conexión.

Por tanto, para este segundo sector de opinión, una red de banda ancha es aquella que, además de proporcionar una tasa de transferencia elevada, permite negociar y asegurar todos estos parámetros a la hora de realizar la conexión; mientras que una red que permita un gran ancho de banda pero que soporte tráfico *best effort* no permita ajustar los parámetros de calidad de servicio no se considerará banda ancha.

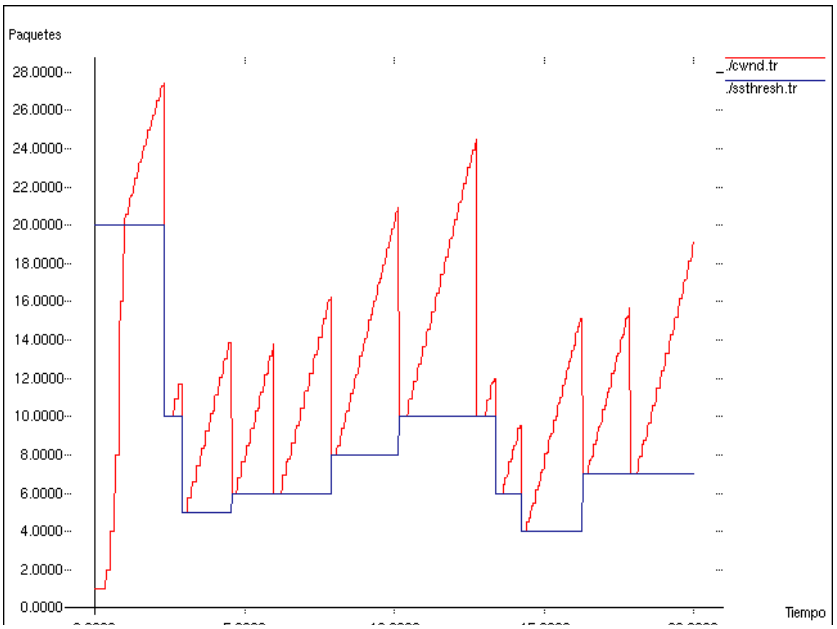
En este estudio adoptaré esta segunda acepción para definir las redes de banda ancha con calidad de servicio que permitan ingeniería de tráfico.

2.1.2. Redes IP: tráfico de datos

Las redes IP han sido tradicionalmente redes para transferencias exclusivamente de datos. Esto implica que la prioridad principal haya sido el asegurar que los datos lleguen; independientemente de cuanto tarden, cómo lo hagan, en qué orden, etcétera. Además, en el momento de creación de los protocolos TCP/IP se suponían unas redes de no mucha calidad y unos equipos terminales no demasiado potentes y la posibilidad de pérdida de infraestructuras por bombardeos, debida a la tensión existente en la guerra fría (no olvidemos que todo esto surgió en el ámbito militar), por lo que se implementó el algoritmo con grandes cabeceras cargadas de información de forma que el trabajo pesado de control se hace en la propia red; esto es bastante bueno para transmisión de datos, pero se pierde gran parte del control sobre el flujo de información, la asignación de recurso, etcétera. Claramente este tipo de tráficos se trata de tráfico *best effort:* lucha por los recursos de la red de forma poco equitativa, aunque existan parches o modificaciones propuestas a TCP para evitar en cierto modo esto; en ningún caso entra en juego el concepto de calidad de servicio. Las redes IP se caracterizan, por tanto, por:

- Enormes paquetes de información.

- Paquetes de tamaño variable.
- Desorden de paquetes en el punto de destino.
- Tamaño de paquetes potencia de dos.
- No orientación a la conexión (al menos en nivel de red).
- No se pueden negociar parámetros para asegurar la calidad de servicio.



Lucha por los recursos en redes con tráfico best effort (TCP/IP)

Todo lo anterior está bien, como digo, si el tráfico a transportar es de datos (y no en todos los casos) pero para nada es una red ideal si lo que queremos transportar es un flujo de información multimedia, como audio, video o aplicaciones de datos que requieran un flujo constante asegurado, ya que este tipo de flujo requieren una red radicalmente distinta, en la que exista un cierto orden, sincronismo… calidad de servicio, para resumir. Con redes de este tipo, es difícil conseguir el objetivo de la banda ancha, si tenemos en cuenta la acepción de banda ancha que hemos adoptado páginas atrás.

2.1.3. Redes ATM: tráfico multimedia

Las redes ATM se comenzaron a planificar partiendo de las carencias de IP para soportar tráfico multimedia por lo que desde un primer momento se planificó, desde el punto de

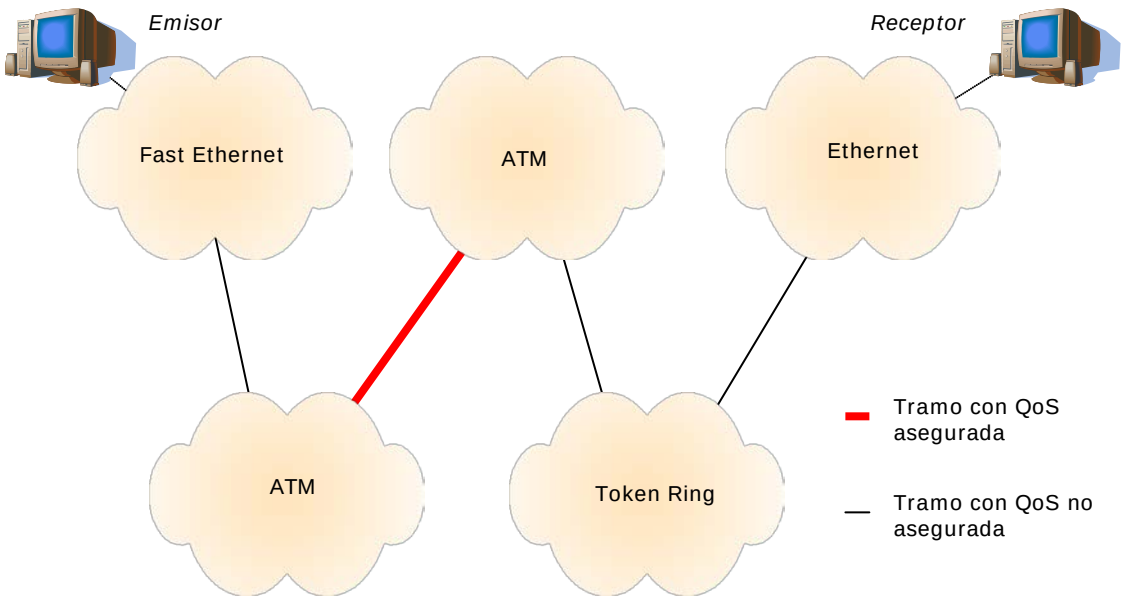
vista de la ingeniería, para tal fin. No importa tanto en este caso asegurar que la información llegue; a fin de cuentas es relevante que en una videoconferencia no llegue una pequeña porción de imagen si con ello la fluidez y rapidez global de la videoconferencia es buena. Tampoco hace falta cabeceras enormes puesto que se suponen redes mejores y, sin duda, equipos terminales muchos más potentes capaces de asumir el trabajo de control. Una red ATM se caracteriza entre otras cosas por:

- Celdas de información de reducidísimo tamaño en comparación con las tramas IP.
- Celdas de un tamaño fijo.
- Los paquetes viajan y llegan al destino de forma ordenada.
- El tamaño de la celda no es potencia de dos.
- Existe orientación a la conexión.
- Se pueden negociar parámetros para asegurar la calidad de servicio.

Evidentemente una red pensada para transportas tráfico multimedia, con capacidades de sincronismo, de gestión del tráfico, de una gran rapidez de conmutación debido al tamaño fijo de las celdas, de gran rapidez en la recomposición de la información en el destino, etcétera. Con redes de este tipo, no resultará difícil conseguir el objetivo de la banda ancha, si tenemos en cuenta la acepción de banda ancha que hemos adoptado páginas atrás; aunque evidentemente no están específicamente pensadas para transportar tráfico de datos y es una salvedad que los partidarios de IP y el tráfico a ráfagas en general, no perdonan.

2.1.4. Problemática de las redes heterogéneas

Actualmente lo que podemos encontrar en la práctica totalidad de Internet y en la mayor parte de las redes WAN, MAN o LAN es, sin duda, sistemas montados sobre esquemas de tráfico *best effort* pero que casi con seguridad están teniendo necesidades de utilización de tráfico multimedia o necesidades de asegurar calidad de servicio. El problema reside en la heterogeneidad de tecnologías de red utilizada que permite asegurar la calidad de servicio sólo en tramos concretos pero rara vez en el ámbito local e incluso en tramos intermedios de la red; Y no parece que se vaya a imponer a medio plazo una solución definitiva y única para este problema. En la siguiente figura se observa mejor este concepto:



Red heterogénea. No se puede asegurar la QoS extremo a extremo

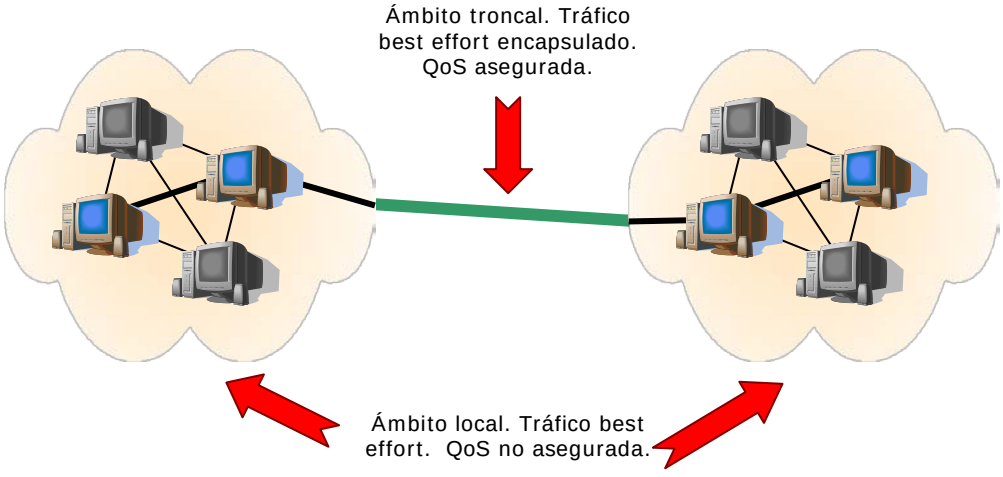
Como podemos observar en la figura, tanto el emisor como el receptor hablan distintos protocolos: Fast Ethernet y Ehernet. Además, la información al salir del ámbito local del emisor, Fast Ethernet, sufre varias transformaciones ya que debe atravesar segmentos de la red ATM y segmentos de la red Token Ring para, al final convertirse en tramas Ethernet que entiende el receptor. Solo existen flujos de información ATM dentro de las propias redes ATM y entre una red ATM y otra, por estar unidas físicamente. Por tanto es sólo en ese ámbito cuando podemos asegurar una calidad de servicio concreta. Pero claro, el problema reside en que la comunicación se realiza extremo a extremo y por tanto no hay forma de poder negociar completamente unas características concretas para la comunicación. Esta es la problemática actual. Si tenemos redes para tráfico de datos, tráficos *best effort* y surgen necesidades de tráfico multimedia, que requiere calidad de servicio hay que proponer soluciones para compatibilizar ambos requerimientos de alguna manera.

Si tomamos la primera acepción de banda ancha, aquella que sólo tenía en cuenta la tasa de transferencia, parece más o menos fácil llegar a una solución: investigar métodos mediante los cuales se pueda exprimir más el ancho de banda de las redes existentes bien instalando nuevas infraestructuras o bien retocando los protocolos, e implantar soluciones basadas en ello. Ya hemos visto que de ningún modo se pueden negociar parámetros de calidad de

servicios con este planteamiento, pero al menos sería una solución temporal; aunque tarde o temprano habría aplicaciones que consumirían el ancho de banda; Además es una solución no escalable; el teorema de Shanon ya impone límites en este aspecto.

Si tomamos la segunda acepción de banda ancha, la que aquí consideramos más completa, como hemos visto arriba, ATM proporciona unas características óptimas para implementarla, pero tiene serias desventajas: las aplicaciones, las redes, los protocolos… toda la infraestructura existente actualmente habla IP, ethernet… tráfico a ráfagas y sin calidad de servicio, como hemos comentado. Implantar ATM desde cero sería una revolución, supondría costes elevados en hardware, software, formación y otros conceptos más. Y en el ámbito local, obligar a usuarios a cambiar de placas de red, conmutadores, etcétera es una propuesta inviable. Así, aunque los proveedores de servicios incorporen tecnología ATM en las dorsales, por ahora no es posible abrir conexiones extremo a extremo con calidad de servicio asegurada, por lo que quedan aún años de tráfico a ráfagas.

Como solución parcial al problema, los proveedores de servicio han llegado a la conclusión de que, si bien no será posible pronto poder negociar conexiones con calidad de servicio extremo a extremo, si se podría intentar garantizar esta calidad de servicio en las troncales o al menos en la mayor extensión posible, abarcando cuanto más segmentos de red, mejor. Es decir, evitar de momento que los usuarios tengan que cambiar su infraestructura, porque no lo van a hacer de hecho e intentar asegurar en el ámbito de los ISP calidad de servicio. En la siguiente figura se observa este concepto:



Y ahora que sabemos someramente cual es el problema y el principio general de cómo se deben plantear las soluciones, ¿qué tecnología es la que permite en la troncal transportar tráfico *best effort* y añadirle cierta calidad de servicio? Hasta ahora se están implantando en este contexto redes ATM que transportan el tráfico a ráfagas en sus celdas; en casi todas las troncales es así, aunque es un campo en bastante movimiento con investigaciones continuas; algunas de ellas apoyando esta tecnología y otras no.

En los últimos años, sin embargo, parece que la industria de soluciones de red está moviendo el mercado hacia el desarrollo de un nuevo protocolo que permita el uso de distintas tecnologías de red a la vez que proporcione un método común que permita asegurar, para todas ellas, cierta calidad de servicio; este esquema de transmisión se llama *Multiprotocol Label Switching* o *conmutación de etiquetas multiprotocolo*, en el que se centra este trabajo y del que hablaremos de aquí en adelante.

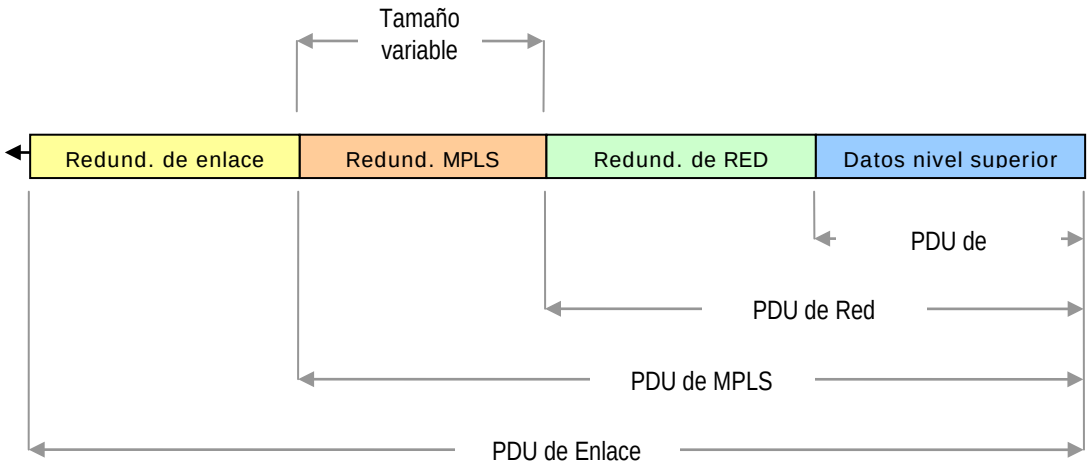
2.1.5. Introducción a MPLS

Para aclarar un poco, de forma somera e incompleta, diremos que Multiprotocol Label Switching (a partir de ahora MPLS) es un esquema similar pero no igual a la encapsulación por la que una PDU de un protocolo cualquiera puede ser transportada en una "PDU" del protocolo MPLS como si datos de un nivel superior se tratara, añadiéndole su redundancia, etcétera. Sin embargo, existen multitud de características que hacen de MPLS una solución atractiva y merecedora de estudio por las universidades y las empresas de *networking*. De no haberlas, parecería innecesario crear un nuevo protocolo, ya que la encapsulación es una técnica de sobra conocida.

MPLS es un esquema de envío de paquetes que no está formado exclusivamente por un protocolo que encapsula a otros sino que define y utiliza conceptos como FEC, LSR, dominio MPLS, LDP, ingeniería de tráfico... Veremos en los siguientes puntos todos estos nuevos conceptos.

MPLS se encuentra situado entre los niveles de enlace y de red del modelo de referencia OSI; por tanto se podría decir que es un protocolo de nivel 2+ que es como se encuentra en gran parte de la bibliografía existente sobre el tema. Esto a efectos prácticos significa hace de nexo de unión entre los protocolos de red (que según lo dicho sería el protocolo encapsulado) y el protocolo de nivel de enlace.

La cabecera de un paquete MPLS se encuentra también entre estos dos niveles, por tanto, entre la cabecera de nivel de enlace y la cabecera de nivel de red; de esta forma, para el protocolo de nivel de enlace un paquete MPLS serán datos empaquetados del nivel superior del modelo. Por tanto, la forma general de una trama entera de comunicaciones tendría el siguiente aspecto:



2.1.6. Glosario de términos específicos

Para poder explicar con propiedad el funcionamiento de MPLS es necesario definir la terminología básica que el IETF propone para ello, o al menos, como es este caso, la terminología más común.

LER (LAYER EDGE ROUTER): router frontera entre capas. Es el encaminador que, según explicábamos, se encuentra en el borde de la zona MPLS y es el encargado de añadir cabeceras MPLS entre las cabeceras de red y de enlace del paquete entrante. Además también es el encargado de retirar esta información cuando un paquete sale de la zona MPLS.

LSR (LABEL SWITCH ROUTER): conmutador de etiquetas. En las explicaciones que se han descrito anteriormente, sería el conmutador del interior de la zona MPLS que interpreta el valor de la cabecera MPLS y la modifica si es necesario, pero en principio no añade ni elimina cabeceras MPLS.

FEC (FORWARD EQUIVALENCE CLASS): clase de envío equivalente. Es el conjunto de paquetes o flujos de información a los cuales, tras entrar en la zona MPLS se le añade una cabecera que hace que sean tratados todos de la misma forma, independientemente de que sean paquetes de distintos tipos de tráfico. A efectos prácticos, es el conjunto de paquetes que ingresan por un mismo LER y a los cuales éste les asigna la misma etiqueta.

LSP (LABEL SWITCHED PATH): camino conmutado de etiquetas. Es el camino que describen el conjunto de encaminadotes y conmutadores que atraviesan los paquetes de un FEC concreto en un único nivel jerárquico en cuanto a la zona MPLS de la que se está hablando. Todos los paquetes del mismo FEC siguen el mismo LSP siempre, de principio a fin (dentro del dominio MPLS).

LABEL O ETIQUETA: información o cabecera que se añade a un paquete cuando ingresa en una zona MPLS. Es la información que añade o quita el LER y que es interpretada por el LSR y gracias a la cual se entiende que diferentes tráficos

forman parte de un mismo FEC. Generalmente la etiqueta define el FEC en el que se incluirá al paquete.

LS (LABEL STACK): pila de etiquetas. Es un conjunto de etiquetas dispuestas en forma de pila. En realidad la redundancia puede ser de tamaño variable como se muestra en la figura de las redundancias apiladas de páginas anteriores. Esto, ya veremos, ocurre porque pueden existir zonas MPLS dentro de otras zonas MPLS y el protocolo lo permite gracias a esta característica. Es la forma en la que proporciona escalabilidad.

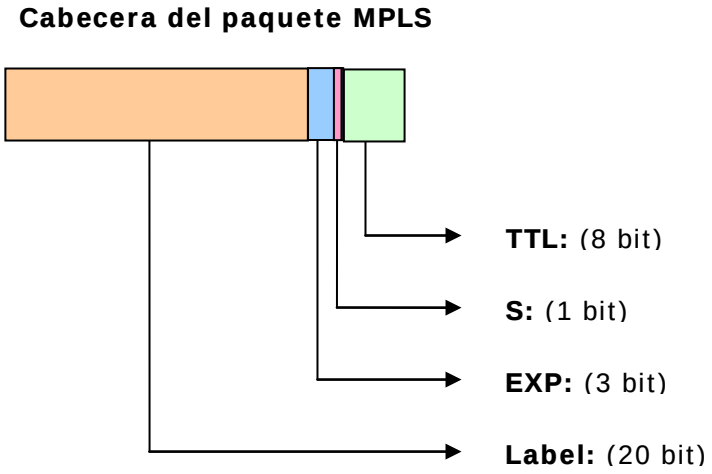
MPLS DOMAIN O DOMINIO MPLS: es el conjunto de encaminadores contiguos capaces de trabajar con MPLS para enrutamiento y/o conmutado y que se encuentran dentro de un mismo ámbito administrativo.

2.1.7. Estructura y características de un paquete

La cabecera de un paquete MPLS tiene un tamaño fijo de 32 bits (4 octetos). Ningún campo es de tamaño variable y además siempre se encuentran localizados en la misma posición. La cabecera MPLS como ya hemos comentado, se situará siempre después de la cabecera de nivel de red y antes de la cabecera de nivel de enlace. Los encaminadores y conmutadores MPLS siempre leerán estos 4 octetos tras la redundancia de enlace.

2.1.7.1. Campos

Cada cabecera MPLS tiene 4 campos: Label, EXP, S y TTL dispuestos como se muestra en la siguiente figura.



TTL (TIME TO LIVE): es el típico campo de casi cualquier paquete de datos que especifica el número máximo de encaminadores que el paquete puede dar antes de ser descartado. Como máximo podrá ser en este caso 65536 saltos. El campo TTL puede tomar el valor del campo TTL del protocolo de red del paquete (si procede) e irse decrementando en cada salto por los encaminadores del dominio MPLS para posteriormente sustituir al valor de TTL del paquete original al salir del dominio MPLS. También se acepta (y aquí lo tomaré así) que el paquete de nivel de red queda encapsulado en un paquete MPLS y el TTL de uno no tiene que ver con el TTL del otro.

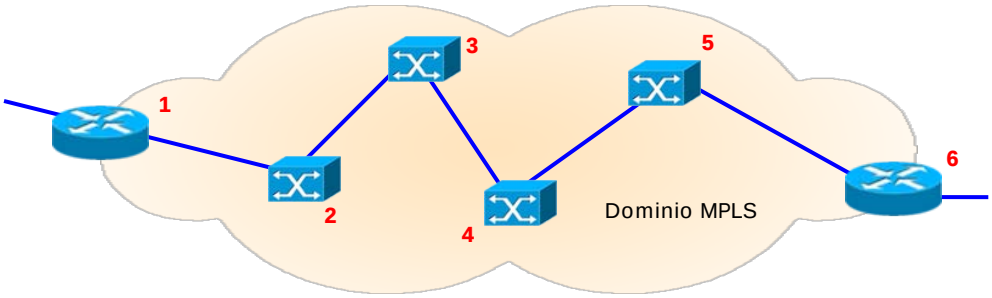
S (STACK BOTTOM): cuando está a 1 indica que esta cabecera MPLS es la última que hay antes de encontrarse con la redundancia de red. Si está a 0, indica que tras esta cabecera MPLS se encuentra otra cabecera MPLS y no la cabecera de red. Ya veremos para que se usa esta característica.

EXP (EXPERIMENTAL): estos bits están reservados para uso experimental. Sin embargo veremos que se han redefinido para albergar información sobre calidad de servicio del paquete.

LABEL: es la etiqueta MPLS, la que da nombre al protocolo. Cuando un paquete ingresa en un dominio MPLS se le asigna (ya lo veremos) una etiqueta que marcará el resto de su viaje a través de la red MPLS.

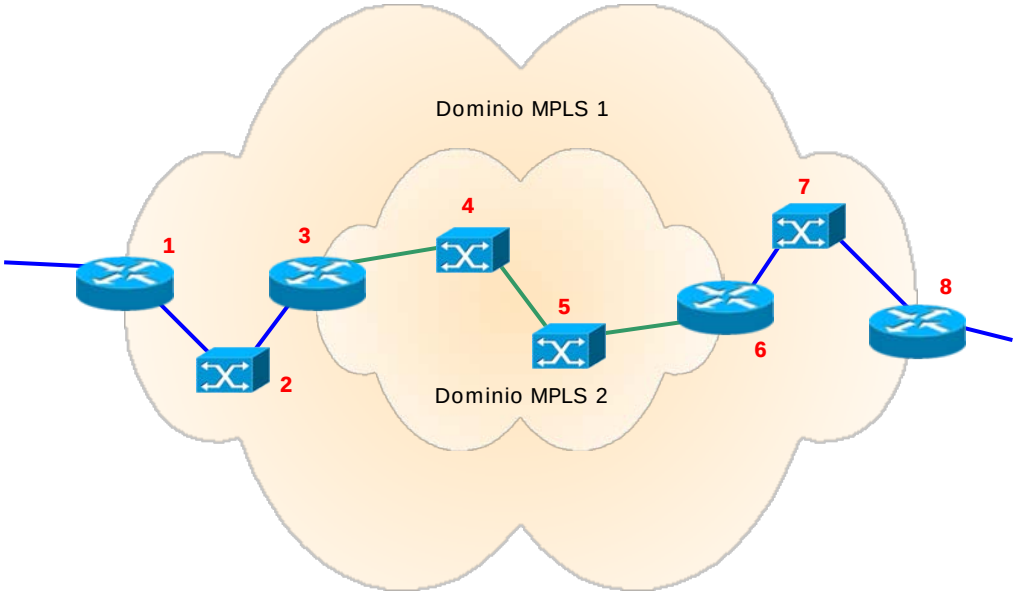
2.1.7.2. Estructura de la pila de etiquetas

Como hemos visto, el campo S nos dice qué nos encontramos tras esta cabecera; o bien la cabecera de red o bien una cabecera MPLS. MPLS "encapsula" cualquier protocolo en un paquete MPLS, aunque este protocolo sea también MPLS. Cuando un paquete entra en un dominio MPLS se le añade una cabecera que tendrá hasta que salga de dicho dominio MPLS. Si durante el trayecto interior el paquete ya etiquetado se introduce en otro dominio MPLS sin haber salido del primero, se le añade una cabecera MPLS más pero en este caso hay que indicar que esa cabecera no es la última existente sino que tras esa que se acaba de insertar, hay otra cabecera MPLS y eso se hace con el campo S. Así cuando un paquete MPLS abandone un dominio MPLS el encaminador que le de la salida sabrá si debe enlutar el paquete con las reglas de MPLS o bien si ya no hay más dominios MPLS y se debe encaminar el paquete con las reglas del protocolo de red que lo haya generado. Veámoslo con un par de figuras:



El caso que hasta ahora conocemos es el de la figura de arriba: Un paquete de una red concreta entra en un dominio MPLS y el encaminador 1 le incrusta una cabecera MPLS entre sus cabeceras de enlace y de red. El paquete se encamina hacia el conmutador 2 el cual, simplificando, decrementa el campo TTL manteniendo la misma cabecera MPLS, sin añadir ni colocar nada, simplemente modificando el campo TTL. A continuación el conmutador 2 envía el paquete MPLS hacia el conmutador 3. En los conmutadores 3, 4 y 5 el proceso se repite; ninguno añade o quita la cabecera MPLS sino que decrementan el valor del TTL y, caso de no llegar a cero (suponemos que no) reenvían el paquete al vecino. Cuando el conmutador 5 envía el paquete al encaminador 6, que es el límite del dominio MPLS, este detecta gracias al campo S de la cabecera MPLS que es la única

cabecera MPLS que hay y que a partir de ahí, por tanto se debe encaminar el paquete con los mecanismos existentes según el tipo de red al que pertenezca (IP, Ethernet…).



El uso de la pila de etiquetas MPLS se ve claramente en esta otra figura. En ella, al encaminador 1 llega un paquete perteneciente a una red de cualquier tipo. Éste le añade una cabecera MPLS con el campo S=1 porque es la última cabecera MPLS que acompaña al paquete, y lo reenvía al conmutador 2 el cual, simplificando para el ejemplo, no hace nada salvo decrementar el TTL y reenviar el paquete al encaminador 3. Este encaminador es el nodo de ingreso a otro dominio MPLS distinto por lo que hace exactamente lo mismo que si el paquete proviniese de una red cualquiera no MPLS, añadir una cabecera MPLS. En este caso será la segunda cabecera MPLS que existirá entre las cabeceras de red y de enlace del paquete: la añadida por el encaminador 1 es la primera y la añadida por el encaminador 3 es la segunda. Esta segunda cabecera deberá llevar por tanto el campo S=0 pues ella no es la última cabecera MPLS que acompaña al paquete. Los conmutadores 4 y 5 reciben el paquete y lo que hacen es, por simplificar, repito, simplemente decrementar el TTL de la cabecera y reenviar el paquete; sin embargo ahora están decrementando el TTL de la segunda cabecera, que es la única que ellos ven. Cuando el paquete llega al encaminador 6, que es fin del "Dominio MPLS 2", éste detecta gracias al campo S que hay más cabeceras MPLS aparte de la que él ve, quita la cabecera (en el orden LIFO, de ahí lo de la pila) que se añadió en el encaminador 3 y envía el paquete al conmutador 7. Hasta aquí el TTL de la primera cabecera ha permanecido como al salir del conmutador 2. El

conmutador 7 recibe el paquete MPLS, le decrementa el TTL y lo reenvía al encaminador 8. Éste al ser fin de un dominio MPLS, el "Dominio MPLS 1" debe eliminar la cabecera MPLS y como detecta por el campo S=1 que es la última cabecera MPLS del paquete la quita y encamina el paquete acorde al tipo de red al que pertenezca éste (IP, Fast Ethernet…).

El proceso de introducirse en múltiples dominio MPLS antes de salir de algunos de ellos se puede repetir tantas veces como sea y por cada dominio MPLS en el que se encuentra el paquete tendrá una cabecera (etiqueta) MPLS.

Hasta ahora, lo dicho, aunque simple, es cierto. Sin embargo hay un problema subyacente en lo que he explicado y es el siguiente: cuando un LER de salida de un dominio MPLS detecta que la cabecera MPLS del paquete es la última, sabe que lo siguiente que encontrará será la cabecera de red y que deberá usarla para encaminar el paquete conforme a los mecanismos de ese tipo de red concreta. Sin embargo MPLS soporta múltiples protocolos de red, realmente cualquiera; y una cabecera IP no tiene la misma estructura que la cabecera Ethernet por lo que aunque el LER sepa que lo siguiente es una cabecera de red, no sabe de qué tipo es esa cabecera y no puede interpretarla; ¿Cómo se soluciona esta situación?.

El mecanismo de cómo se soluciona esto no está excesivamente documentado debido a lo novedoso del protocolo MPLS. Se indica que la última etiqueta MPLS (la primera que se añadió al paquete), la que lleva el campo S=1, no es una etiqueta normal sino que en su campo "Label" lleva un valor de un rango de valores reservados que indican a qué tipo de red pertenece el paquete; por lo tanto cuando un LER de salida detecta en un paquete MPLS el campo S=1, analiza el campo "Label" y este le indicará el tipo cabecera de red que se va a encontrar a continuación y por tanto, será capaz de entender lo que ésta le diga.

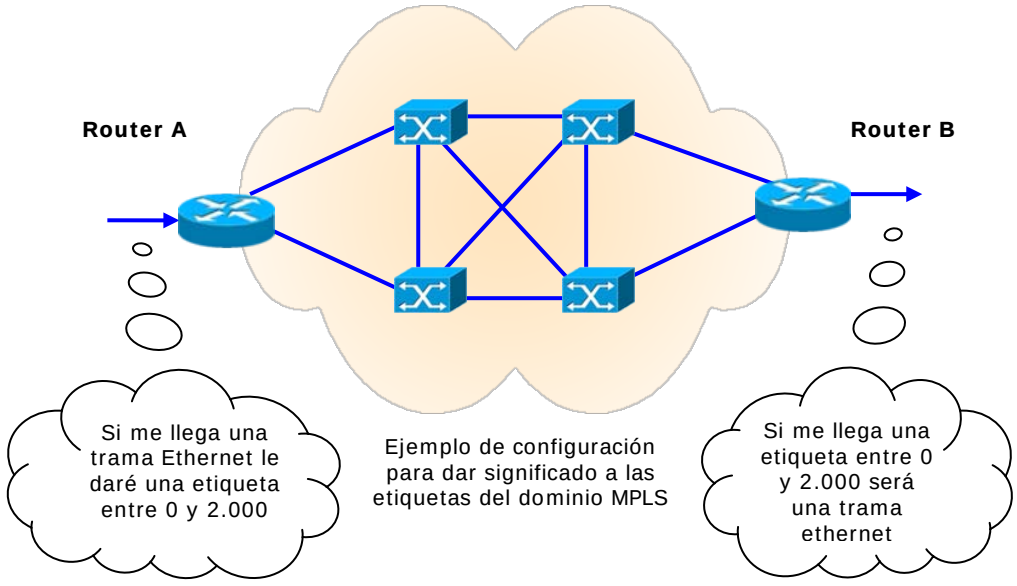
Los valores reservados para el campo "Label" de la primera etiqueta que se añade al paquete (la que tiene el campo S=1) son los siguientes:

- **LABEL = '0':** se usa para indicar que el paquete proviene de una red IP versión 4.
- **LABEL = '2':** se usa para indicar que el paquete proviene de una red IP versión 6.

– **DESDE LABEL = '4' HASTA LABEL = ''15:** están reservados para su posterior uso por parte del IANA.

Existe un par más etiquetas reservadas pero con otros propósitos, por lo tanto no lo explicaré.

Es prácticamente imprescindible en la práctica que el campo "Label" de la última cabecera MPLS especifique el protocolo de red que se ha de usar para sacar el paquete fuera del dominio MPLS. MPLS no impone restricciones al uso de etiquetas para especificar protocolos de red en la última etiqueta de la pila de etiquetas MPLS; los casos reservados, lo están por ser frecuentes, pero nada impide usar otras etiquetas para especificar paquetes IPv4 o IPv6. Por otro lado hay posibilidad de usar hasta 2^{20} – 16 etiquetas, con lo cual no hay problemas en ese aspecto. MPLS concreta que el significado de las etiquetas queda fuera de la especificación del protocolo y son los LER del dominio MPLS los que deben conocer qué significa para ellos la existencia de una etiqueta u otra en la cabecera MPLS con campo S=1. En última instancia serán las personas que administran y configuran los encaminadores del dominio MPLS quienes deciden esto.

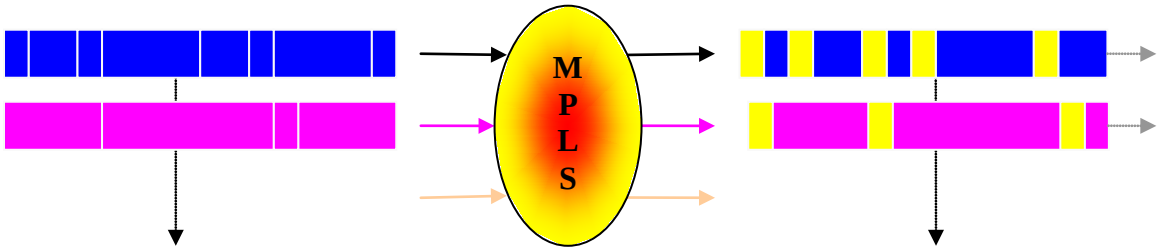


Por ejemplo puedo configurar los LER para que para todos los paquetes Ethernet que ingresen en el dominio MPLS se usen etiquetas desde la 16 hasta la 2.000. Así en el LER de salida del dominio, para paquetes con cabeceras cuyo campo S sea 1 y cuyo campo

Label esté entre 16 y 2.000 sabré que es la última cabecera MPLS y a continuación se encontrará una cabecera de red con estructura Ethernet y que esa cabecera debe ser usada para encaminar el paquete desde el dominio MPLS hacia la red Ethernet. Esto anterior implica que el tráfico ya etiquetado que llegue a un LER de ingreso a un nuevo dominio MPLS ser de confianza ya que si no, se corre el riesgo de hacer reenvío de paquetes de forma ilegítima.

2.1.7.3. Etiquetación frente a encapsulación

Hasta ahora he utilizado el término encapsular entrecomillado porque realmente MPLS no encapsula la información sino que la etiqueta. Cuando se encapsula un protocolo en otro, se construye una PDU_{N-1} del protocolo que encapsula utilizando la información de la PDU_N del protocolo encapsulado, esto es: se adapta el tamaño de la PDU_N de la capa superior y se prepara para ser enviada como una PDU_{N-1}. MPLS toma la PDU de red y la deja y transmite intacta; únicamente coloca una redundancia MPLS entre esta cabecera y la cabecera de enlace; esto significa que si el protocolo de red tiene tramas de tamaño fijo y pequeño, MPLS transmitirá con esas características; si la capa de red es IP, por ejemplo, las tramas serán variables y potencialmente grandes; MPLS las transmitirá así también.



Protocolos de red con distintas características: distinto tamaño y variable, tamaños máximos de trama distintos…

MPLS preserva la longitud de la trama del protocolo de red; simplemente añade etiquetas delante de cada paquete.

2.1.8. Funcionamiento general de MPLS

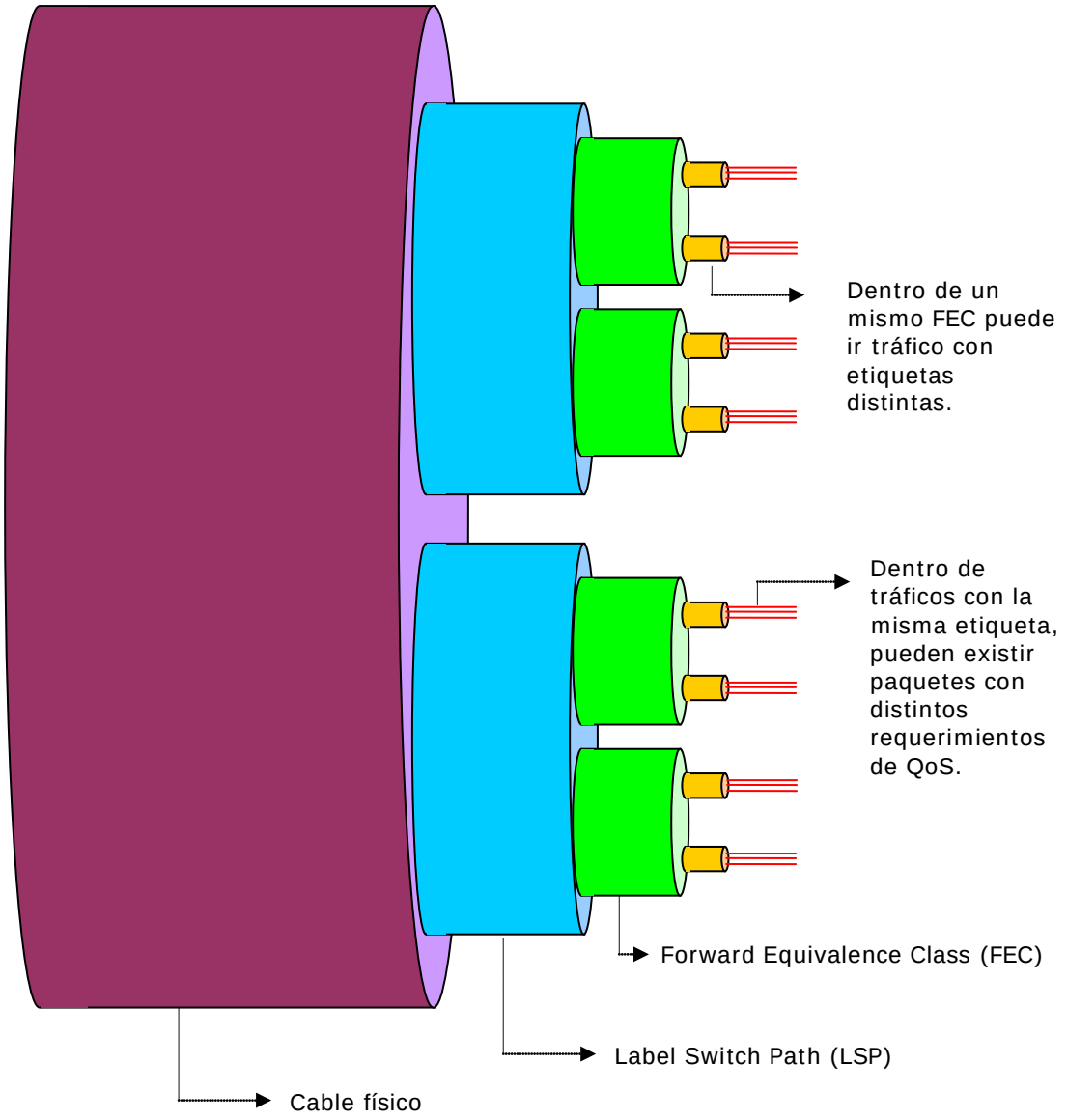
Con lo que sabemos hasta ahora del nivel al que trabaja MPLS, de la cabecera del paquete y cómo se compone le pila de etiquetas MPLS sabemos mucho en cuanto al

funcionamiento del protocolo; sin embargo en las siguientes líneas veremos el funcionamiento global del mismo.

Cuando los administradores e ingenieros de red diseñan e implementan un nuevo dominio MPLS, generalmente para unir redes de distintos tipos ya que como dijimos al principio el ámbito donde MPLS encaja más es en las troncales, se tienen que encargar de definir el significado que van a tener las etiquetas, a no ser que estén seguros de que al dominio MPLS sólo pueden entrar y sólo pueden salir paquetes cuya red sea de un único tipo, por ejemplo una nube MPLS que interconecte distintas redes IPv4 exclusivamente.

TRÁFICO DE E/S	RANGO DE ETIQUETAS	
IP versión 4	16	50
Fast Ethernet	51	150
Token Ring	151	155

Para ver el funcionamiento con un **ejemplo concreto** supongamos que información que van a tener todos los LER de entrada y salida del dominio va a ser la que se muestra en la tabla anterior.

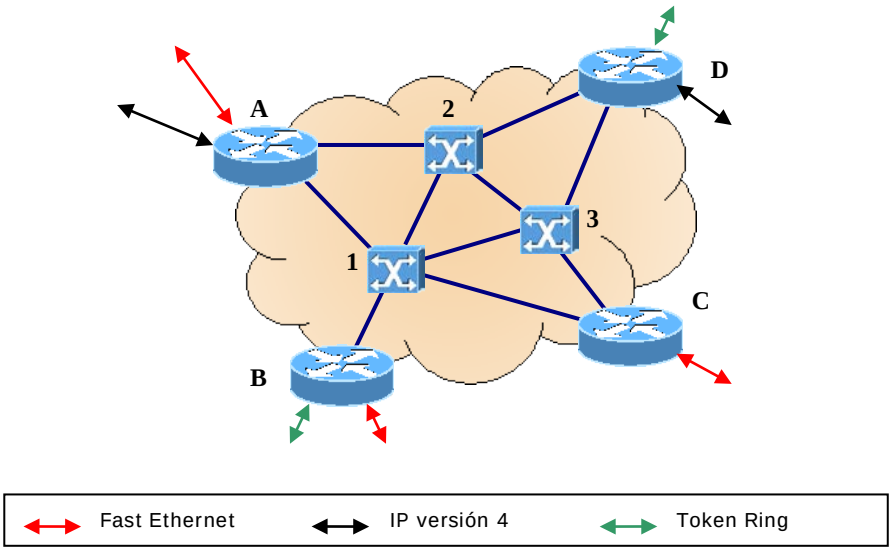


En este caso se supone que hacia y desde el dominio MPLS pueden entrar o salir paquetes desde redes IPv4, Fast Ethernet y Token Ring y para cada red se determina un rango de etiquetas de las 2^{20} posibles exceptuando las 16 que están reservadas. Ya veremos cómo, pero cuando un paquete sea etiquetado e ingrese en el dominio, si los conmutadores LSR del interior cambian la etiqueta, lo harán por otra del mismo rango para que así en todo el dominio se guarde la consistencia y no haya posibilidad de que al LER destino llegue una etiqueta que indique, por ejemplo Ethernet si lo que debe indicar es IPv6.

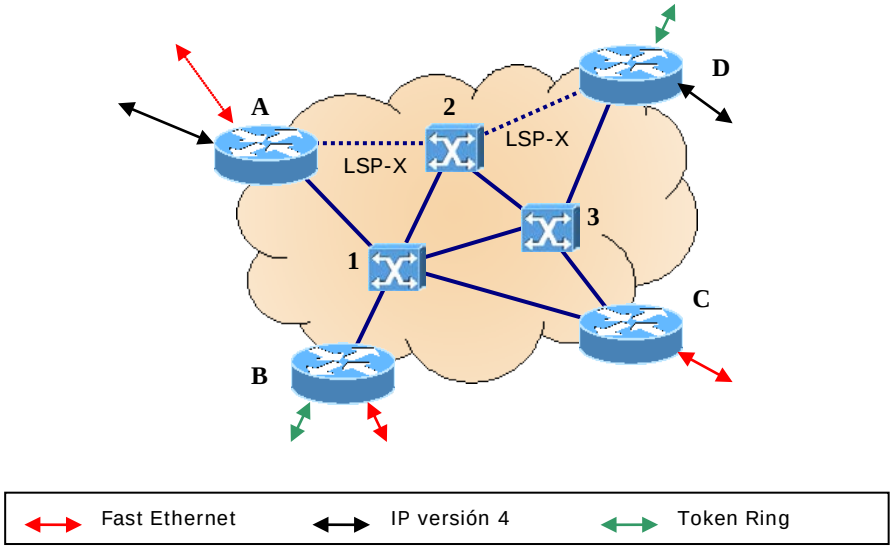
Asimismo, el rango de etiquetas se puede subdividir para que cada subrango pertenezca a un FEC o para que tenga distintos requerimientos de calidad de servicio (ya veremos como

se hace). MPLS no impone criterios en cuanto a esta cuestión sino que emplaza a las personas que se tienen que encargar de diseñar e implantar el dominio MPLS a que tomen las decisiones que crean oportunas.

Vamos a suponer que llega al dominio MPLS propuesto un paquete Fast Ethernet por el LER A que tiene que ser redirigido a una red Token Ring saliendo por el LER D. La topología podría ser:

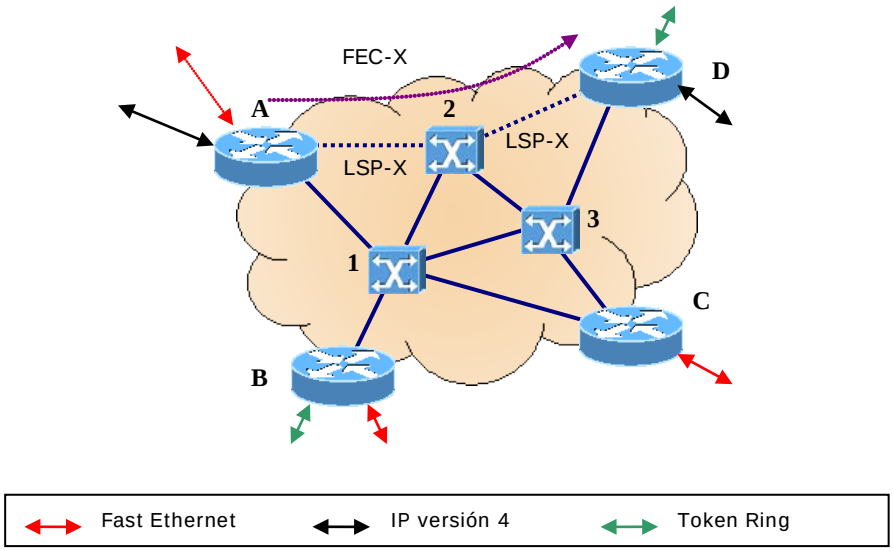


El LER A detecta que el paquete le llega por una interfaz concreta y de una red Fast Ethernet y le asigna una etiqueta entre 51 y 150 (en este caso, porque es el ejemplo que hemos definido), por ejemplo, la etiqueta 80. Detecta que ese paquete ha de salir por el LER D hacia su red de destino y establece un LSP simplex desde A a D pasando por el LSR2 (A,2,D).



Estoy obviando a propósito el proceso de creación del LSP y la distribución de etiquetas entre los LSR porque interesa explicar en este caso el funcionamiento básico de MPLS sin entrar en grandes detalles.

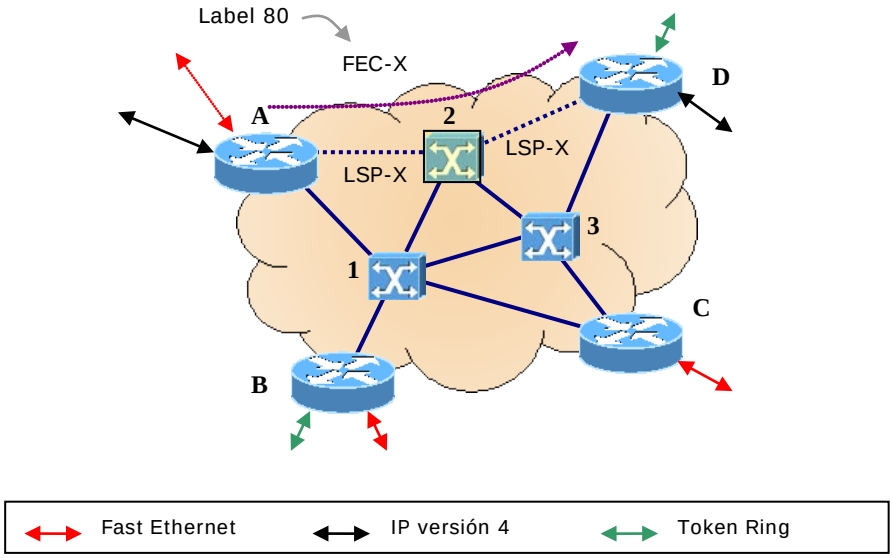
A continuación, el LER A decide crear un FEC al que llamaremos FEC-X indicando que debe viajar sobre el LSP creado, al que llamaremos LSP-X, entonces tenemos creados el LSP y un FEC que viajará sobre él. Podría aprovechar un LSP y/o un FEC que ya existiera si se ajustar a las necesidades del paquete que ha llegado, pero suponemos que no hay LSP ni FEC creados con anterioridad.



Ahora el LER A establece que los paquetes etiquetados con la etiqueta 80 pertenecen al FEC-X creado y de esta forma el paquete Fast Ethernet entrante está completamente determinado en todo el dominio MPLS porque viajará siempre, sólo o acompañado por otros paquetes, en el FEC-X por el LSP-X al que éste está asociado, hasta llegar a D. Vemos que el trabajo que hace el LER de ingreso al dominio es arduo, pero merece la pena puesto que ya no se vuelve a hacer.

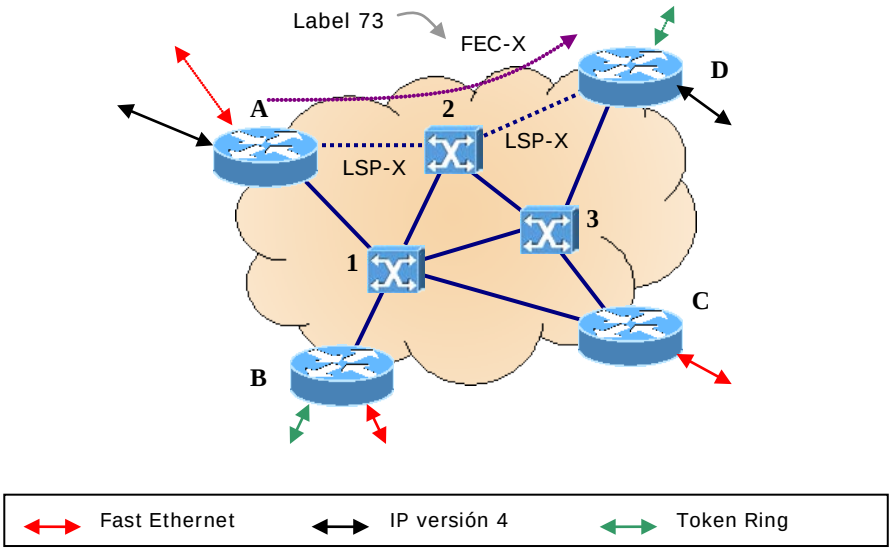
El siguiente paso que hace el LER A de entrada al dominio MPLS es sacar el paquete ya etiquetado con la etiqueta 80 por la interfaz que lo une con el LSR 2, el siguiente componente del LSP que se ha creado. Al llegar a éste, lo único que hace este LSR es detectar la etiqueta, comprobar en una tabla de etiquetas que tiene (la que no hemos visto aún cómo se crea) para ver qué nueva etiqueta le tiene que asignar y lanzar el paquete con la nueva etiqueta (sigue teniendo una pero ha cambiado) por la interfaz que lo une con el siguiente nodo del LSP, el LER D. Esta operación es sumamente rápida puesto que ya no se toman decisiones de encaminamiento sino que se hace una operación de conmutación en base a una matriz de conmutación que es en realidad lo que es la tabla de etiquetas que tiene el LSR.

La etiqueta que ponga el LSR ha de pertenecer al mismo rango que la asignada inicialmente por el LER A porque debe especificar el mismo tipo de red de entrada; tendrá un valor por tanto entre 51 y 150. Vamos a suponer que el cambio es 80 → 73, por ejemplo.



El LER D de salida del dominio MPLS recibe por una de sus interfaces un paquete etiquetado con 73 dirigido a la red Token Ring para la que el LER D hace de pasarela. Como es un LER, sabe que debe quitar la etiqueta MPLS que esté en la cima de la pila de etiquetas y reenviar el paquete hacia su destino. Detecta en este caso que nada más que hay una etiqueta (debido al campo S de la cabecera MPLS) y sabe entonces que el valor de la etiqueta es el que le especifica qué tipo de cabecera de red se va a encontrar en el paquete. Como tiene una tabla como la que describimos al principio de este ejemplo, sabe que la etiqueta 73 pertenece al rango 51-150 que está asociado con una red Fast Ethernet. Así, lee la cabecera de red como una cabecera Fast Ethernet y la interpreta.

Si el paquete fuera dirigido a una red Fast Ethernet se podría reenviar tal cual, por la interfaz correspondiente del LER de salida, pero no, en este caso va a una red Token Ring por lo que el LER D transforma el paquete que ya sabe que es Fast Ethernet en una trama Token Ring, toma una decisión de encaminamiento basándose en la cabecera y lo saca por la interfaz que lo une con le red Token Ring. Aquí acaba el trabajo de MPLS.

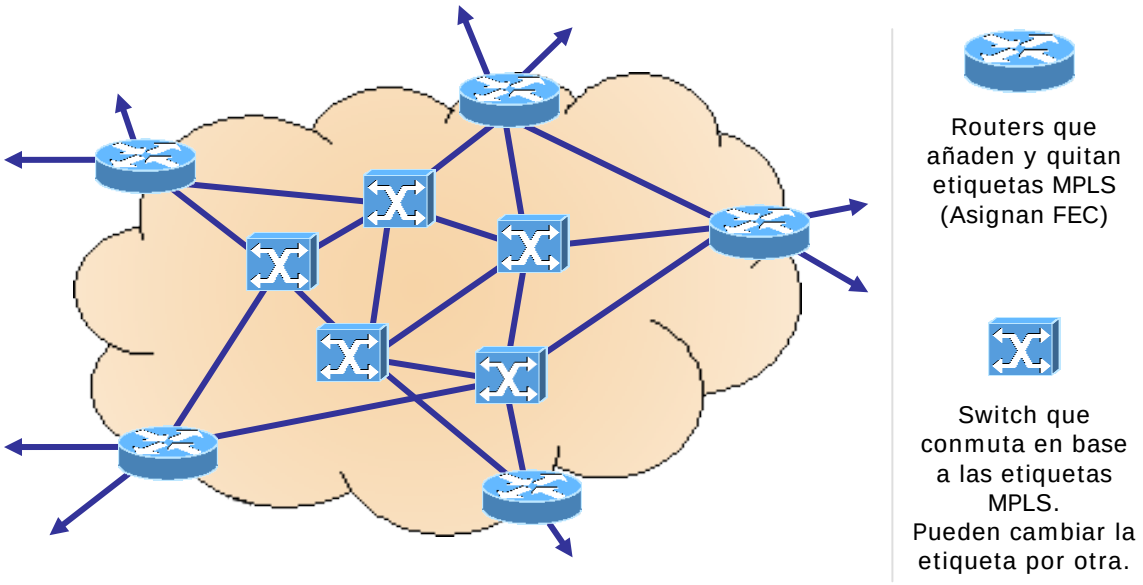


Con este ejemplo ha quedado claro el funcionamiento global de MPLS sin entrar en detalles de cómo se realiza la comunicación de control entre unos LSR y otros y entre éstos y los LER. Más adelante profundizaremos.

2.1.9. Routing en los bordes y switching en el núcleo

Como hemos visto hasta ahora, la estructura general de un dominio MPLS se basa en el concepto de que los LER del borde del dominio son los que realizan labores de *routing* de paquetes con funciones de decisión del encaminamiento que pueden llegar a ser extremadamente complicadas y que pueden estar basadas en la interfaz por la que viene el paquete, la red a la que pertenece, los valores de la cabecera de red, etcétera. Independientemente de las decisiones de los LER antes de asignar una etiqueta MPLS al paquete, los LRS del interior del dominio hacen *switching* de los paquetes que han ingresado en el dominio de forma muy veloz y eficaz sin tener en cuenta nada más que la etiqueta de la cabecera MPLS que los LER han decidido dar.

Los paquetes son encaminados en los nodos frontera del dominio MPLS y son conmutados rápidamente en los nodos interiores que ya no distinguen distintos tipos de paquete al leer exclusivamente la primera etiqueta de la pila de etiquetas MPLS.

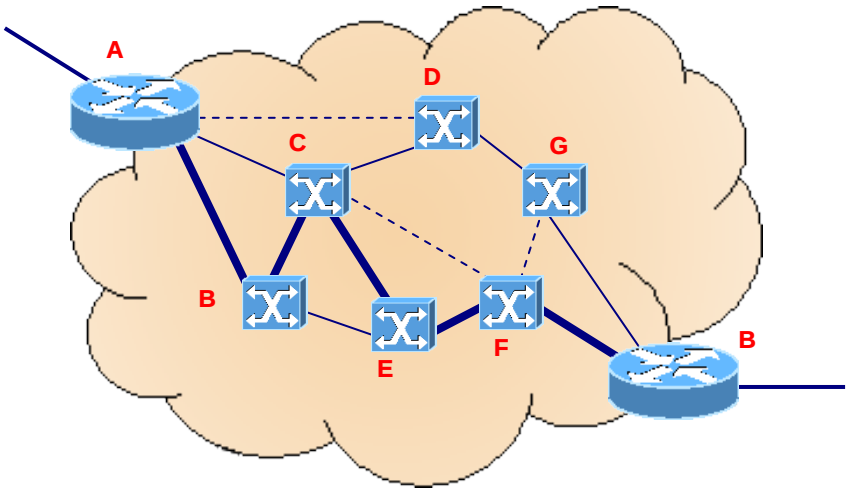


Ejemplo más concreto de un dominio MPLS

2.1.10. Soporte para servicios diferenciados

El modelo de servicios diferenciados propuesto por el IETF en los RFC 2474 y 2475 se basa en clasificar el tráfico en un número finito de clases de tal forma que para cada clase se pueda especificar una política de cola, de reenvío, etcétera distinta. Este modelo viene a sustituir el modelo de servicios integrados propuesto en el RFC 1633, poco escalable ya que en él se trata el tráfico por flujo y no por clase de tráfico así que conforme el número de flujos aumenta, el número de estados de los encaminadores aumenta exponencialmente, volviéndose esta situación insostenible.

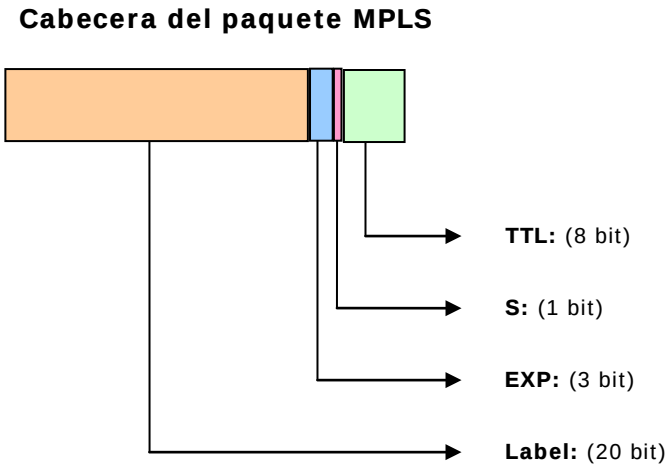
En MPLS, cada LSP puede llevar varios FEC y se puede asignar tantos flujos de información a cada FEC como se desee por lo que, a efectos prácticos, podemos elegir qué tráfico va a ser encaminado por qué cable físicos concretos (los que forman el LSP) y, de este modo se puede especificar que un tráfico vaya por fibra óptica mientras que otro va por cobre, teniendo distinta calidad de servicio únicamente por este hecho. Pero esto se puede hacer a nivel de FEC o conjunto de FEC y podría incluir a cientos de flujos distintos de información sin tener que tener cientos de estados en los encaminadores, por lo que es más escalable, como hemos comentado, que el modelo de servicios integrados.



LSP A,C,D,G,B → Medias prestaciones. UTP categoría 5e.
LSP A,B,C,E,F,B → Altas prestaciones. Fibra óptica.

Ejemplo de LSP con más o menos calidad

Pero esto es sólo una de las opciones que permite MPLS y que no está directamente relacionada con el modelo de servicios diferenciados. Para este propósito se necesitaría poder especificar junto con cada paquete la clase de tráfico a la que pertenece y de este modo podría ser tratado de una forma u otra por los LSR del interior del dominio MPLS. Volvamos a echar un vistazo a la cabecera MPLS.



Habíamos comentado que los 3 bits etiquetados como EXP estaban reservados para uso experimental. Pues bien, en el RFC 3270 este campo de la cabecera se ha redefinido y ahora se usa para especificar CoS (Class of Service), justo lo que necesitábamos para poder

implementar el modelo de servicios diferenciados. Ahora cada paquete puede llevar en su cabecera MPLS un identificador de entre los 2^3=8 posibles para que los LSR traten el paquete añadiéndole calidad de servicio

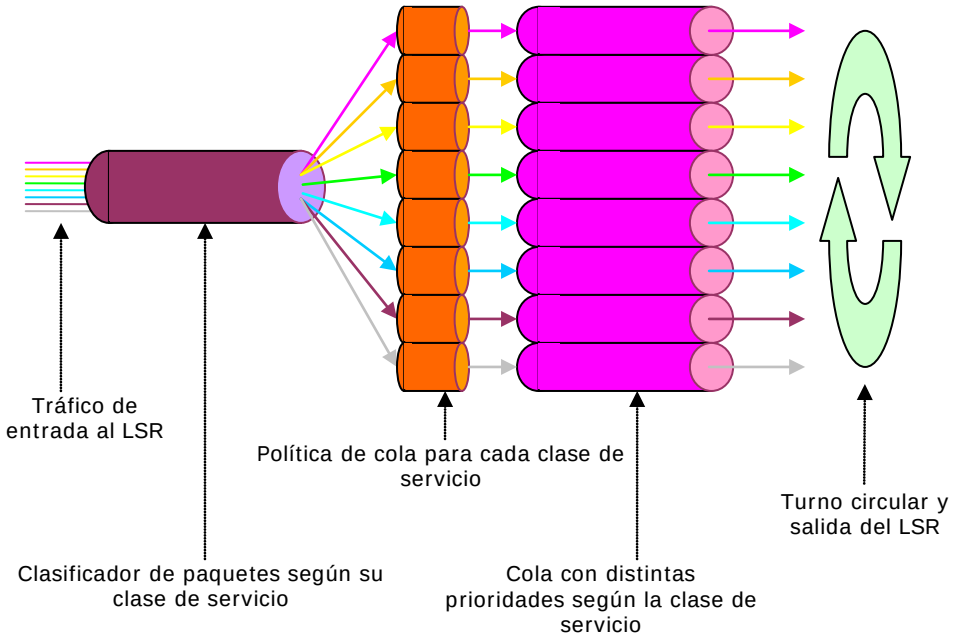
Con esto, podríamos especificar lo mismo que antes, es decir, que un LSP vaya por cierto camino físico de fibra óptica con más calidad que otro LSP que va por otro camino físico de cobre, pero además podremos llegar a especificar a un punto de detalle tal que podemos seleccionar que alguno de los paquetes que van igualmente etiquetados dentro de un mismo FEC por un mismo LSP (y por tanto por un mismo camino físico) tengan más prioridad que otra. Esto es una ventaja que veremos cuando proponga ejemplos de uso de MPLS.

Así, un LSR que soporte el modelo de servicios diferenciados de MPLS debería conmutar en base a los 20 bits de la etiqueta MPLS y de los siguiente 3 bits, correspondientes ahora a la clase de servicio. Por ejemplo, un LSR de este tipo podría tener la siguiente tabla interna:

Paquete MPLS de entrada			Paquete MPLS de salida		
Etiqueta entrada	Clase de servicio	Interfaz de entrada	Etiqueta de salida	Clase de servicio	Interfaz de salida
80	1	1	76	1	2
80	3	1	76	3	2
125	0	0	454	0	6

Donde se ve que son fácilmente diferenciables tráficos que vienen por la misma interfaz, con la misma etiqueta pero que deben tener un trato distinto porque pertenecen a clases de tráfico distinto. Pero, ¿cómo se implementa esta clase de servicio? El LSR que implementa el modelo de servicios diferenciados clasifica todo el tráfico entrante en base a su clase de servicio. Para cada clase de servicio (8 como mucho porque sólo hay 3 bits) hay una cola gestionada por una política que puede ser independiente para cada tipo de servicio: RED, DropTail, FIFO… de tal forma que resulta en ocho colas cada una con tráfico clasificado como de una sola clase (cada clase podría contener FTP, HTTP, VoIP o cualquier mezcla de ellos). Cada una de estas ocho colas debería tener distintas prioridades en función de la calidad que se quiera ofrecer para cada servicio. Al final, basándose en estas prioridades, un algoritmo de turno circulas irá despachando los paquetes. Este mecanismo tiene como

resultado algo parecido a lo que se consigue aplicando CBQ (Class Based Queuing) en TCP/IP convencional.



El modelo de servicios diferenciados es una de las formas en las que MPLS permite ofrecer calidad de servicio y aunque de momento no está ampliamente extendido su uso todo apunta a que acabará siendo esta la clave entorno a MPLS en el futuro inmediato. La descripción que he realizado sobre la forma en que MPLS soporta el modelo de servicios diferenciados del IETF es muy imprecisa e incompleta debido a que explicar este modelo no es el propósito de este trabajo sino explicar MPLS; Por ejemplo, valores requeridos por el modelo de servicios diferenciados podrían ser codificados en la etiqueta MPLS en lugar de en el campo. Así, para profundizar en el soporte MPLS para servicios diferenciados habría que referirse antes a los RFC que explican el modelo de servicios diferenciados.

2.1.11. Distribución de etiquetas

Hasta ahora, lo explicado hace referencia al plano de datos del esquema MPLS. Sin embargo, MPLS necesita de un plano de control para mantener las rutas, los LSP, las etiquetas, etcétera, que no voy a explicar, sino únicamente a comentar. Hemos hablado de que cada LSR tiene una tabla de encaminamiento que, es utilizada para la creación de los

LSP que los LER indiquen. Esta tabla se crea dinámicamente con protocolos de enrutamiento tradicionales: camino más corto, menor número de saltos… de forma que cada LSR tiene la tabla siempre actualizada y es relativamente fácil escoger la dirección de salto cuando hay varias posibilidades.

Para llevar a cabo la conmutación, la creación de LSP y algunas cosas más, MPLS necesita además de un protocolo de enrutamiento tradicional un método para que cada LSR y LER puedan mantener coherencia entre ellos en cuanto a las etiquetas utilizadas para los distintos tráficos que conmutan; no basta con que un LSR elija una etiqueta para tráfico de un FEC, sino que debe propagar esta elección al resto de sus vecinos, así se evitarán situaciones como que a un LSR llegue tráfico con una etiqueta para la cual ese LSR no tiene un entrada en su tabla. También en este caso MPLS es abierto y flexible, permitiendo la utilización de diversos protocolos de distribución como BGP, PIM o LDP. Este último es el de los más modernos y se ha creado para trabajar bien con MPLS por lo que será el que comentaré, aunque de forma muy escueta.

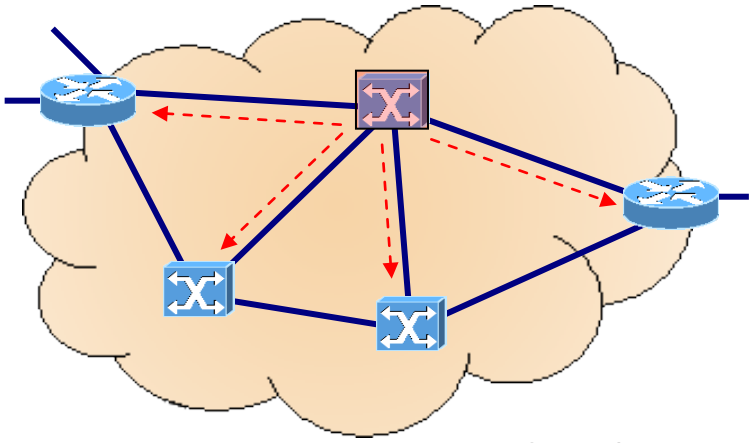
2.1.11.1. Resumen del protocolo LDP

El protocolo LDP es complejo y se sale de los objetivos de este trabajo explicarlo. Sin embargo haré un pequeño resumen.

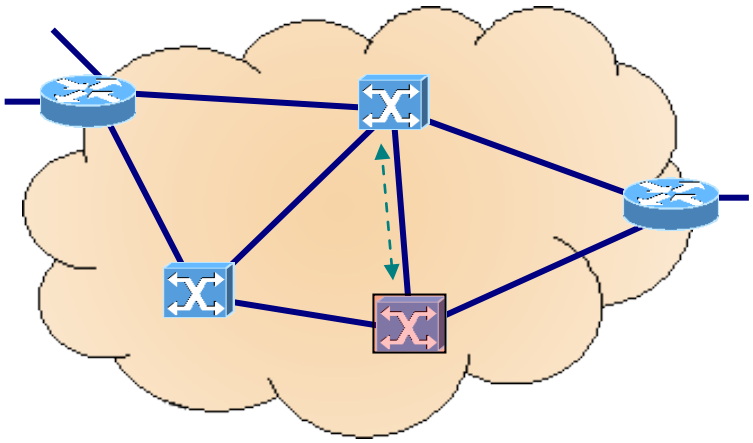
Un LER o LSR que soporte el protocolo LDP es capaz de mantener sesiones LDP con otros LSR/LER que hagan lo mismo. Durante el tiempo que dura una sesión LDP se generan varios tipos de mensajes que a continuación explicaré, y todos tienen la misma finalidad: dar a conocer a otros encaminadores que el encaminador está vivo, mantener este conocimiento, dar a conocer las asociaciones que el LSR/LER haga de etiquetas/FEC, solicitar etiquetas a otros LSR/LER, anunciar que una cierta asociación etiqueta/FEC deja de ser válida, etcétera. En definitiva, el protocolo LDP se utiliza para mantener en el dominio MPLS una coherencia en cuanto al uso de las etiquetas y las correspondencias entre éstas y los distintos FEC que pululan por la red.

Los mensajes LDP deben contar con la máxima fiabilidad pues es gracias a ellos que MPLS puede funcionar. Generalmente cuando estamos tratando con IP como protocolo de

red, un LSR/LER anuncia mediante UDP/IP que está vivo, que está listo para establecer sesiones LDP. En el caso de usar IP, lo hace a una dirección *multicast* a la que deben estar suscritos todos los encaminadores del dominio MPLS; también lo puede hacer mediante difusión. El resto de LSR/LER a su vez están haciendo lo mismo, este método es como un ping. Como parte de este mensaje "*hello*" vía UDP, viaja la dirección IP propia del servidor que se anuncia, de tal forma que si alguno de los LER/LSR que han escuchado el saludo desean establecer una sesión LDP, lo harán mediante TCP contra esta dirección de red. Si es así, la conexión se establecerá y existirá entonces una sesión LDP entre ambos LER/LSR.



El LSR marcado de rojo soporta LDP. Envía periódicamente mensajes "*hello*" por si algún vecino quiere establecer una sesión LDP con él. Lo hace vía el protocolo UDP.

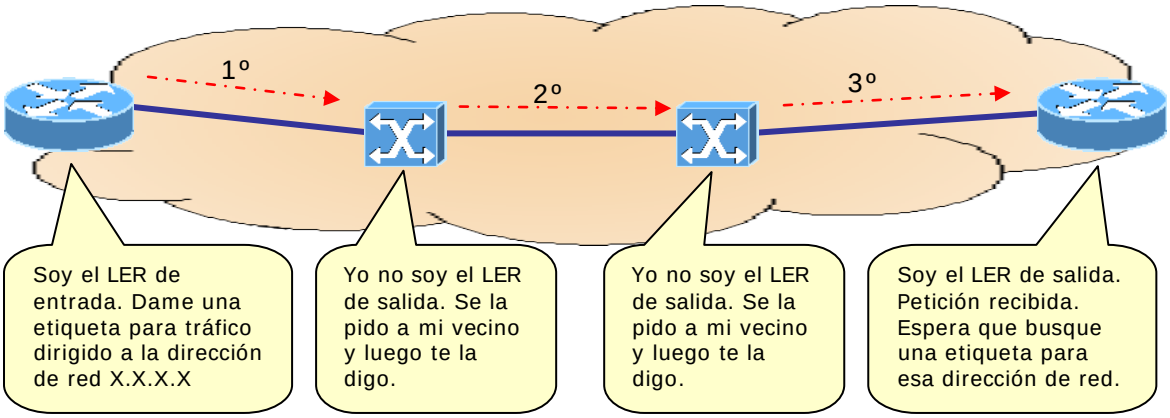


Todos los LSR/LER reciben los "*hello*" pero sólo al que está marcado ahora le interesa establecer una sesión LDP con el que envió el anuncio. Abre una conexión TCP con él y a partir de entonces sobre ella se establece una sesión LDP.

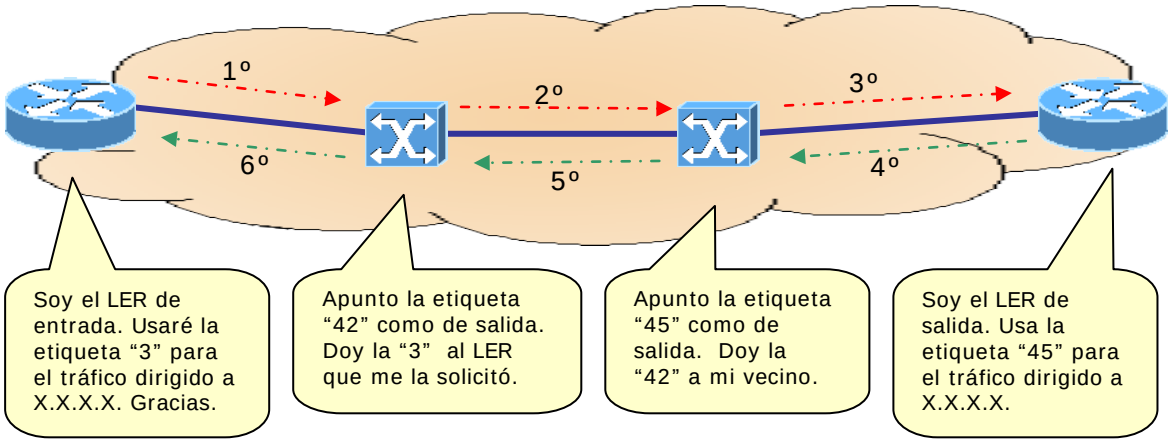
Ahora que está abierta la sesión LDP existen varios métodos por los cuales se comunica una asociación etiqueta/FEC entre los LSR/LER involucrados:

Un LSR/LER pide a su vecino que le informe de la etiqueta que debe usar para enviarle tráfico por cierto interfaz, dirigido a un destino concreto; entonces éste le responde enviándole la etiqueta para ese destino que mejor le convenga. Este proceso es una petición de etiquetas bajo demanda. Suele ser la más usual y acaba significando que la distribución de etiquetas/FEC se realiza en el sentido contrario al que va a seguir el tráfico.

Por otra parte, el otro método consiste en que cada LSR/LER realice la asociación etiqueta/FEC que quiera y entonces la transmite a sus vecinos, que las almacenaran, si desean, en su tabla de etiquetas. En este caso la información también ocurre en el sentido contrario al que al final circulará el tráfico, pero la diferencia estriba en que el vecino recibe información sin haberla pedido. De este modo se suele tener siempre las tablas de etiquetas actualizadas y a la hora de formar un FEC suele ser más rápido. Sin embargo, se eleva bastante el tráfico de control.



Solicitud de etiqueta bajo demanda para un FEC



Asociación de etiquetas y difusión de las mismas. Creación del LSP bajo demanda para el FEC concreto.

En las figuras anteriores se muestra cómo se establece un LSP bajo demanda. El LER de entrada pide a su LSR vecino una etiqueta para un tráfico que va a una dirección concreta. Esta petición se propaga hasta que llega al LER que proporciona la salida hacia la red con dicha dirección. Éste asigna una etiqueta para dicho tráfico y la propaga hacia atrás donde los LSR intermedios van encadenando el LSP con las etiquetas que ellos eligen y las que le pasan los vecinos. Al final el LER de entrada recibe la etiqueta que pidió y en todo el camino ya están creadas las tablas de conmutación para cuando la etiqueta sea usada.

2.1.12. Otros conceptos en relación a MPLS

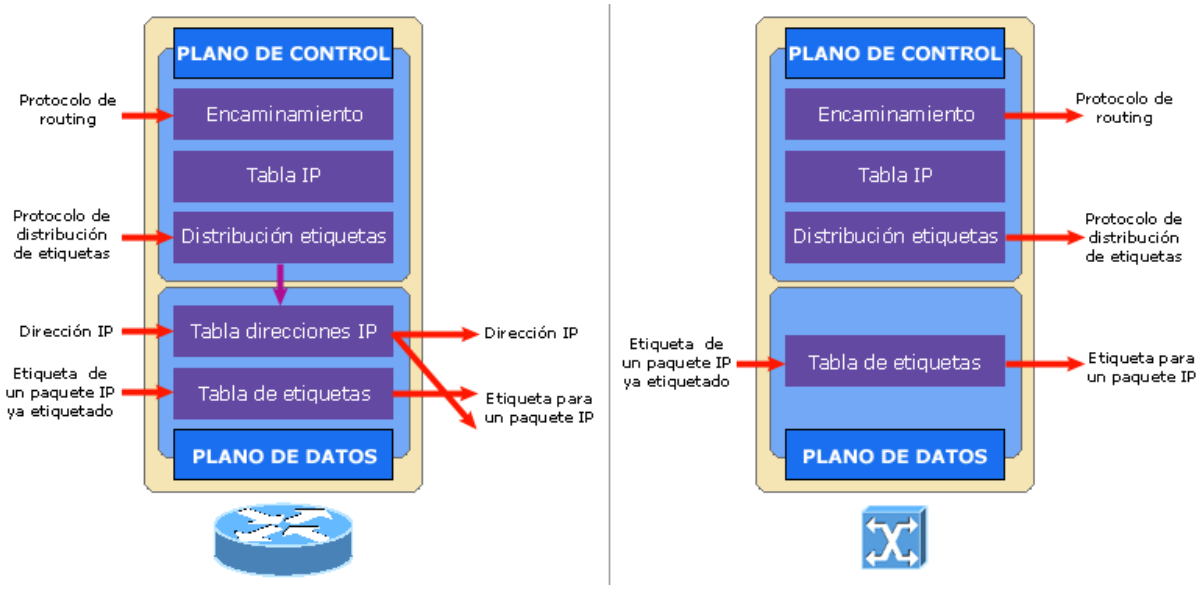
MPLS es muy extenso. No es el objetivo de este proyecto hacer una revisión completa de esta técnica de reenvío que, por otro lado, es bastante completa y compleja. Pero sin embargo hay cuestiones importantes a tener en cuenta de las que al menos se ofrecen unas pinceladas a continuación.

2.1.12.1. Esquema de un LER y de un LSR

Ahora que ya sabemos bastante del funcionamiento del protocolo MPLS y para concluir en su estudio, vamos a ver el modo de funcionamiento a un nivel de detalle mayor. Los encaminadores MPLS, tanto LER como LSR tienen un plano de control y un plano de

datos, separación de la que no hemos mencionado nada directamente pero al que no hemos dejado de hacer referencia indirectamente durante las páginas anteriores.

El plano de control está formado por aquellos protocolos y procedimientos que actúan en segundo plano manteniendo las tablas de encaminamiento de los LER y LSR, las asociaciones entre etiquetas y FEC's, etcétera y el plano de datos está formado por los procedimientos de asignaciones y modificaciones de etiquetas, de asignación de flujos a FEC, etcétera. En la figura se ve más claro este concepto.



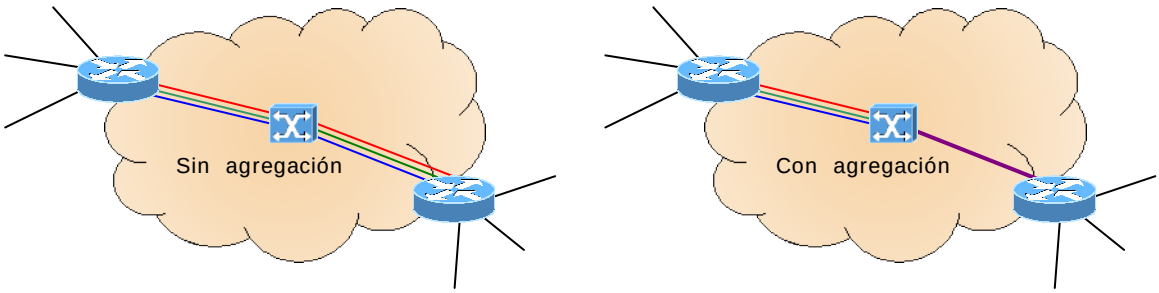
En este esquema se ve de forma mejor que la complejidad y por tanto el retardo está en el LER que corresponde a la figura de la izquierda; es éste quien hace labores de *routing* en el borde y quien se encarga de clasificar los paquetes en los distintos FEC. El esquema de la derecha es más simple y está orientado a la conmutación rápida.

Los protocolos de encaminamiento pueden ser cualquiera de los existentes: RIP, OSPF, IS-IS, BGP-4, etcétera y lo mismo ocurre con el protocolo de distribución de etiquetas que, aunque me haya centrado en LDP, realmente podría haber sido RSVP-TE, LDP-CR, BGP-4 o cualquier otro. En cuanto a la mención al protocolo IP, se hace por concretar una tecnología de red concreta, pero MPLS es multiprotocolo como hemos comentado así que se debería entender *"Tabla de red"* donde pone *"Tabla IP"*, *"Dirección de red"* donde pone *"Dirección IP"* y *"Paquete de red"* donde pone *"Paquete de IP"*. El no hacerlo es

porque posiblemente se esté más familiarizado con IP y sea más fácil comprender la función de cada módulo.

2.1.12.2. Agregación

Cuando a un LSR llega más de un FEC con la misma procedencia y destino dentro del dominio MPLS es decir, FEC que han sido creados por el mismo LER de entrada y han sido asignados al mismo LSP puede ser buena idea aglutinar todos como si fuesen el mismo FEC y asignar una sola etiqueta para ellos. De esta forma se ahorra mucha comunicación en cuanto a intercambio y almacenamiento de etiquetas.

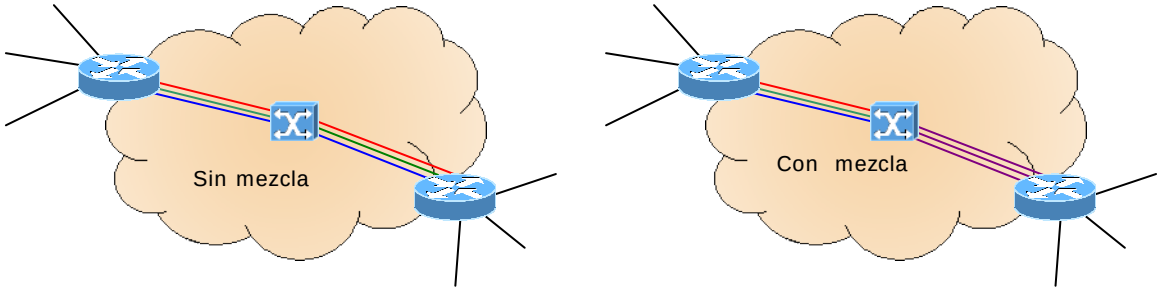


Agregación de tres FEC que llegan al LSR en uno solo, ahorrando dos asociaciones de etiquetas

El proceso de agregación implica más conceptos y es más extenso, pero solo pretendo que se comprenda para qué sirve esta operación por lo que no haré más hincapié en este aspecto.

2.1.12.3. Mezcla de etiquetas

La agregación es la capacidad de unir varios FEC en uno sólo y añadirle una sola etiqueta para él. La capacidad de mezcla de etiquetas es distinta y se aplica a diferentes flujos y/o tráficos que llegan por el mismo FEC a un LSR. Si llegan por el mismo FEC a un LSR ¿Por qué no ponerles a todos la misma etiqueta y que así todos los paquetes pertenecientes a ese FEC vayan igual etiquetados?



Mezcla de las etiquetas de tres flujos de un mismo FEC que llegan al LSR

En este caso se ahorra también espacio en el almacenamiento de etiquetas y se ahorra comunicaciones en el plano de control al no tener que mantener la asociación de muchas etiquetas. Cuando se hace mezcla de etiquetas, todo el tráfico del FEC acaba teniendo la misma y en este proceso se elimina de los paquetes toda la información referente al LER de entrada al dominio o la etiqueta que se le asignó.

Un LSR no puede realizar mezcla de etiquetas si los paquetes deben salir por diferentes interfaces hacia el mismo siguiente LSR o si deben tener distintas etiquetas al llegar al destino, por ejemplo para indicar al LER de salida que proceden de distintas redes (Fast Ethernet y Token Ring, por ejemplo).

2.1.12.4. Segmentación de paquetes MPLS

Según he comentado anteriormente, un paquete MPLS no es más que una trama de nivel de red con una cabecera MPLS. Entonces ¿Bajo que circunstancias habría que segmentar un paquete MPLS?

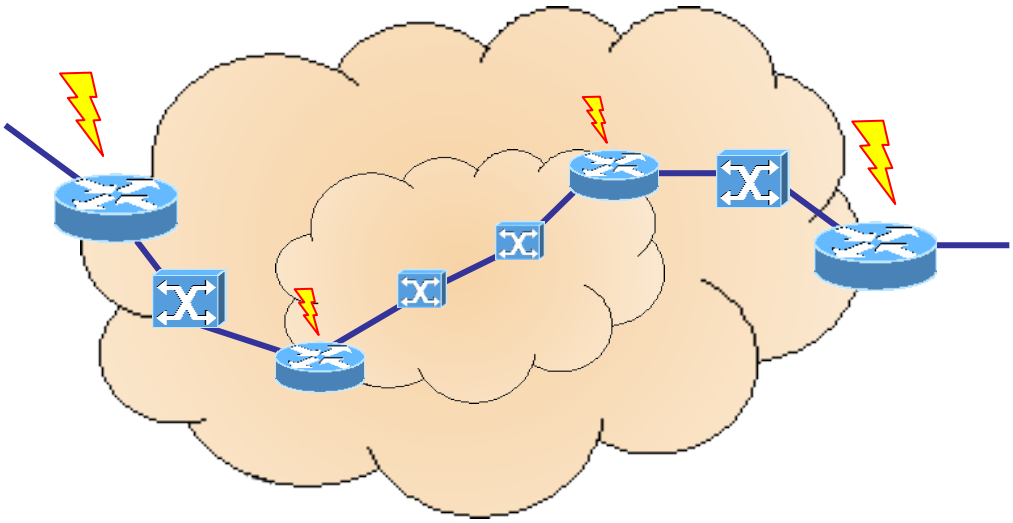
- Si un LER de entrada a un dominio MPLS separa dos redes con distinta MTU (Maximum Transfer Unit), se puede dar el caso de que el paquete etiquetado no pueda ser transportado por la red del dominio MPLS. No es un caso común porque generalmente el dominio MPLS estará haciendo labores de troncal y es probable que esté implementado sobre una red de alta calidad y alta MTU, pero no es imposible.
- Por otro lado, podría pasar que el paquete de red estuviese en el límite del MTU pero fuera transmisible por la red, pero que al añadirle los 32 bits de la cabecera MPLS, se pasara de este límite y entonces tampoco pudiera ser enviado.

En ambos casos el paquete debería ser segmentado en varios y es aquí donde empiezan los problemas. No está definido (al menos no lo he encontrado) de que manera se debe hacer esta operación. MPLS define el LSP que no es más que un camino virtual por el que la información fluye de forma ordenada y por tanto no es necesario que la cabecera MPLS tenga un campo para número de secuencia. Sin embargo si debería haber algo que indique cual de los trozos que llega es el primero de una serie de trozos de paquete y cuál es el último y así el destino podrá hacer el ensamblado.

No he encontrado información al respecto así que propongo como una posible solución que el LER que deba segmentar un paquete, utilice una combinación prohibida de etiqueta (ya que no hay otro espacio disponible en la cabecera MPLS) para informar de esta situación. En concreto, un valor “1” es válido y reservado para todas las etiquetas de la pila excepto la última. Actualmente el uso de la etiqueta “1” en el fondo de la pila no está contemplado. Esa combinación puede ser utilizada para indicar principio y fin de una operación de segmentación.

2.1.12.5. Modificación del campo TTL

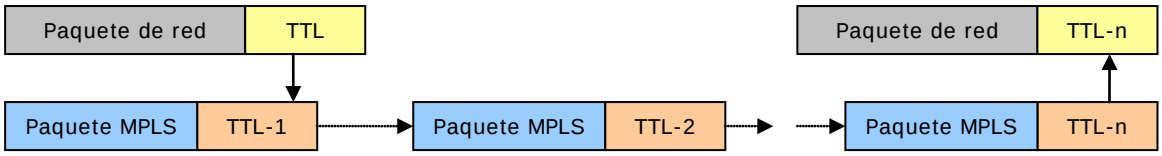
El campo TTL (Time to live) en la cabecera MPLS es de 8 bit, como en la mayoría de los protocolos de comunicaciones que cuentan con un campo igual o similar. Este campo se debe modificar cada vez que el paquete que lo lleva asociado pasa por un encaminador/conmutador.



El problema está en que con MPLS si existe el campo TTL en la cabecera de red del paquete, es ignorada puesto que se utiliza exclusivamente la etiqueta MPLS para conmutar dentro del dominio MPLS. Entonces, ¿se debe crear un nuevo campo TTL, el de la cabecera MPLS, independiente del TTL que traiga el propio paquete de nivel de red y utilizarlo de forma exclusiva en el dominio MPLS para, al final, eliminar la redundancia MPLS y dejar el paquete de red con el mismo valor TTL que cuando entro en el dominio? Esta visión tiene la desventaja de que es fácil que puedan producir bucles puesto que el campo TTL de nivel de red no se modifica realmente. Además para soportar el comando *traceroute* y otros, este método no vale.

De otro lado tenemos la opción que parece más correcta: que el LER de entrada al dominio MPLS tome el valor del campo TTL de nivel de red del paquete que ingresa y sea ese el valor de inicialización del TTL en la cabecera MPLS que se va a añadir. Los LSR interiores decrementarán este campo en cada salto (mirando exclusivamente cima de la pila de etiquetas) y al final el LER de salida debe tomar el valor del TTL y codificarlo de nuevo en el campo TTL de la cabecera de red del paquete que hora sale del dominio, de forma que el TTL del paquete de red realmente refleja el número de encaminadores por los que ha pasado. Si la cabecera de red no lleva paquete TTL, entonces el LER crea un TTL válido solo para el dominio MPLS. Si el LER, lo es de un dominio MPLS interior, el paquete entrante estará ya etiquetado y en ese caso, el valor TTL que debe usar será el que aparezca en la cima de la pila de etiquetas.

En la figura anterior se muestran con rayos los lugares conflictivos donde se debe tomar la determinación de qué TTL utilizar, si uno independiente u otro basado en el TTL de la cabecera de red.



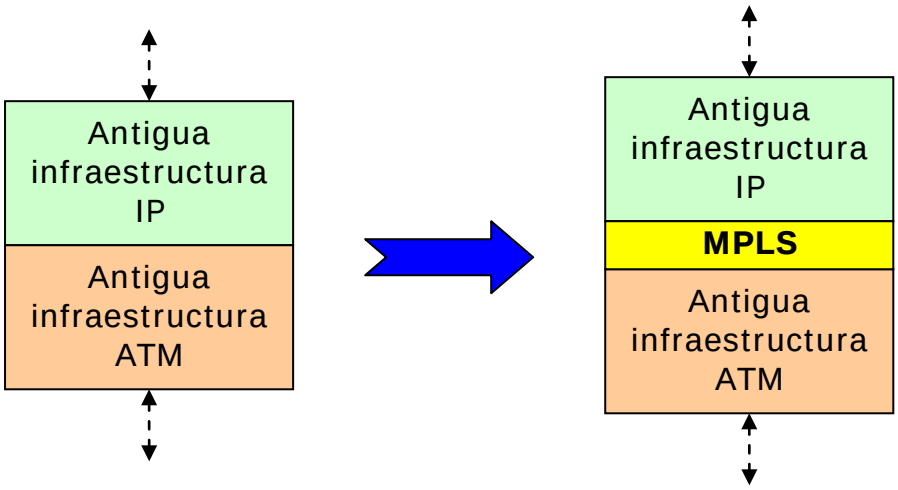
Sobre estas líneas, la figura explica cómo aún pasando por un dominio MPLS, al salir de éste, el paquete de red tiene el valor correcto de TTL conforme al número de LSR y LER por los que ha pasado. Si el protocolo de red tras el dominio MPLS no usara campo TTL, al salir del dominio, este campo se ignora.

2.1.13. Algunas aplicaciones de MPLS

MPLS puede tener multitud de aplicaciones, pero generalmente se usa para los propósitos que se especifican a continuación.

2.1.13.1. Integración de IP y ATM

MPLS es multiprotocolo como su propio nombre indica; esto significa que permite cualquier protocolo por encima, pero también permite cualquier tecnología de nivel de enlace o físico por debajo, por lo que se puede aprovechar fácilmente la infraestructura actualmente desplegada en el ámbito troncal y esto es muy importante y un punto a favor de MPLS pues facilita la migración de tecnologías aunque, evidentemente, al añadir complejidad el rendimiento siempre es peor, pero el caso es que los millones que se han estado invirtiendo desde hace años para tener IP sobre ATM pueden seguir siendo válidos para tener IP sobre MPLS y MPLS sobre ATM.



Estructura tradicional existente en las troncales de los proveedores de servicio.

Estructura de troncales de los proveedores de servicio cuando se añade MPLS. Las tecnologías anteriores siguen valiendo.

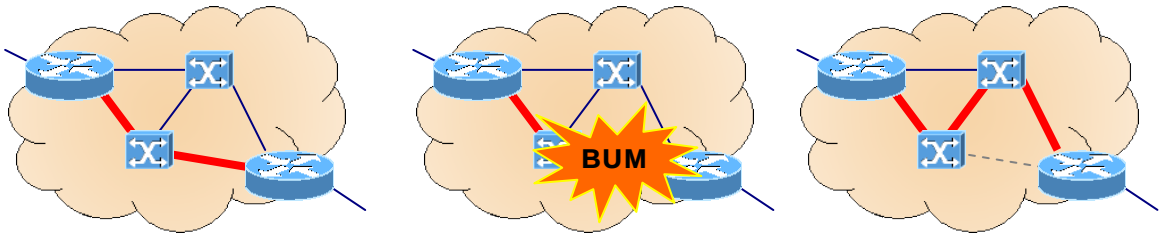
2.1.13.2. Redes Privadas Virtuales IP

Por el motivo anterior, que se soporte cualquier protocolo de red y al estar el reenvío MPLS basado en etiquetas y no en los datos que transporta, cualquier cosa que esté creada por encima del nivel de enlace, será soportada fácilmente por MPLS y así es el caso de las redes privada virtuales IP, ya se basen en túneles IP, en protocolos seguros como IP-Sec o cualquier otra tecnología. Además, se puede hacer coincidir mediante procedimientos de ingeniería de tráfico un enlace privado virtual con un FEC/LSP concreto. Como se observa, aunque MPLS es multiprotocolo se ha pensado para coexistir en todos los aspectos con Frame Relay, ATM e IP (versión 4 y 6) puesto que son las tecnologías predominantes.

2.1.13.3. Ingeniería de tráfico

El concepto de ingeniería de tráfico en MPLS se explica como las facilidades que ofrece MPLS para que una persona experta en redes y tráfico sea capaz de manipular de forma manual y en una forma rápida, los caminos por los que el tráfico pasa. Esto tiene muchas aplicaciones, por ejemplo podría utilizarse ingeniería de tráfico cuando:

- Algún problema con los enlaces entre LSR hace que un LSP no funcione. En ese momento se puede habilitar un camino o varios alternativos para evitar que la comunicación se interrumpa.



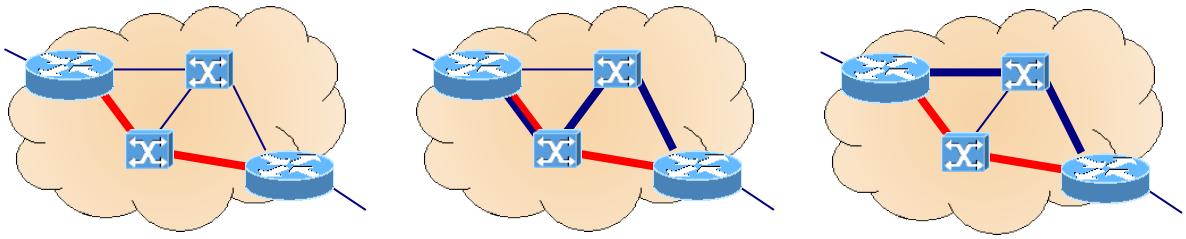
El LER de entrada crea una LSP para transportar un determinado tráfico.

De pronto, un enlace por el que va el LSP cae. No es probable, pero no es imposible.

El ingeniero de tráfico aplica ingeniería de tráfico para crear una ruta alternativa.

Ejemplo de ingeniería de tráfico para cambios de topología

- Problemas de desequilibrio del tráfico provocan saturación de la red que es evitable. Los protocolos de encaminamiento que eligen el camino más corto de los posibles pueden provocar que, pese a haber caminos alternativos, sólo usen uno y por tanto se sature. Con ingeniería de tráfico se puede desviar parte de ese tráfico por otro camino posible.



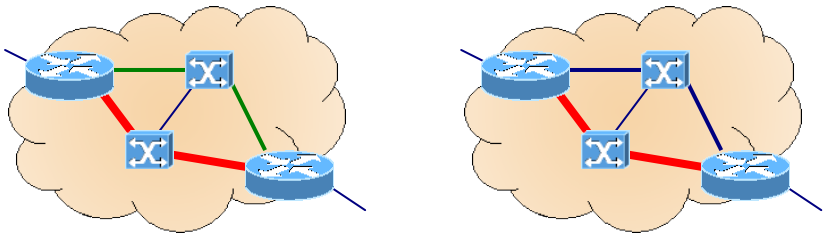
El LER de entrada crea una LSP para transportar un determinado tráfico.

El LER crea otro LSP para el nuevo tráfico. Este LSP comparte parte del camino físico con el otro por lo que el tramo compartido se satura.

El ingeniero de tráfico aplica ingeniería de tráfico, cambia los costes de los caminos para crear una ruta alternativa y evitar el colapso.

Ejemplo de ingeniería de tráfico para cambio de los pesos de los caminos para el *routing*

- El ingeniero desea que cierto tráfico fluya por un LSP concreto porque, por ejemplo, sabe que ese camino está sobre un cableado físico de fibra óptica de altísima calidad.



El LER de entrada puede crear dos LSP, el verde (cobre) y el rojo (fibra óptica).

El ingeniero de tráfico elige el rojo (fibra óptica) porque el tráfico es muy importante.

Ejemplo de ingeniería de tráfico para seleccionar LSP por la calidad del camino

Como vemos, las posibilidades de la aplicación de la ingeniería de tráfico son tremendas y diferencian, en mi opinión, una red moderna y profesional de una red pasiva, un simple cable para transportar información. MPLS permite al ingeniero de tráfico afinar mucho en cuanto al transporte de la información.

2.1.13.4. Clases de servicio y calidad de servicio

Como ya hemos comentado, la propia cabecera MPLS tiene un campo EXP de 3 bits que se ha redefinido para diferenciar las clases de servicio distintas que se consideran en la operación de reenvío por lo que se puede implementar el modelo de servicios diferenciados propuesto por el IETF. Sin embargo, hasta la fecha nada impide que estos 3 bits sean utilizados de forma arbitraria por cada proveedor de servicios para alquilar a sus clientes servicios de distinta calidad/precio al margen del modelo de servicios diferenciados. Por tanto, esta una buena aplicación de MPLS y que es radicalmente distinta al paradigma del tráfico *best effort*/servicios integrados.

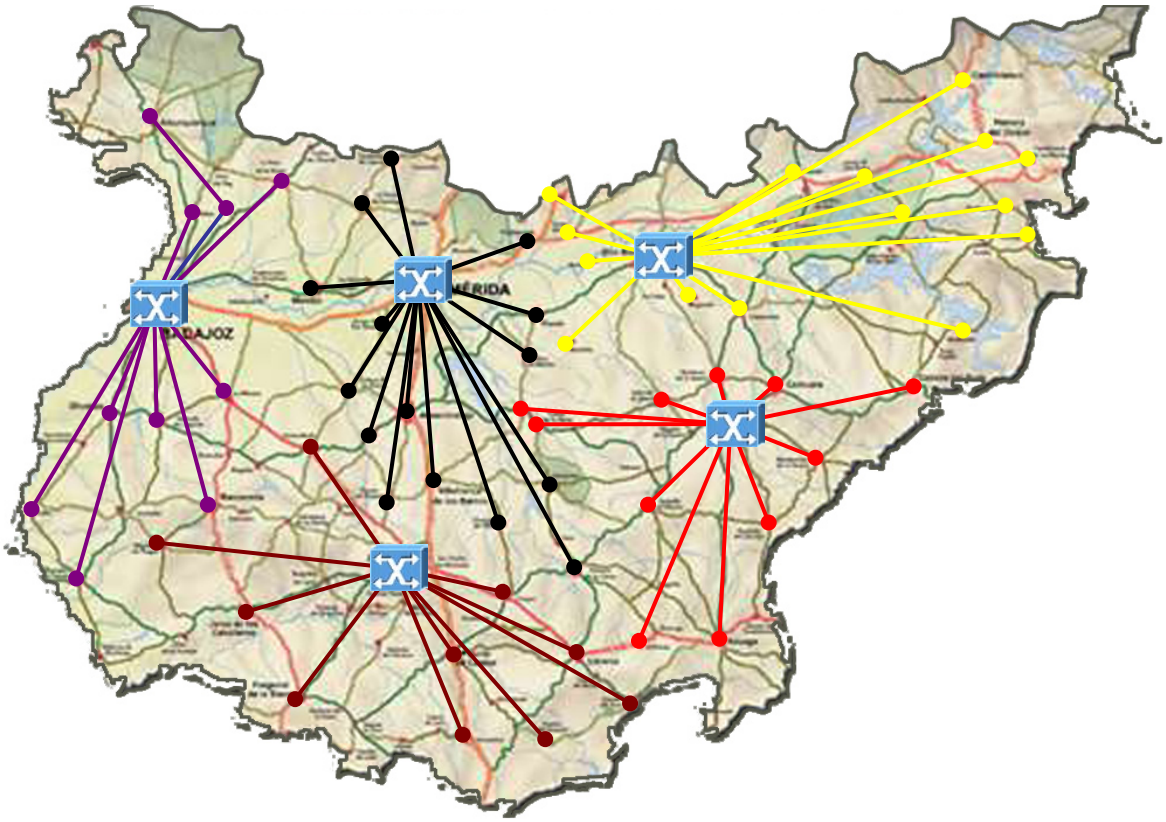
De todas maneras, MPLS no define aún de forma clara un método por el cual los protocolos de encaminamiento sean capaces de calcular una ruta con restricciones de calidad de servicio y ancho de banda. El IETF está trabajando en extensiones para los algoritmos de encaminamiento tradicionales para que soporten esta característica. Con esto se consigue que para un LSP con requerimientos de ancho de banda, se pueda calcular una función de encaminamiento tal que:

$$Funcion(P) = \sum_{i=1}^{k} \frac{1}{r_i}$$

Donde P es un camino que está formado por k segmentos de camino (saltos) y r_i es el ancho de banda disponible que ofrece el canal para cada segmento del camino.

2.1.14. Ejemplo de aplicación de MPLS

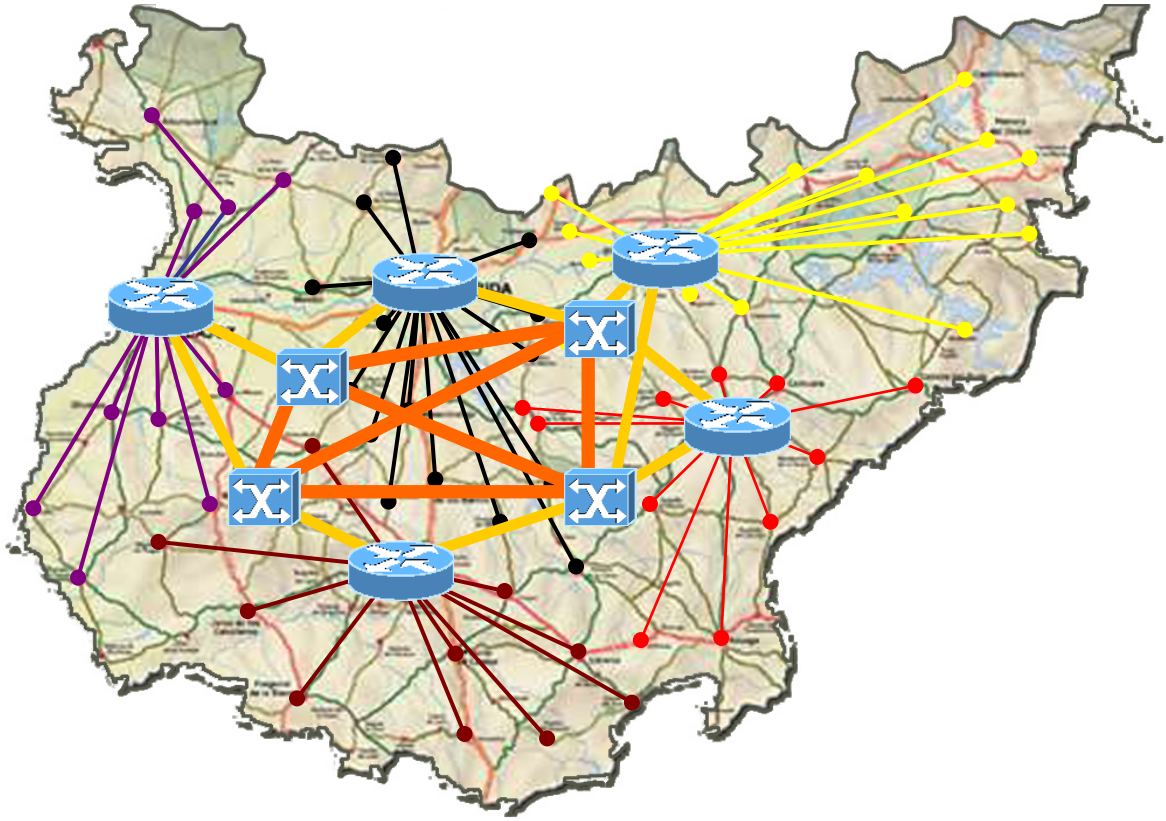
Dada la provincia de Badajoz (Extremadura, ESPAÑA), si suponemos que las ciudades de Badajoz, Mérida, **Zafra**, Don Benito y Zalamea de la Serena son centros de conmutación para cinco supuestas redes WAN para cada una de las zonas que rodean a dichas ciudades y cada una de estas WAN tiene una tecnología de red distinta: Token Ring, Gigabit Ethernet, Token Ring, etcétera (cada un de un color) la topología existente podría ser la siguiente:



En un momento dado se decide que se desean unir todas las redes comarcales para vertebrar más las comunicaciones en la provincia y hacerlo con una tecnología

vanguardista que permita la integración de todas ellas y, a ser posible con calidad; pero surge un problema: cada comarca tiene montada su infraestructura de red y ninguna quiere cambiarla por lo que no es fácil conectarlas directamente y poder garantizar un mínimo de calidad.

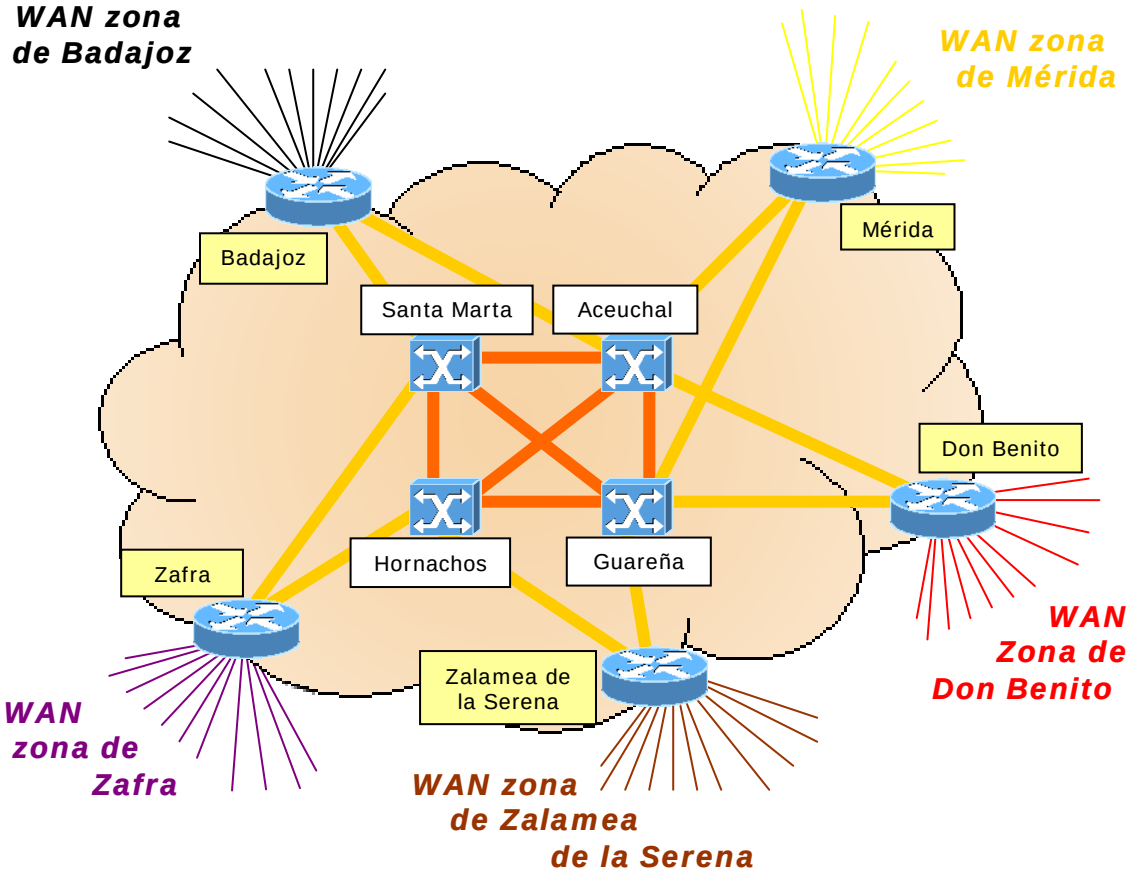
El ingeniero de redes encargado de diseñar una solución, conoce MPLS y sabe que es relativamente sencillo aprovechar la infraestructura existente y modificándola un poco se puede construir un dominio MPLS, con las ventajas que tiene, como hemos visto. La solución que piensa es la siguiente:



Es decir, convertir los actuales conmutadores de las WAN comarcales en LER de entrada al dominio MPLS que se va a crear. El núcleo de este dominio MPLS estará formado por cuatro LSR en topología de conexión total colocados a distancias intermedias en Santa Marta, Aceuchal, Guareña y Hornachos. De este modo, con un dominio MPLS haciendo labores de troncal, queda resuelto el problema de comunicación entre las distintas tecnologías de red, además de reutilizar la infraestructura existente, seguir manteniendo los

protocolos de red propios de cada WAN, proporcionar calidad de servicio y posibilidad de ingeniería de tráfico. Y si más adelante las redes comarcales cambian de tecnología, la estructura MPLS creada ahora, no necesitará cambiar.

Para ver mas clara la estructura final, voy a eliminar el mapa y a ordenar un poco los enlaces:



Con este caso real, espero haber terminado de quedar clara la organización de un dominio MPLS y cómo se usa en entornos de troncal para unir redes heterogéneas.

2.1.15. Ventajas con respecto a otras tecnologías

Ya he comentado durante todas las páginas anteriores las ventajas que ofrece MPLS; por eso aquí haré sólo una relación de algunas de ellas, las más importantes:

- MPLS es un esquema de reenvío que es independiente tanto de la tecnología de nivel de red que esté sobre él, como de la de enlace que esté por debajo. Esto posibilita que se puedan aprovechar las tecnologías existentes mientras se migra a otras más modernas, facilitando así la recuperación de las inversiones en infraestructura de red.
- Es una tecnología escalable. Debido a la estructura de la pila de etiquetas MPLS es fácil construir jerarquías de dominios MPLS por lo que se puede pasar de ámbitos más reducidos a ámbitos más globales de forma casi transparente.
- Permite aplicar técnicas de ingeniería de tráfico con lo que la red deja de se un simple elemento físico de transporte de información y se vuelve mucho más versátil.
- Permite usar cualquier protocolo de distribución de etiquetas tradicional o de última generación.
- Permite usar cualquier protocolo de encaminamiento tradicional o de última generación.
- Soporta el modelo de servicios diferenciados del IETF.
- Es orientado a la conexión por lo que los paquetes llegan en orden desde el origen del domino MPLS hasta el destino.
- No encapsula las tramas de red sino que les coloca una etiqueta, dejándolas con el mismo tamaño y propiedades que como le llegaron.
- Proporciona una conmutación basada en etiquetas que es muy rápida y eficiente.
- Realiza una única clasificación de los paquetes entrantes al dominio MPLS por lo que este proceso re reduce enormemente con respecto a tecnologías como IP.
- Proporciona un mecanismo eficiente para la realización de túneles. Los caminos LSP pueden ser usados como tales por lo que el proceso es bastante sencillo cuando se entiende ya MPLS.
- Permite ofrece caminos virtuales con calidad de servicio y ancho de banda asegurados si se usan los protocolos de enrutamiento y señalización de etiquetas convenientes.

2.1.16. Problemática de MPLS

MPLS es tan flexible precisamente porque no ha definido los detalles de cómo se debe utilizar ciertas características o el modo en que se deben implementar otras, sino que lo deja en manos de cada fabricante de soluciones de red. Esto, unido al hecho de que MPLS

es un protocolo muy joven ha hecho que haya funcionalidades de MPLS como la calidad de servicio, las clases de servicio, la reestructuración dinámica de los LSP cuando un enlace cae, el control del tráfico en ese caso y un montón de características más que no se están usando a pleno rendimiento actualmente, sino que se están aún especificando estándares y borradores al respecto.

Aunque en principio parece que el futuro de MPLS es halagüeño, deben pasar aún varios años para que llegue al estado de madurez en el que al menos las características principales estén bien y claramente definidas.

Por otro lado, al ser el mundo IP el que predomina en cualquier parte del mundo, tanto la industria del hardware como los investigadores, están desviando un poco su atención de la característica multiprotocolo de MPLS y están creando muchas soluciones IP/MPLS/ATM para dar solución a lo que hoy en día se encuentra en cualquier troncal. Esto es bueno por un lado, pero es bastante difícil encontrar información sobre cómo adaptar otras tecnologías de red a MPLS o incluso, encontrar hardware, *routers*, conmutadores, etcétera que soporte otra cosa distinta de IP; por ejemplo, en la bibliografía existente generalmente se indica que las etiquetas están normalmente relacionadas con el rango de direcciones IP al que va dirigido el tráfico; pero ¿y si no se está usando IP? ¿y si se asignan las etiquetas conforme a otros criterios que no sean el destino del tráfico? En este trabajo estoy intentando dar una visión global de MPLS sin asociarlo con IP, aunque a veces se hace realmente difícil pues la documentación existente es monotemática en este sentido.

2.1.17. ¿Por qué la coexistencia IP/MPLS?

Los intentos de emparejar IP con el concepto de calidad de servicios siempre se han visto avocadas al fracaso debido principalmente a la filosofía de no orientación a la conexión llevada por IP. ¿Cómo se puede reservar recursos para un tráfico si no se sabe a priori por dónde va a circular? Este es el meollo de la cuestión. El protocolo RSVP (ReSerVation Protocol) nació con la idea de proporcionar reserva de recursos a IP pero chocó de frente con este problema. El protocolo en sí funciona pero IP no responde de forma suficientemente determinista como para que llegue a ser una solución factible.

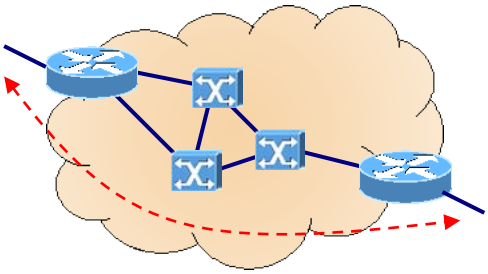
Con la aparición de MPLS, se ha visto la posibilidad de reutilizar ideas pasadas. Ya que MPLS ofrece orientación a la conexión para cualquier tipo de tráfico y en concreto para tráfico IP, se puede transportar tráfico IP sobre MPLS y utilizar RSVP para reservar recursos para el LSP que crea MPLS. Con esta solución además se puede integrar el actual tráfico IP y ofrecer, aunque sólo sea en las troncales, cierta calidad de servicio.

Además, como MPLS soporta el modelo de servicios diferenciados, la unión de este modelo con MPLS y con el protocolo RSVP permite que las aplicaciones adquieran los requisitos de calidad de servicio. Si el protocolo de red tiene tramas de enorme tamaño, la sincronización de tráfico de audio y vídeo será complicada, pero no es un problema de MPLS sino del protocolo de red. Sin embargo, al menos el ancho de banda, la orientación a conexión y la rapidez en la conmutación estarán asegurados.

Actualmente varias empresas implementan esta técnica, pues el modelo de servicios diferenciados en MPLS está en un estado aún no muy maduro; En concreto, Telefónica de España lo usa para hacer converger todas las tecnologías actuales de conectividad que posee: RTB, XDSL, ATM, RDSI, IPv4, Gigabit Ethernet…

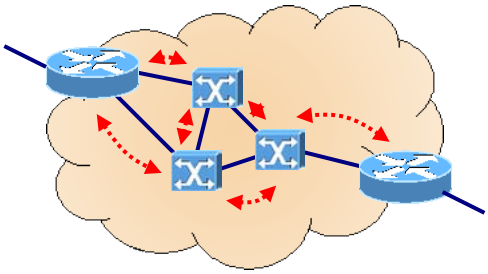
2.1.18. Tendencias

De un tiempo a esta parte parece claro que la mayor parte de los ingenieros tienen las ideas más o menos claras: las redes físicas son cada vez más fiables; fibra óptica, cableado estructurado y cables de categorías cada vez más altas con una frecuencia potencial de errores mínima están siendo cada vez más habituales. Por tanto, no parece que merezca la pena crear protocolos que se encarguen en demasía del control de errores, de las retransmisiones, etcétera.



Red con control de errores extremo a extremo. No ser pierde tiempo ni recursos en comprobar errores, corregirlos o hacer retransmisiones en el interior de la red.

Con una red fiable es muy eficiente



Red con control de errores punto a punto. Se pierde mucho tiempo y recursos en comprobar errores, corregirlos o hacer retransmisiones en el interior de la red.

Con una red fiable es poco eficiente

Partiendo de esta idea y de que los equipos terminales son suficientemente potentes, se están creando protocolos muy simples en cuanto a su estructura y se deja, cada vez más, el control de la transferencia a los *hosts* extremo a extremo, y no punto a punto. MPLS no es una excepción y por tanto funciona de esta forma. La preocupación principal está, por tanto, en fabricar conmutadores más veloces cada vez ya que es ahora en este punto donde se encuentra el cuello de botella de las redes.

Además, la mayor parte de las investigaciones tienden ya no tanto a proporcionar un servicio, sino a proporcionarlo con calidad, con ciertos parámetros que permitan soportar tráfico de audio, multimedia y sobre todo, de distintos usuario con distintas características sobre la misma red. En torno a este aspecto están centradas las mayores pruebas e investigaciones de MPLS; incluso el protocolo LDP para distribución de etiquetas se ha modificado para ser más útil a la hora de que MPLS pueda soportar clases de servicios diferenciados.

Por tanto se buscan protocolos sin muchas parafernalias, con el control fuera de banda, con una red única para todos los tipos de tráfico y distinción entre todos ellos para otorgarles más o menos recursos en las que el trabajo más pesado se realice extremo a extremo por los nodos emisores y receptores bajo la suposición de que la red física es fiable. Aquí MPLS tiene mucho que decir.

2.1.19. Conclusiones

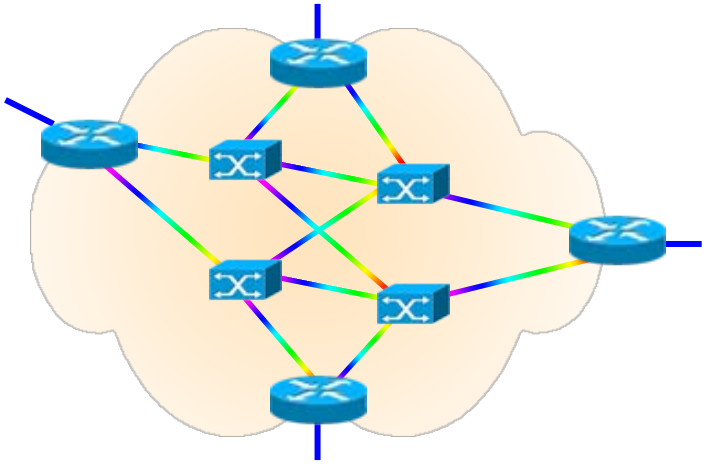
Aunque existen muchas tecnologías en estudio para solucionar el tema de la calidad de servicio, para poder proporcionar ancho de banda, etcétera, en definitiva, para crear redes modernas adaptadas a las necesidades de caracterizar el tráfico en la medida de lo posible, parece que MPLS es una solución de consenso, una solución que satisface a todas las partes implicadas en el mundo de las comunicaciones.

MPLS es un protocolo que sigue el principio de ser compatible con lo que existe en el mercado actualmente, con posibilidad de sustituir estas infraestructuras en el futuro sin problemas, una tecnología abierta y flexible que permite una buena gestión del tráfico y con una relativa facilidad de implementación. Además permite ofrecer calidad de servicio extremo a extremo en las troncales, diferenciación de tráfico, optimización de los recursos, balanceo de carga de la red y multitud de cosas más.

En mi opinión, MPLS tiene amplias posibilidades de futuro porque es una tecnología que permite ser adaptada a las necesidades de cada cual aprovechando las infraestructuras actuales de forma más óptima y, sin duda, porque las empresas componentes del MPLS Forum han apostado claramente por desarrollarla. Además, es un intento realista de cambio de la situación en las redes; no intenta cambiar radicalmente cosas que de hecho nunca cambiarían en el ámbito local. Es una solución muy buena para gestionar el tráfico de las troncales.

2.2. Introducción a GMPLS

El esquema de MPLS tradicional permite algo realmente innovador: integrar en una única red de transporte tecnologías tan dispares como IP, ATM y Frame Relay, con lo que permite ahorrar el coste de inversión y actualización de las redes actuales a la vez que provee de los mecanismos necesarios para poder realizar ingeniería de tráfico, especificar clases de servicios, etcétera.



MPLS no puede aprovechar correctamente las características de las redes ópticas.

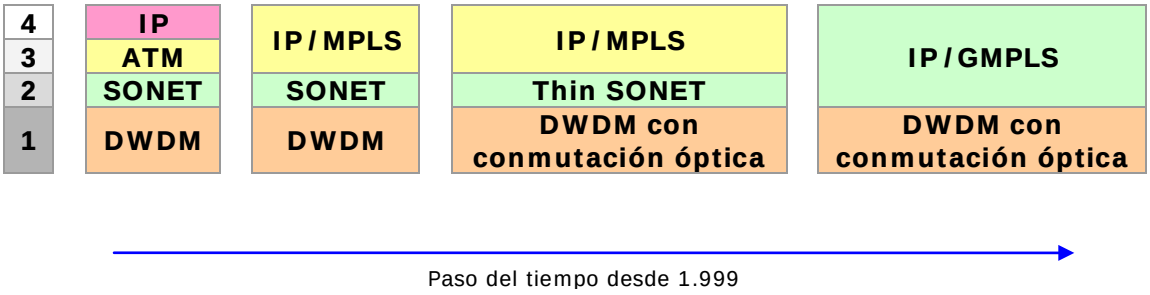
Sin embargo, la tecnología MPLS no permite tener control ni aplicar técnicas de ingeniería de tráfico sobre redes de conmutación por división de longitud de onda (DWDM), de multiplexación por división de tiempo (TDM), etcétera, ya que MPLS se encuentra por encima del nivel en que trabajan este tipo de redes y no tiene métodos de señalización ni encaminamiento para ellas; Ha sido necesario rediseñar MPLS y un gran número de protocolos de todos los niveles implicados, para ser capaz de adaptarse a este tipo de redes con tecnologías tan distintas y poder situar a MPLS un nivel más abajo, incluso en el nivel físico. A esta tecnología rediseñada a partir de MPLS y de MPλS (un primer acercamiento de MPLS a las redes ópticas) es a la que se llama *Generalized Multiprotocol Label Switching* (GMPLS) o conmutación de etiquetas multiprotocolo generalizado.

2.2.1. Evolución de MPLS a GMPLS.

GMPLS surge de forma natural como una evolución de MPLS y de MPλS. No olvidemos que MPLS se encuentra en el nivel 2+ de la pila de protocolos. Sin embargo poco a poco las redes tienden a simplificarse a medida que aumenta la calidad de las mismas compactando los niveles definidos por el modelo de referencia OSI de la ISO hasta límites insospechados.

Aún así, el surgir de las redes ópticas ha cambiado mucho el mundo de los protocolos de comunicaciones. Las redes ópticas no están pensadas, diseñadas, ni preparadas para poder inspeccionar el contenido de la información que transportan por lo que difícilmente se

podrá conmutar en base a dicha información. Más bien se hace necesario dispositivos de nivel físico o poco más, capaz de conmutar en base a la señal portadora de la información.



Surgen aquí un sinfín de problemas como el hecho de la creación de etiquetas para los paquetes GMPLS, el soporte de distintos tipos de conmutación a nivel físico, de enlace o de red, problemas de compatibilidad entre dispositivos de estos tipos, etcétera.

La primera de las cuestiones era definir por tanto tipos de interfaces a través de los cuales GMPLS podría conmutar la información; cada uno de estos tipos asociado a un nivel concreto y/o a una tecnología, pero todo ello regulado para poder estandarizar GMPLS con éxito. De este modo se podrá construir un superdominio GMPLS que contenga subredes de distinta tecnología y el tráfico pueda fluir de extremo a extremo de forma transparente usando GMPLS como tecnología integradora en todo su conjunto. Las adaptaciones del tráfico se deberán hacer mediante convertidores haciendo uso de los interfaces que comentábamos.

A continuación se muestran estos tipos de interfaces.

2.2.1.1. Packet Switching Capable (PSC).

En GMPLS, una red es PSC o tiene la capacidad de conmutar en base a paquetes cuando está formada por dispositivos que conmutan en el dominio de los paquetes o las celdas; con tráfico, por ejemplo, como IP o ATM. En este tipo de entornos el esquema de reenvío de paquetes se realiza en base a las etiquetas normales que conocemos de MPLS, colocadas entre las cabeceras de red y la de enlace o bien en base al identificador de canal, VCC, de ATM.

Algunos ejemplos de dispositivos PSC son los tradicionales encaminadores IP, conmutadores ATM, etcétera.

2.2.1.2. TDM Switching Capable (TSC).

En GMPLS un interfaz TSC es aquella que tiene la capacidad de conmutar en el dominio del tiempo. Un ejemplos del tráfico que circula por este tipo de interfaces puede ser TDM/SONET, como podemos ver, un dispositivo TSC trabaja a un nivel bastante más bajo que otro PSC. El esquema de reenvío en TSC es el *Round Robin* con rodajas de tiempo cíclicas para cada uno de los flujos de información que deban atravesar el dispositivo.

Por ejemplo, son dispositivos TSC los sistemas digitales de interconexión *(Digital cross-connect system).*

2.2.1.3. Lambda Switching Capable (LSC).

LSC, como lo define GMPLS, son aquellas interfaces que trabajan en el dominio de las longitudes de las ondas (luminosas). En este tipo de dispositivos entramos en un terreno claramente hardware donde las operaciones se llevan a cabo realizando modificaciones a la señal portadora (la onda) y donde el tráfico que se está transportando es totalmente transparente al dispositivo, que no puede examinarlo. En este tipo de dispositivos el reenvío de la información se lleva a cabo mediante la diferenciación entre unas longitudes de onda y otras, en lo que se conoce como etiquetamiento implícito basado en las landas.

Los dispositivos DWDM para el esquema GMPLS, son dispositivos LSC.

2.2.1.4. Fiber Switching Capable (FSC).

Por ultimo, los dispositivos que trabajan a más bajo nivel, las interfaces FSC son aquellas cuyo dominio de actuación es el medio físico directamente. Al igual que las interfaces LSC, que trabajan a un nivel de abstracción mayor, el tráfico que circula por estas

interfaces es transparente, incluso conceptos de tan bajo nivel como una longitud de onda. Estos dispositivos realizan conmutación de fibras ópticas completas, de líneas de comunicaciones, etcétera.

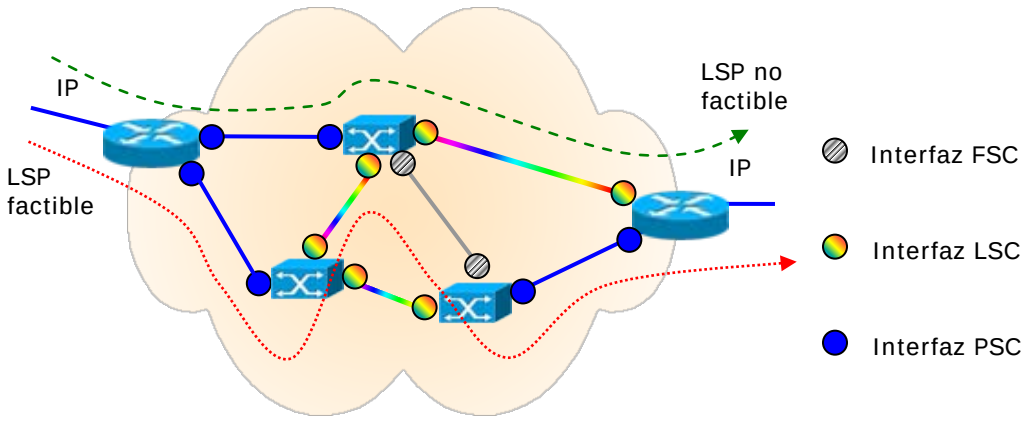
Ejemplos de interfaces PSC son por ejemplo los OCX.

2.2.2. Introducción al funcionamiento de GMPLS.

El funcionamiento general de GMPLS no difiere demasiado del funcionamiento de MPLS puesto que es una generalización del mismo. GMPLS está siendo definido para las interfaces que se han comentado unas líneas más arriba lo que significa que entre dispositivos PSC puede haber un LSP, al igual que entre dispositivos TSC, LSC y FSC. Esto supone que GMPLS actúa en todos los niveles existentes en la comunicación. Pero más allá, GMPLS permite también crear LSP entre nodos heterogéneos (y por tanto redes de tecnologías distintas) y es aquí donde se muestra la funcionalidad mayor de GMPLS.

GMPLS permite el establecimiento de un dominio MPLS con distintos dispositivos de distintas tecnologías trabajando a niveles distintos, de forma simultánea y transparente siempre y cuando los dispositivos cuenten con interfaces englobadas dentro de alguno de los cuatro tipos definidos.

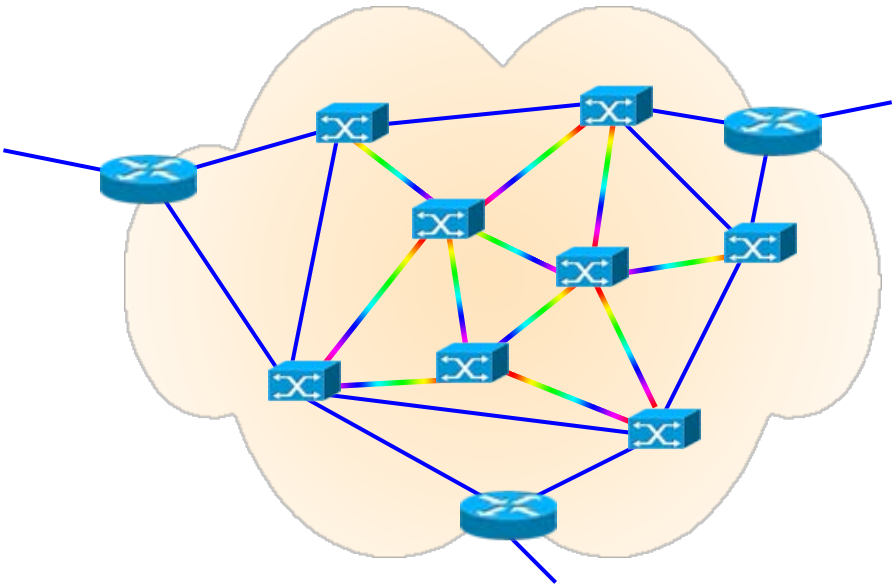
La única restricción existentes es que **las interfaces origen y destino de los LSP creados deben tener la misma capacidad**, es decir, ser del *mismo tipo* (PSC, TSC, LSC o FSC).



GMPLS permite la creación de LSP sobre redes de distinto tipo a distintos niveles

En el ejemplo anterior observamos que el LSP en rojo es factible. GMPLS permite establecer este tipo de LSP. Sin embargo, el LSP en verde no sería factible al amparo de GMPLS puesto que comienza en una interfaz PSC y termina en una interfaz LSC, de distinto tipo por tanto.

Esto está pensado así, porque generalmente la estructura típica de una red de tecnología GMPLS será de múltiples capas, generalmente representadas como concéntricas como si de subdominios MPLS interiores se tratara, de tal forma que los LSP generalmente se establecerán entre nodos pertenecientes a la misma capa. No es una obligación pero por motivos organizativos y de coherencia suele hacerse así.



Estructura típica de una red GMPLS

De esta forma, se suelen colocar dispositivos con interfaces PSC de forma más abundante en la periferia del dominio GMPLS y posteriormente, a medida que nos acercamos al núcleo del mismo, colocar TSC, LSC y FSC paulatinamente y reduciendo su número respectivamente.

Generalmente dispositivos de más bajo nivel pueden conmutar mucha más información en una sola operación que dispositivos de alto nivel. Por ejemplo un dispositivo capaz de conmutar grupos de fibras directamente, podría estar conmutando cientos de miles de FEC

existentes entre entidades PSC, simultáneamente; Cada uno de esos FEC podría contener cientos de LSP y éstos a su vez multitud de flujos de información.

2.2.3. Protocolos comunes de señalización y control

En MPLS se podían utilizar distintos protocolos de señalización para la distribución de etiquetas y para el mantenimiento de los LSP; no obstante, se ha extendido el uso de LDP y CR-LDP para este menester. GMPLS, como tecnología posterior y derivada de MPLS sigue este paradigma de una forma un poco distinta. Se ha creado un nuevo protocolo llamado LMP (Link Management Protocol) o Protocolo de Gestión del Enlace y usualmente se combina con RSVP, que vuelve a cobrar protagonismo en la escena, para obtener reserva de recursos para los LSP que se establecen. CR-LDP se sigue usando, pero todo indica que el binomio LMP-RSVP tiene todas las de ganar en esta batalla, por el momento.

- **LMP** es un protocolo encargado de establecer, mantener y gestionar los enlaces entre dos pares de nodos. Funciona sobre IP y se encarga, entre otras cosas, del envío de etiquetas generalizadas (se verá más abajo) para crear tramos del LSP que se pretende establecer extremo a extremo. Funcionalmente es muy parecido a LDP, pero con muchas mejoras introducidas para poder gestionar correctamente etiquetas generalizadas y a los distintos tipos de codificaciones de los canales de comunicación.

- Por su parte, **RSVP** (Resources ReSerVation Protocol) o Protocolo de Reserva de Recursos, ha sido convenientemente modificado para dar soporte a la ingeniería de tráfico sobre GMPLS y funciona bastante bien como complemento de LMP para el establecimiento de TE-Links o enlaces con ingeniería de tráfico. RSVP no tuvo el esperado éxito en su aplicación sobre comunicaciones IP por la falta de orientación a conexión de este protocolo; sin embargo, con el establecimiento de un camino constante para la comunicación, su utilidad vuelve a resurgir.

En los siguientes puntos veremos cómo se combinan estos protocolos para mantener coherencia en la señalización para GMPLS.

2.2.4. Etiquetado en GMPLS

Tenemos que GMPLS incorpora a MPLS, entre otras cosas, la capacidad de crear caminos conmutados e base a etiquetas a distintos niveles, bajando incluso al nivel físico de la red; Más novedoso resulta, incluso, que todos los dispositivos independientemente del nivel en el que se encuentren puedan "hablar" GMPLS entre sí.

Sin embargo, hemos comentado que las redes ópticas, por ejemplo, en el nivel físico o incluso de enlace, no están preparadas para examinar el contenido de las etiquetas MPLS por lo que en principio habría que idear otro método para que dispositivos de niveles cercanos al físico puedan comprender cómo y cuándo deben conmutar. Es ahí donde entra en juego LMP.

2.2.4.1. Etiqueta generalizada

La etiqueta generalizada es un concepto nuevo de GMPLS que viene a sustituir a la etiqueta MPLS tradicional en la señalización del protocolo. Esto permite la señalización en diferentes dominios.

Esta es la abstracción genérica de una etiqueta generalizada, aunque su implementación real es más compleja. Puede representar algo parecido a una etiqueta MPLS, en otro caso la propia longitud de onda asignada en una comunicación hará de etiqueta (etiquetamiento implícito), etcétera, para adaptarse a los distintos niveles de la comunicación y permitir a los dispositivos configurarse para conmutar y reenviar la información de la manera adecuada.

La etiqueta generalizada lleva codificada en sí misma tres valores:

- El tipo de codificación de la etiqueta que está siendo transportada.
- El tipo de conmutación que el dispositivo puede proporcionar.
- Un identificador que expresa el tipo de la carga útil que se va a transportar.

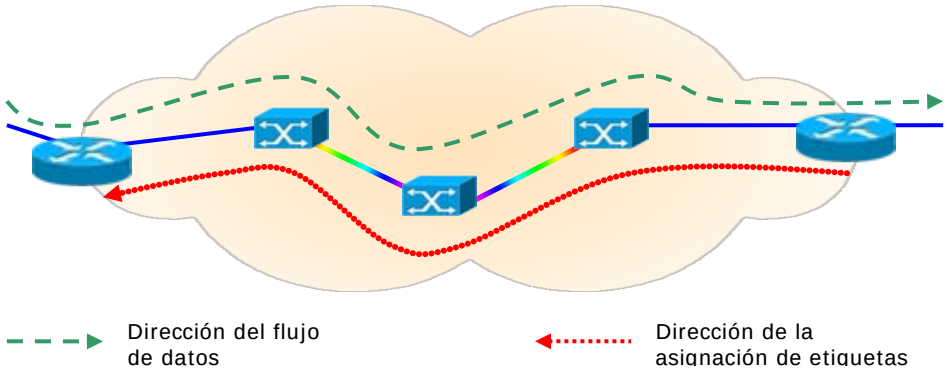
En la fecha de creación de este documento, la información al respecto es bastante escasa y la mayor parte de los trabajos del IETF y de aquellas empresas que están involucradas en el desarrollo de GMPLS, y concretamente de la señalización mediante etiquetas generalizadas, aún están en fase de desarrollo y todavía no fácilmente accesibles al público.

2.2.4.2. Distribución de etiquetas generalizadas

Las etiquetas generalizadas se distribuyen, generalmente, de forma similar como ocurre en MPLS, pero usando el protocolo LMP.

- Cuando un tráfico llega a un LER de entrada del dominio GMPLS, LMP indica al protocolo RSVP si hay o no una adyacencia para el reenvío de información con otro nodo (que existe un enlace), además de los recursos disponibles en dicho enlace y de diversa información para el control de la comunicación entre ambos.
- De las características requeridas para el LSP y notificadas por LMP, GMPLS indica a RSVP que inicie la señalización para obtener de esta forma un camino al destino; esto se hace salto a salto mediante mensajes PATHMSG.
- El proceso se repite hasta llegar al LER de salida del dominio, el nodo buscado; ahí, RSVP indica a LMP, que localice los recursos necesarios según los datos incluidos en la señalización (mensaje PATHMSG). En ese momento LMP localiza los recursos y, si es factible, lo notifica a RSVP que enviará hacia atrás un mensaje RESVMSG.
- El proceso se repite, de nuevo, hasta llegar al nodo origen de la señalización en cuyo caso, si todo ha ido correctamente, se tendrá el LSP establecido.

La única salvedad con MPLS es que ahora, no todos los nodos trabajan al mismo nivel ni todos los LSP son iguales: un tramo puede ser por conmutación de paquetes mientras que otro tramo puede ser por conmutación de fibras completas.



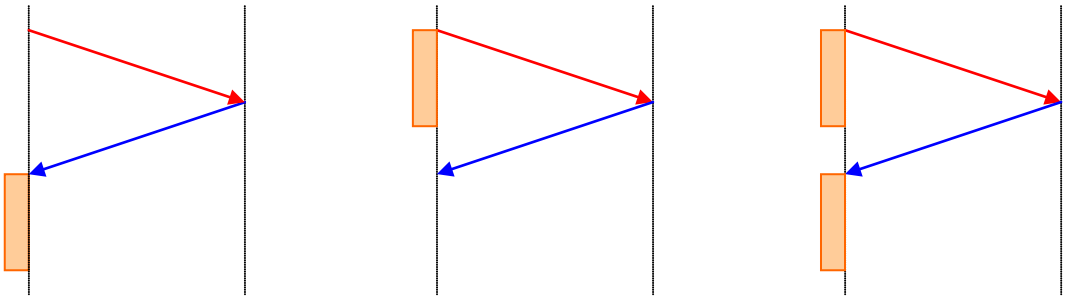
Distribución de etiquetas generalizadas

2.2.4.3. Sugerencia de etiquetas

Al igual que en MPLS, un nodo puede solicitar a su adyacente el establecimiento de un LSP mediante la sugerencia de una etiqueta. Esto es, en lugar de pedirle una etiqueta para mandar información, le anuncia qué etiqueta va a utilizar, directamente.

Este tipo de sugerencia que existe también en MPLS tradicional, tiene más importancia si cabe en GMPLS. En GMPLS, un dispositivo capaz de conmutar mazos de cientos de fibras, cada una de ellas con cientos de longitudes de ondas como portadoras y en cada una distintos flujos de información, puede tardar en autoconfigurarse para la conmutación un tiempo considerable y que no puede solapar con la petición de la etiqueta puesto que es la etiqueta generalizada, una vez concedida, la que contiene información necesaria dicho proceso de configuración

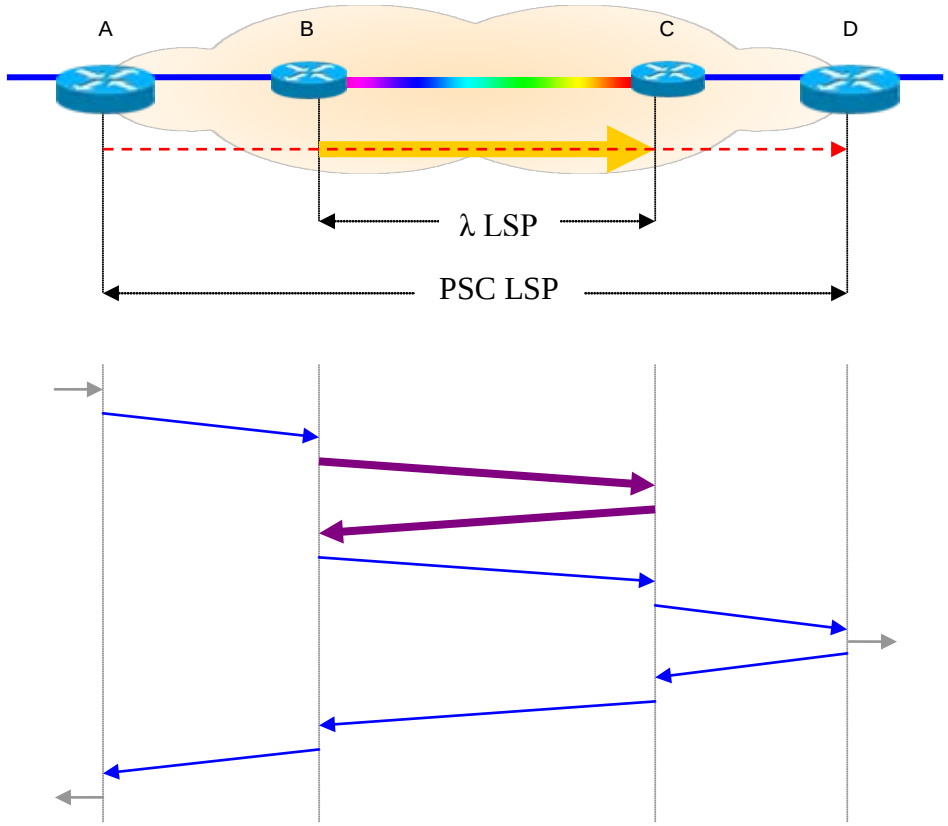
Por supuesto, el nodo al que llega la sugerencia tiene la capacidad de aceptar esa sugerencia o, en caso contrario, ofrecer otra etiqueta para el establecimiento del LSP.



A) Petición de etiqueta normal. Hasta que no llega la etiqueta concedida no se puede configurar el conmutador.
B) Sugerencia de etiqueta. El conmutador puede configurarse inmediatamente. Si luego llega una aceptación, ya podrá conmutar.
C) Sugerencia de etiqueta. El conmutador puede configurarse inmediatamente. Si luego llega una denegación, debe volver a configurarse. Trabajo hecho en balde.

2.2.5. Establecimiento de caminos conmutados

La creación de caminos conmutados prácticamente no difiere con respecto a MPLS, como se ha descrito en el apartado *"Distribución de etiquetas generalizadas"*. Como en GMPLS un LSP sólo puede ser establecido entre nodos con la misma capacidad de conmutación, un LSP se establecería como muestra la siguiente figura:



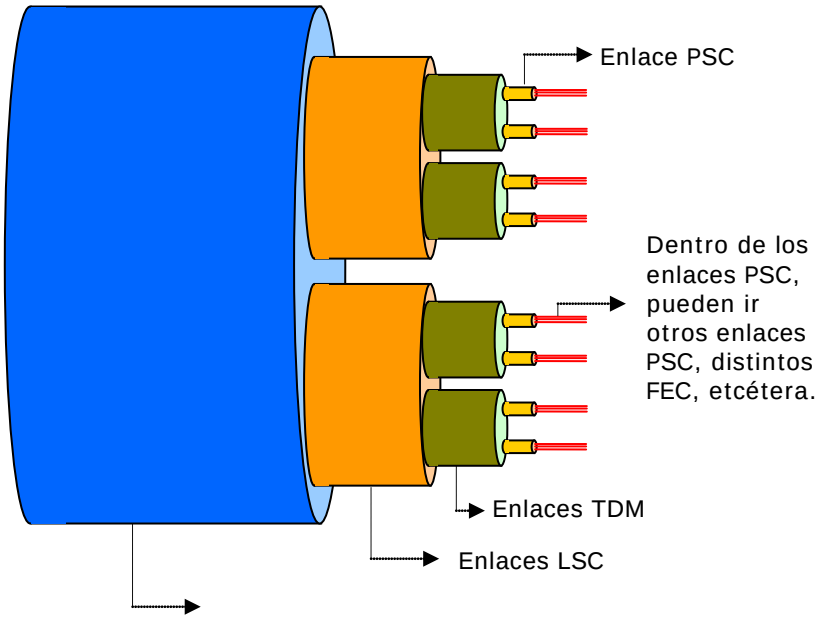
Creación de un LSP

Es decir, tal como en MPLS, el LSP se establecería generalmente mediante la asignación consecutiva de etiquetas desde el LER de salida del dominio GMPLS hacia el LER de entrada al mismo, tras las respectivas peticiones de etiquetas en sentido contrario. Sin embargo, cuando se llegara a una subred interna de más bajo nivel, una red óptica en el caso de la figura, en ella se establecería un LSP basado en landas, antes de proseguir, y luego se utilizaría este como túnel para proseguir la creación del LSP extremo a extremo entre dispositivos PSC. Esto se haría así con todas las redes de distintos niveles que se encontraran en el camino.

En el ejemplo, el LER A solicitaría al B una etiqueta para establecer un LSP con D. Antes de continuar, B establecerá un LSP de nivel inferior, basado en landas, con C y posteriormente se continuaría realizando la petición a C y D para el LSP solicitado por A. Una vez establecido el LSP, el tráfico circularía por un LSP desde A a B y desde C a D, pero de B a C circularía por dos, uno dentro del otro, en un túnel.

2.2.5.1. Jerarquía de LSP

El concepto anterior desemboca en lo que se denomina en GMPLS la jerarquía LSP.

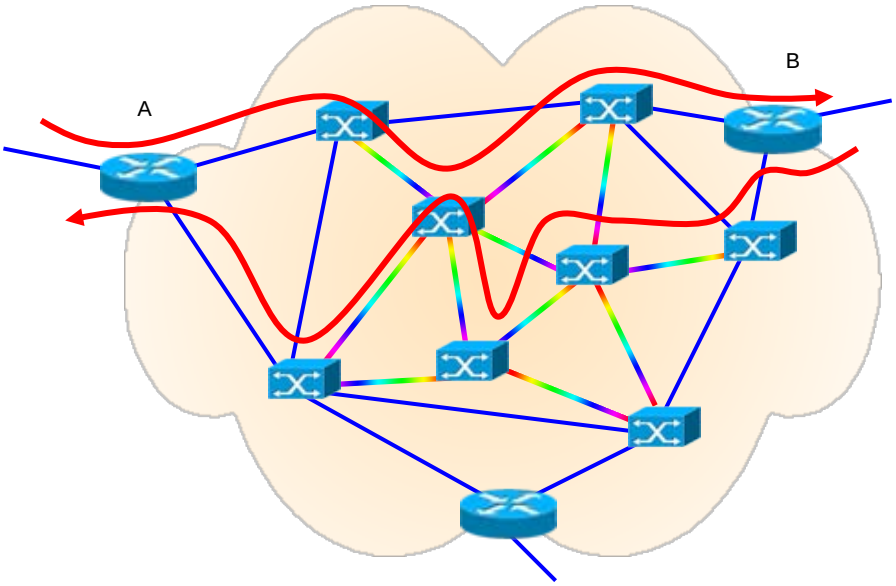


En MPLS los LSP podían albergar dentro de si distintos FEC, cada uno con distintos flujos, etcétera. En GMPLS esta jerarquía viene dada, de forma casi obligatoria, por el nivel en el que trabaje la red y por tanto, el tipo de LSP que se debe crear; redes de nivel más bajo tunelizan redes de nivel más alto y están, por tanto, en un nivel superior de la jerarquía. Asimismo, estos niveles superiores de la jerarquía, al ser conmutados, permiten la conmutación simultánea de muchísimos flujos de información; esta granularidad se torna más fina a medida que nos acercamos a los LSP interiores de la jerarquía.

Además, las redes más bajas de la jerarquía sólo tienen la posibilidad de asignación de anchos de banda discretos, que la mayor parte de las veces no se ajustan al ancho de banda requerido por un LSP de nivel superior, desaprovechándose los recursos; sin embargo introduciendo varios (muchos, generalmente) LSP de niveles superiores en uno de nivel inferior, los primeros suelen ocupar completamente el ancho de banda asignado, evitando este desperdicio. Esta característica de GMPLS le proporciona bastante escalabilidad.

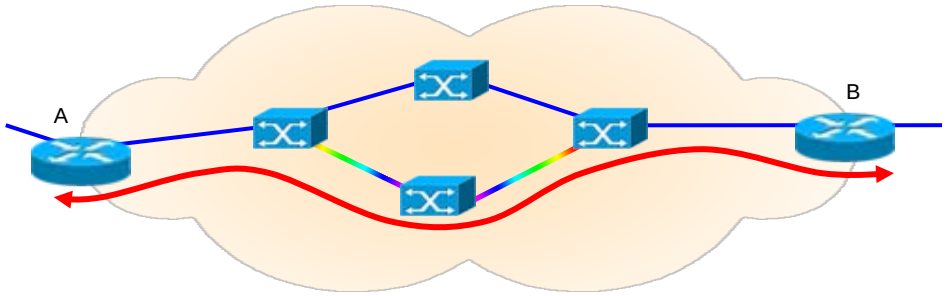
2.2.5.2. Creación de LSP bidireccionales

En MPLS, los LSP sólo podían establecerse de forma unidireccional, simplex. Sin embargo, la naturaleza del tráfico que van a transportar es, generalmente, full duplex. Eso significa que cada una de las partes se tiene que encargar de establecer un LSP con el otro extremo de la comunicación.



En la figura anterior vemos cómo sería este proceso: A crea un LSP hacia B, con unas determinadas características y pasando por unos determinados nodos. B hará lo mismo; creará un LSP hacia A, con sus propias características, distancias, etcétera. Esto supone que la latencia necesaria para el establecimiento del LSP "bidireccional" es la suma de la latencia de establecimiento de cada uno por separado; y además las características de ambos y las circunstancias que los rodean no tienen por qué ser las mismas.

GMPLS permite el establecimiento de LSP bidireccionales en una única operación.



LSP bidireccional en GMPLS

Esto lo consigue mediante mensajes de señalización y combinando además la sugerencia de etiquetas, por ejemplo mensajes PATHMSG mediante RSVP; aunque GMPLS, al igual que MPLS, no está sujeto únicamente a este protocolo, como ya sabemos. En la figura anterior se muestra el establecimiento de un LSP bidireccional en GMPLS. Con respecto a MPLS, la diferencia estriba en que es el mismo LSP el que establece el camino de ida y vuelta, por lo que el establecimiento se realiza más rápido, sólo es necesario configurar internamente el nodos una vez y además el camino final de ida y el camino final de vuelta (el LSP bidireccional, por tanto), discurren por los mismos nodos, con las mismas características, distancias, congestión, averías, etcétera.

2.2.5.3. Agrupamiento de enlaces

Los protocolos LMP junto con RSVP (binomio más utilizado) generan gran cantidad de datos internos en los nodos para mantener la información de todos los enlaces existentes o LSP que están establecidos. El tamaño de esta base de datos aumenta mucho cuando bajamos en la jerarquía de la red hacia redes ópticas o incluso en conmutación de fibras; esto es así porque estas redes hacen de autovía para transportar todos los LSP que atraviesan la red.

Para evitar el tener que gestionar cantidades de datos muy grandes en los nodos, GMPLS permite crear una agrupación de enlaces, que consiste en mantener una sola entrada en la bases de datos LMP interna de los nodos para todos aquellos LSP que compartan origen, destino y características (jitter, delay, requerimientos de ancho de banda…). Esta agregación de LSP en un único enlace se notifica a los protocolos de routing, como OSPF o IS-IS, que ya han sido modificados para este menester. Además, una agrupación de

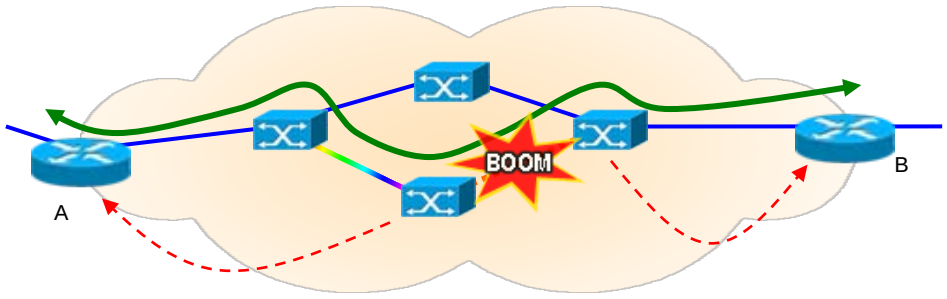
enlaces necesita mensajes de señalización como si se tratara de un solo enlace, en lugar de miles, lo cual es muy ventajoso.

Para que sea posible el agrupamiento, todos los enlaces que se desean agrupar deben cumplir:

- Comenzar y terminar en el mismo par de LSR.
- Ser del mismo tipo (Point-to-point, Multicast, Broadcast, etcétera).
- Tener las mismas características (tipo de protección, ancho de banda, etcétera).
- Tener el mismo tipo de capacidad de conmutación: PSC, TDMC, LSC o FSC.

2.2.5.4. Recuperación ante caídas de la red

En GMPLS se ha añadido la propiedad de enviar mensajes de notificación a nodos no adyacentes. Esto es importante porque ante fallos en la red un LSR debe ser capaz de notificar a los nodos encargados de reestablecer el LSP por otro cauce, de que la red se ha caído. Debe hacerse lo más rápido posible y de la forma más inteligente, evitando mensajes intermedios que retrasarían la puesta en marcha del proceso de reestructuración. Por ejemplo se puede notificar que un LSP ha fallado, el porqué ha fallado, si ha fallado el LSP de control, el de transporte o los dos, etcétera.



Notificación de la caída del LSP

Estos mensajes de notificación se envían vía RSVP-TE, gracias a una extensión que se ha añadido a este protocolo. En este caso, de momento, CR-LDP no cuenta con esta capacidad.

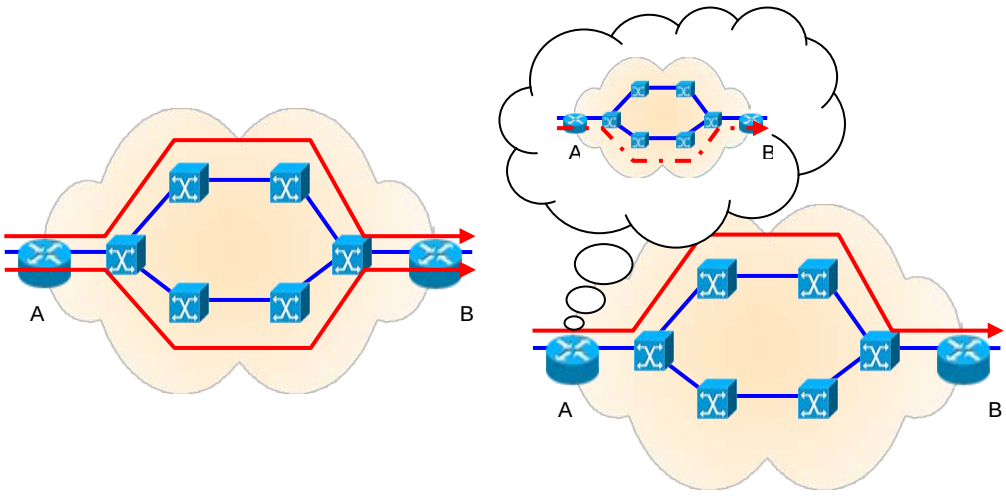
La gestión de caídas de la red se puede desglosar en cuatro puntos principales:

- Detección.
- Localización.
- Notificación.
- Mitigación.

La detección del fallo debe correr por cuenta de la capa que más cercana esté a lugar donde ha ocurrido el problema. Cada tipo de red tiene su método de detección de caída de la red; generalmente es cuando no se detecta portadora o se detecta con demasiada baja calidad, o bien el número de errores de transmisión es muy elevado.

La localización necesita que los nodos se comuniquen para detectar el enlace caído; en LMP se ha definido un procedimiento de localización mediante unos mensajes, denominados CHANNEL_FAIL, entre nodos adyacentes.

Cuando se ha notificado y localizado el lugar del fallo y todo es conocido por lo protocolos de señalización y control, entran en juego los métodos de protección y/o restauración.



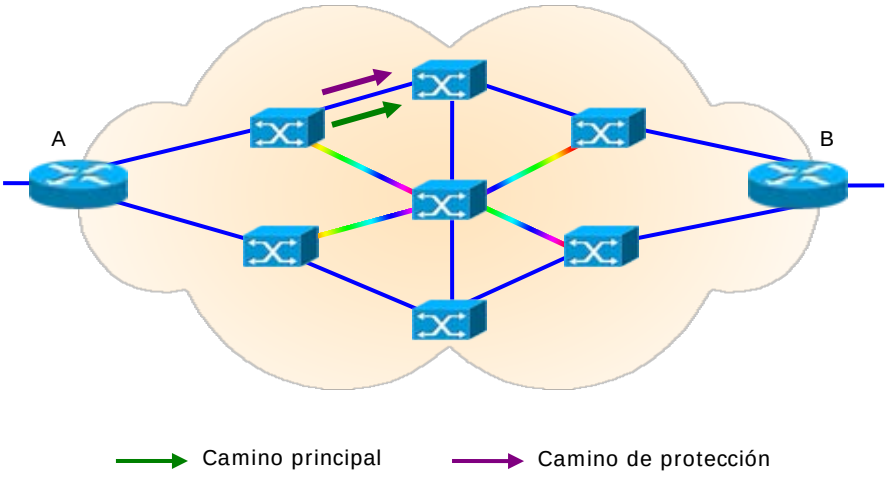
Mecanismo de protección (izquierda) y restauración (derecha)

En el primero de los casos, durante el establecimiento del LSP se habrá creado y establecido otro LSP de idénticas características, que estará consumiendo recursos aunque

no los use. Es un camino de backup. Este método permite que ante la caída de un enlace de la red el tiempo que se tarda en reestablecer la transmisión de información sea mínimo.

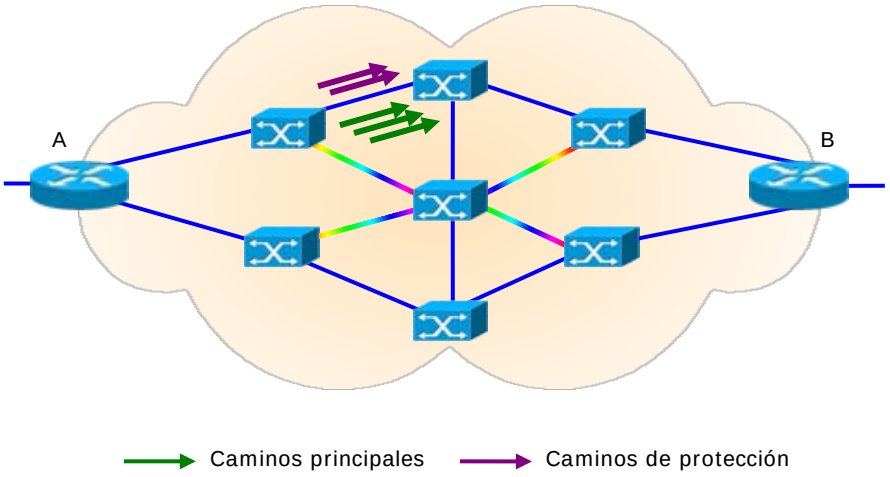
En el segundo de los casos, la restauración, el camino estará precalculado pero no estará establecido, por lo que en el momento de tener que hacer uso del camino de backup, hay que establecer el camino y puede ocurrir que no haya recursos disponibles para él. En el mejor de los casos será una operación de recuperación mucho más lenta que la propuesta de protección, pero en cambio no necesita tener ocupados recursos innecesariamente.

En GMPLS se han definido diferentes tipos de protección, que pueden ser señalizadas mediante RSVP-TE o CR-LDP. Son las siguientes:



Span protection 1+1

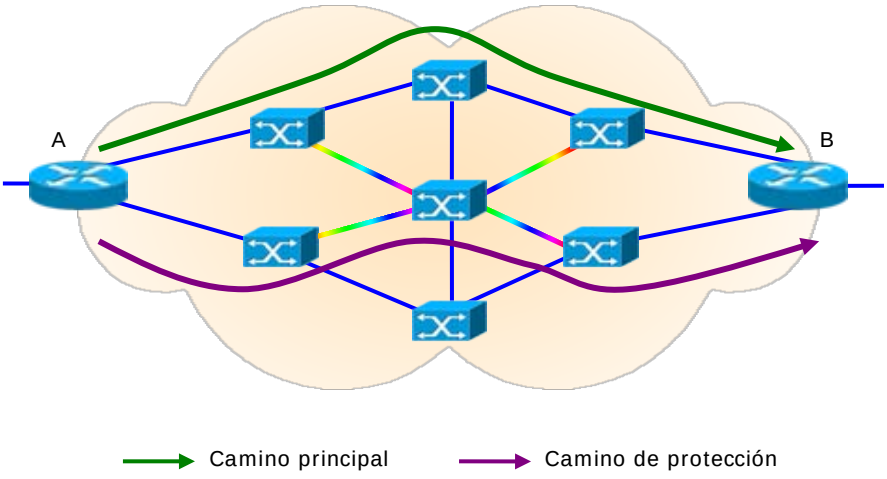
- **Protección 1+1:** en este tipo de protección, se duplica por completo el tramo de LSP entre dos nodos y el respectivo camino de control, con los mismos requisitos que sus homólogos principales y ocupando, por tanto, el doble de recursos que una comunicación sin protección.
- **Protección M:N:** en este tipo de protección, existen N canales principales de comunicación y para todos ellos existen M canales de reserva o backup, que deben ser compartidos. Como no es normal que todos los caminos principales fallen, no es en principio necesario que cada uno tenga uno de reserva, sino que se establecen algunos a compartir entre todos los principales.



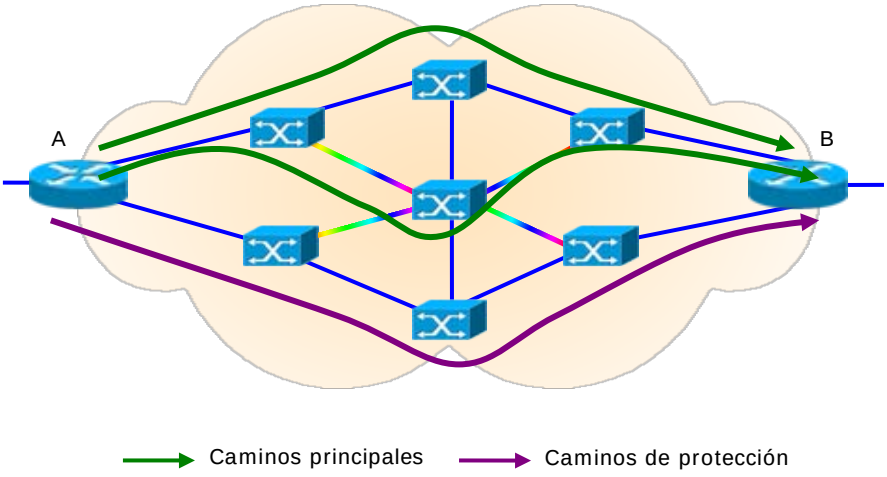
Span protection M:N

Los dos modelos anteriores se conocen como *span protection* y tienen lugar entre dos nodos adyacentes; por tanto, punto a punto. Los otros dos modelos, que completan el conjunto de cuatro posibilidades que habíamos comentado, se denominan *path protection* y tienen lugar extremo a extremo

- **Protección 1+1:** en este tipo de protección, se duplica por completo el LSP desde el origen hasta el destino, por caminos distintos y también el respectivo camino de control, con los mismos requisitos que sus homólogos principales y ocupando, por tanto, el doble de recursos que una comunicación sin protección.
- **Protección M:N:** en este tipo de protección, existen N canales principales de comunicación y para todos ellos existen M canales de reserva o backup, que deben ser compartidos. Como no es normal que todos los caminos principales fallen, no es en principio necesario que cada uno tenga uno de reserva, sino que se establecen algunos a compartir entre todos los principales.

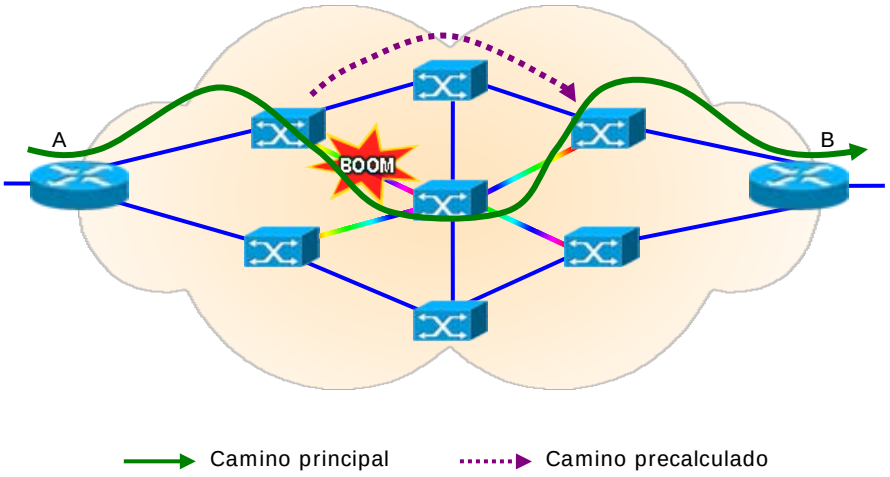


Path protection 1+1



Path protection M:N

Con respecto a la restauración, se suele utilizar únicamente en nodos intermedios de la red. Se precalcula un camino y se utiliza para rodear un fallo en la red en un tramo concreto, algo así como un *by-pass*. La técnica consiste en calcular dos formas de llegar desde un nodo intermedio a otro. Si esta técnica se aplica a un LSP completo extremo a extremo el tiempo necesario para señalizar toda la operación, reencaminar el tráfico y reestablecer un LSP sobrepasa lo aceptable.



Ejemplo de restauración

2.2.5.5. Enlaces sin numerar

Generalmente en MPLS y GMPLS cada enlace de la topología, y no sólo cada nodo, tienen direcciones IP asignadas. Se eligió este método debido a la unicidad de las direcciones IP. Es necesario para que los protocolos de señalización (para ingeniería de tráfico) sean capaces de identificar inequívocamente un enlace concreto. Pero el potencial uso de direcciones IP en enlaces TDM u ópticos es otro de los obstáculos de GMPLS. Para ello, en lugar de asignar una dirección IP diferente para cada enlace TDM u óptico, utilizamos el concepto de “enlace no numerado”. Es necesario por lo siguiente:

- El número de canales TDM, longitudes de onda distintas y fibras puede alcanzar fácilmente un punto donde su gestión, mediante direcciones IP, puede consumir mucho tiempo de proceso.
- Las direcciones IP son un bien escaso.

Un enlace no numerado es un enlace TDM u óptico al que no está identificado por una dirección IP sino que en su lugar se le ha asignado el identificador interno y único del encaminador y un número, único para ese encaminador, que representa al enlace. Este hecho puede ser señalizado mediante protocolos como RSVP-TE o CR-LDP, que han sido extendidos para ello, al igual que OSPF-TE o IS-IS-TE.

2.2.6. Ventajas de GMPLS con respecto a otras tecnologías

Se podría entender que más que multiprotocolo, GMPLS es multicapa. En realidad casi todas las innovaciones que incorpora GMPLS con respecto a MPLS derivan de que permite operar de forma transparente en distintos dominios de conmutación, hasta llegar a los niveles más bajos del modelo de referencia OSI. Así que la ventaja principal es precisamente esta, que una misma tecnología puede ser aplicada en distintos ámbitos, con distintas características.

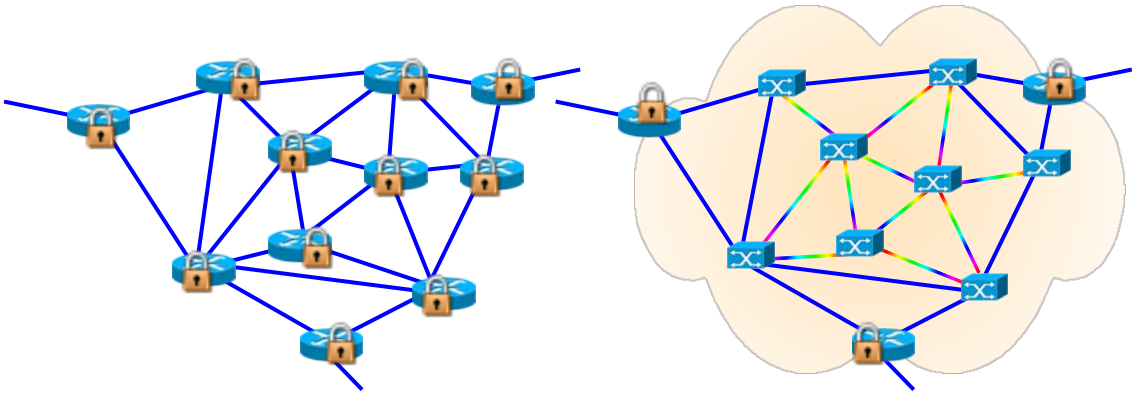
2.2.7. Problemática en GMPLS

Aunque GMPLS es un refinamiento que parte de las problemáticas encontradas en MPLS, de momento también tiene sus propios problemas, que se resumen a continuación.

2.2.7.1. Control en el acceso a la red

En las redes tradicionales, como las redes IP en Internet, el contenido de la cabecera de cada paquete se utiliza para encaminar, salto a salto, dicho paquete. Este proceso consume mucho tiempo y es una de las técnicas que MPLS y GMPLS han evitado desde el comienzo, pero tiene sus ventajas: las direcciones IP tienen significado global y pueden ser usado, en cada decisión de encaminamiento, para establecer mecanismo de seguridad como cortafuegos o algún control de acceso a la red. En GMPLS, como en MPLS, una vez establecido el LSP hacia el destino, todo lo que se hace es conmutar rápidamente basándose en las etiquetas concedidas; pero estas etiquetas tienen significado local al nodo y no pueden ser utilizadas como en IP para cortafuegos, políticas de seguridad, etcétera, ya que los nodos MPLS no ven más allá de la etiqueta MPLS/GMPLS de los paquetes.

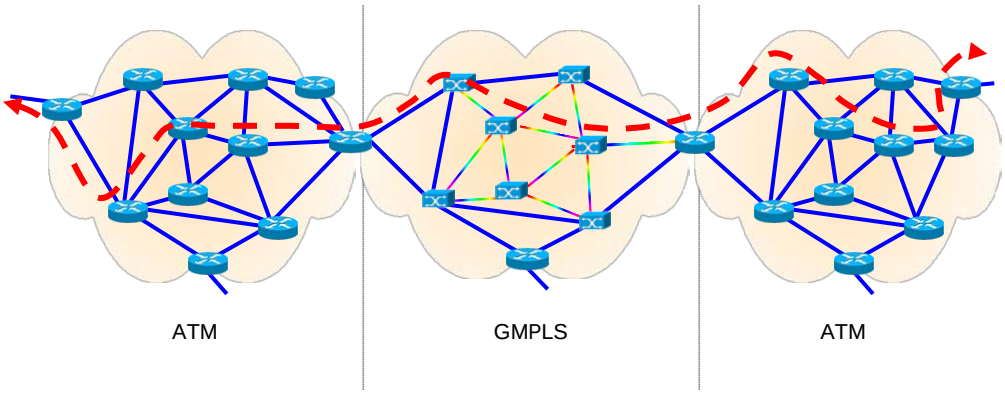
Por tanto, hay que establecer mecanismos de control fuertes en el momento de crear el LSP hacia el destino, pues los LER de entrada a los dominios MPLS/GMPLS si pueden y deben examinar la cabecera de los paquetes no etiquetados.



Lugares con posibilidad de control en redes IP y GMPLS

2.2.7.2. Interred

Gran parte del éxito de GMPLS dependerá de la habilidad que tenga para hacer de nexo de unión entre el tumulto de redes diferentes que actualmente existen. En concreto, debe permitir por ejemplo conectar dos redes del mismo tipo, como ATM a través de una red GMPLS y, por supuesto, conservando los requerimientos de las conexiones establecidas vía ATM.



Interred entre redes ATM usando una red GMPLS

Pero este tipo de técnicas tienen en GMPLS algunos escollos que no están de momento claramente resueltos:

- El funcionamiento entre redes es muy complicado en el plano de control debido a que se usan muchos y muy variados protocolos distintos en cada una de las redes.
- El mantenimiento de los parámetros de calidad de servicio requeridos extremo a extremo vía una interred por las redes ATM resulta muy difícil establecer en las redes GMPLS que intentan interconectarlas.
- Otro problema es que en el plano de datos, las redes GMPLS permiten muchas más combinaciones en el tipo de conmutación utilizada que Frame Relay o ATM, por ejemplo. A mayor heterogeneidad, mayor complejidad en la implementación de la red y la aseguración del cumplimiento de los requisitos exigidos.

2.2.7.3. Estabilización de la red

En una red IP tradicional, el volumen de información de control y señalización que se intercambia en la red es muy pequeño comparado con la cantidad intercambiada en redes GMPLS, que afecta a muchos protocolos de control, señalización, etcétera. Esta diferencia es esencial para el reestablecimiento del funcionamiento normal de la red cuando algún nodo o enlace de la misma cae. Teóricamente, las redes GMPLS tardan un tiempo sensiblemente mayor en reestructurarse a nivel de señalización y control que redes IP.

2.2.7.4. Sistemas de gestión de red

Las redes IP tradicionales son gestionadas mayormente basándose en la accesibilidad de cada nodo por su dirección IP; no son muchos los parámetros a tener en cuenta en comparación con las redes GMPLS donde se debe llevar un seguimiento de cientos de miles de LSP, tablas para el encaminamiento, las actuaciones en cuanto a ingeniería de tráfico, etcétera. En definitiva, el volumen de información generado por una red GMPLS es muchísimo mayor que el que podría generar nunca una red IP y los factores a tener en cuenta también son muchos. Todo esto hace que la creación de sistemas de gestión en redes GMPLS no sea una tarea fácil.

2.2.8. Tendencias

Cada vez se busca con más ahínco una fórmula mágica que permita tener redes heterogéneas, las que están actualmente implantadas, y que sin embargo permita ofrecer servicios avanzados a los usuarios sin tener que invertir demasiado dinero en un cambio de infraestructuras. En fin, lo que siempre se ha buscado.

GMPLS está desde su origen pensado para ser compatible con las tecnologías existentes a la vez que con los requerimientos futuros, al igual que MPLS. Además se debe entender GMPLS como una tecnología que parte de este último intentando subsanar los errores que rápidamente han aparecido en MPLS y hacerlo resurgiendo de nuevo, no constituyéndose en un mal parche.

Con la llegada inminente de las tecnologías ópticas de manera generalizada, las protocolos existentes se quedaban atrás y GMPLS viene a cubrir ese hueco, de forma progresiva, a la vez que plantea la necesidad de una organización, una jerarquía en las redes, que permita manejar el tráfico que fluye, de una forma eficiente, rápida, económica y sin demasiados costes de implantación; a la vez permite ofrecer a los posibles clientes servicios más avanzados.

2.2.9. Conclusiones

Muchas de las carencias que podemos encontrar actualmente en MPLS están siendo subsanadas desde el origen en GMPLS; sin embargo este último es aún tan inmaduro que pocos fabricantes de electrónica de red disponen de equipamiento, salvo de un modo experimental para poder desplegar una infraestructura GMPLS que entre en producción. Se prevé que MPLS y GPLS continúen su desarrollo de forma paralela al igual que ocurrió con ATM e IP hasta que en el futuro las redes vayan incorporando conmutadores y encaminadores GMPLS en lugar de MPLS.

Sin embargo, de momento, es interesante buscar mecanismos que subsanen las carencias de MPLS respetando el estándar, para acercarlo a GMPLS al menos en aquellas áreas o en aquellas situaciones en que sea necesario, ya que parece que la tecnología GMPLS tardará aún un tiempo en estar lo suficientemente madura para ser implantada de forma masiva, pues quedan muchos aspectos por resolver, investigar y unificar, porque se están

modificando infinidad de protocolos de forma paralela para que GMPLS pueda ser tan genérico como se pretende.

2.3. Redes activas y sistemas multiagente

Los investigadores tienen una gran limitación para poder investigar sobre las redes porque el ciclo de vida y evolución de las redes tiene un ritmo de cambio mucho más lento que, por ejemplo, el software, pese a estar íntimamente ligado a esta. El despliegue una infraestructura de red, la estandarización del funcionamiento en interred, la necesidad de acuerdos globales entre fabricantes (desde el auge importante de Internet, las redes ya no son propietarias), etcétera, hacen que poder desplegar redes de prueba sea muy caro y complicado; Innovar en los protocolos puede ser factible, pero la adopción de dichas innovaciones, en tiempo, dinero y trabajo son en muchas ocasiones, inviables. De hecho, muchos buenos proyectos de investigación en protocolos han quedado en el cajón porque, pese a ser útiles, su implantación es compleja e implica a muchos agentes. Las redes activas y los sistemas multiagente vienen a solventar en cierta medida este problema.

2.3.1. Redes activas

Una red activa es aquella en la que sus nodos tienen una componente de configuración dinámica en tiempo real que permite a sus usuarios configurar el comportamiento de la red para adaptarlo a sus necesidades.

Las redes tradicionales de datos aplican sobre los paquetes que circulan por ella un tratamiento uniforme; es decir, dado un paquete con una dirección de origen y una de destino y un puerto de entrada, los nodos calculan el puerto de salida. Siguiendo este paradigma de forma repetitiva, los paquetes son guiados desde el origen al destino en un proceso que se llama encaminamiento.

El primer problema que tiene este enfoque es ¿Qué ocurre si los protocolos de comunicaciones o el formato de la trama cambian? ¿Cómo responde la red entonces? Sencillamente no puede responder. La estructura e un nodo tradicional es estática; esto es:

está creado para un fin y no se puede cambiar. Aunque algunos dispositivos de red permiten la actualización del *firmware* para solucionar problemas o ampliar algún detalle, esto no es una actualización dinámica ni flexible que modifique por completo la funcionalidad del nodo.

Como muy bien está explicado por Pedro Andrés Aranda Gutiérrez (Telefónica I+D), para dar solución a estos problemas, se comenzó a investigar sobre la posibilidad de que los paquetes de datos llevaran adjunta cierta información sobre cómo deben ser tratados por parte de los nodos. De este modo, el funcionamiento del nodo, un nodo activo, sería dinámico; no siempre se comportaría igual dependiendo del tráfico que lo atravesase. Este es el concepto inicial de una red activa.

Existen dos modelos principales de redes activas: las basadas en cápsulas y las basadas en el protocolo de redes activas ANEP *(Active Networking Encapsulation Protocol).*

2.3.1.1. Redes activas basadas en cápsulas

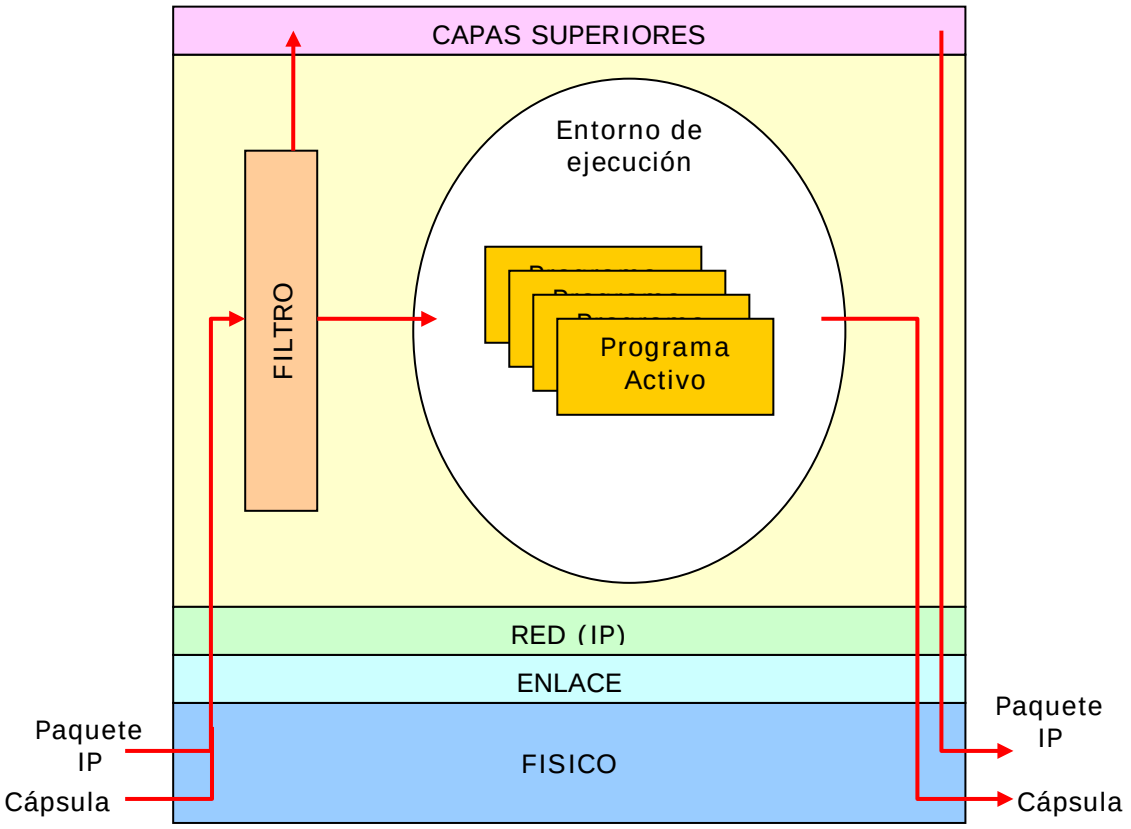
En las redes activas basadas en cápsulas, la propuesta es que cada paquete lleve incorporado no sólo la información necesaria para que el nodo activo sea capaza de tratarlo sino el propio código que el nodo debe ejecutar para tal fin. El nodo transfiere los paquetes hacia un espacio aislado, conocido como entorno de ejecución, de los que existen muchas variantes, en el cual se extrae y ejecuta el código que contienen. Este espacio o entorno de ejecución define las operaciones que se pueden realizar sobre el paquete y los recursos del nodo que tiene disponibles, y es el encargado de garantizar los aspectos de seguridad de la red, entre los que destacan:

- La comprobación del origen del código, denegando el acceso a la red o al menos la generación de cápsulas a todo aquel no esté autorizado para ello. Esto además de ser soportado por la red, obliga a modificar la política de acceso a la red en aquellos lugares donde se desee implantar una red activa.
- La comprobación de la integridad del código "líquido" introducido en la red a través de cápsulas, comprobando que el código que llega al nodo no ha sido

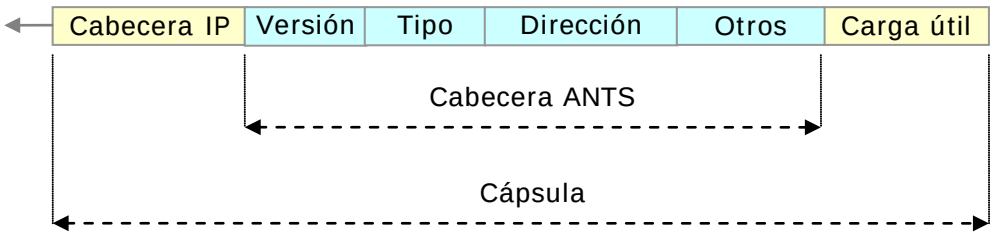
modificado por terceros antes de llegar al nodo activo, lo que podría provocar que se ejecutase código malicioso en el nodo activo.

- La definición de una interfaz de programación segura, que no permita a ningún código activo hacer un uso monopolístico de los recursos del nodo. Esto define una API de red, que es el conjunto de interfaces ofrecida por la red para programar su funcionamiento.

La estructura típica de nodos activos para este enfoque de redes activas (suponiendo redes IP) es, con un gran nivel de abstracción, como se muestra en la figura:



ANTS *(Active Network Transfer system)* es una de las propuestas más asentadas para redes activas basadas en cápsulas y define cómo debe estar formada la cápsula de datos-código. Su aspecto es como se muestra en la siguiente figura:

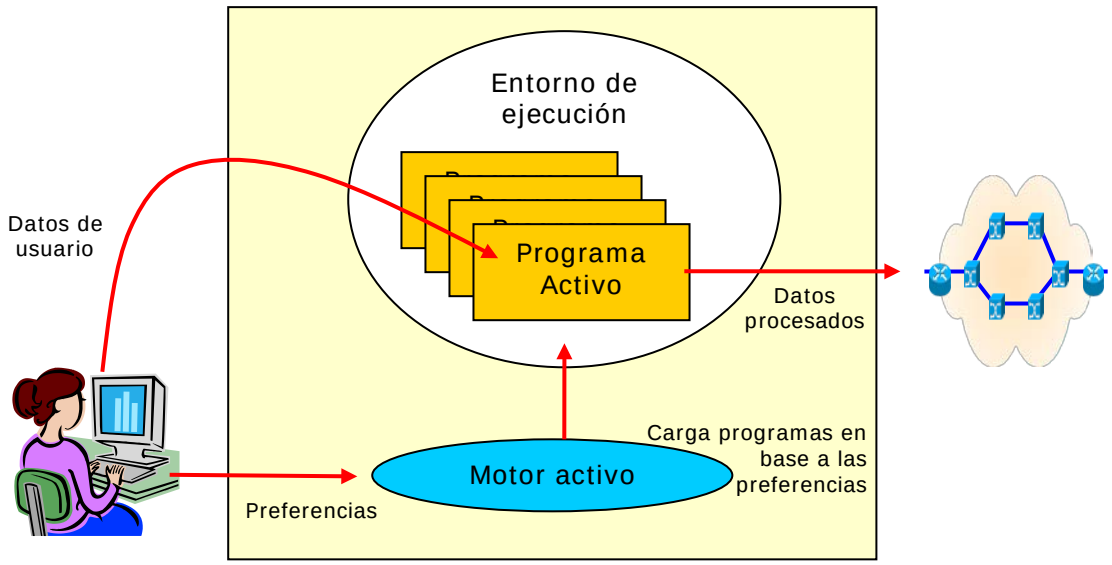


Por tanto, la cabecera ANTS que es la que realmente forma la cápsula se intercala entre la cabecera de red y la cabecera de transporte. Tiene diversos campos que permiten a un nodo activo tratarla de forma concreta:

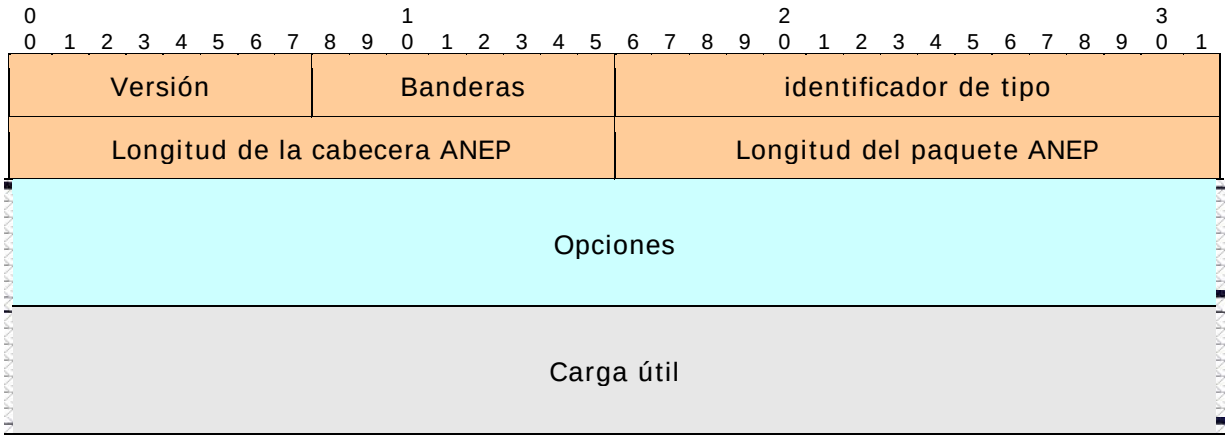
- **Versión:** indica la versión de ANTS utilizada.
- **Tipo:** es un identificador que indica el protocolo y el método de reenvío que necesita la cápsula. Está basado en una función HASH que entre los valores de protocolo, el método de reenvío y el código que lleva incorporado la cápsula, conforma una huella dactilar o *fingerprint* que impide el *spoofing* del código en el tránsito de la cápsula.
- **Dirección:** dirección usada por el sistema de distribución de código de ANTS.
- **Otros:** campos opcionales y variables que dependen del valor especificado en Tipo.

2.3.1.2. Redes activas basadas en ANEP

Este es el segundo enfoque de las redes activas en este momento. Se basa en trasladar las características de red activa de un nodo a los niveles que están por encima del nivel de red. Aquí, los programas y los datos van separados, lo que implica que no existe el concepto de cápsula. En este enfoque, los nodos activos reciben peticiones por parte de aplicaciones clientes solicitando unas ciertas características activas. El nodo activo se limita a cargar o descargar los programas necesarios para dotar de esas características activas al nodo y a partir de ahí, gracias a que programas activos están cargados, el usuario tiene la opción de hacerle llegar al nodo activo lo datos que tendrá que procesar.



Este modelo se basa en el protocolo ANEP, que define el aspecto que deben de tener los paquetes activos en una red de este tipo; como sigue:



Donde la ordenación de los paquetes en la cabecera es *Big-endian* y los campos tienen el significado que se muestra a continuación.

- **Versión:** indica la versión del protocolo ANEP usada. Esto determina también el formato de la cabecera ANEP que debe suponer el nodo activo. De momento sólo debe terne el valor 1, pues es la única versión de ANEP existente.
- **Banderas:** este campo almacena valores binarios *(flags)* para distintas opciones. De momento sólo se usa el bit más significativo. Indica qué debe hacer el nodo activo si no entiende el identificador de tipo especificado. 0 indica que intente

reenviar el paquete como tradicionalmente lo haría. 1 indica que lo descarte. Los demás bits se irán usando a medida que madure el protocolo ANEP.

- **Identificador de tipo:** este valor especifica la operación que debe realizar el nodo activo sobre el paquete, cómo debe procesarlo, los requerimientos que tiene, etcétera. Debe ser un valor especificado por la ANANA *(Active Network Assigned Number Authority)*. El valor 0 está reservado.
- **Longitud cabecera ANEP:** especifica el tamaño total de la cabecera ANEP en palabras de 32 bits.
- **Longitud paquete ANEP:** este campo especifica el tamaño total del paquete ANEP, que incluye la cabecera ANEP y toda la redundancia y datos de niveles superiores, o sea, la cabecera ANEP y la carga útil. Está especificado en octetos (8 bits).
- **Opciones:** pueden especificarse diversas opciones mientras el tamaño de la cabecera lo permita y son valores de 32 bits que siguen un formato TLV y que permiten especificar distintas opciones como reglas de seguridad y otras.

2.3.1.3. Entornos de ejecución

En ambos modelos existe el concepto de entornos de ejecución. Como ya comentamos, un entorno de ejecución es una zona de aislamiento que permite a un programa ejecutarse de forma segura, sin interferir con el resto. Es un concepto parecido al de máquina virtual o al de contenedor de componentes. Este concepto es crítico para el buen rendimiento y el éxito de una red activa por lo que se están investigando mucho y en los últimos años han aparecido distintas propuestas de entornos de ejecución. Algunas de ellas, brevemente explicadas, son las siguientes:

- **Smartpackets:** en esta propuesta, el código móvil es introducido y distribuido en paquetes activos IP de datos y permanece en los nodos activos tanto tiempo como sea necesario para ser ejecutado; luego es descargado. Tiene como requisito que los paquetes activos deben caber completamente en una trama de nivel de enlace por lo que los diseñadores, basándose en que existen muchas redes ethernet, han delimitado que el tamaño de los programas activos debe ser de 1500 octetos. Utilizan un lenguaje de su creación llamado Sprocket que una vez compilado crea

un código denso binario llamado Spanner, que es el que interpreta el nodo activo y es por tanto el código que portan los paquetes activos.

- **Active Network Transport System (ANTS):** ANTS, propuesta creada por el MIT, permite programas escritos en java y compilados a *bytecodes*. En los nodos activos el entorno de ejecución es una máquina virtual java extendida con algunas funciones propias de ANTS, que permiten que los paquetes activos (cápsulas) sean decodificados e interpretados correctamente. Distribuye el código mediante un enfoque de código bajo demanda e incorpora mecanismos de huella digital para comprobar la autenticidad e integridad del código inyectado.
- **Liquid Software:** comparte características de otros prototipos de entornos de ejecución, pero fija su atención en la lentitud con la que trabaja Java por ser un lenguaje interpretado. Su mayor logro consiste en acelerar el funcionamiento del entorno de ejecución transformando para ello el código Java a código C, con lo que evitan la interpretación del código. Además, el proceso de compilado se realiza con compiladores específicos para este contexto.
- **SwitchWare:** es una propuesta de una Unviersidad de Pensilvania. Permite código embebido en los paquetes, creados mediante PLAN *(Programming Language for Active Networks)* que es un lenguaje funcional de propósito específico. Los programas escritos en PLAN no pueden violar la política de seguridad de los nodos activos; se pretende que sea tan seguro que no haga falta autenticación para permitir la ejecución de código en los nodos activos.
- **NetScript:** es una propuesta creada por la Universidad de Columbia. Se basa en la utilización un conjunto de herramientas llamadas Netscript que son pequeñas piezas en base a las cuales se pueden construir servicios más complejos. En el entorno de ejecución se utilizan intérpretes capaces de ejecutar estas primitivas de Netscript. El código que se transporta en los paquetes activos es Java y los flujos de datos son abstraídos y mapeados en clases del lenguaje de la taza.
- **Liane:** es una creación de Georgia Tech. Su finalidad es construir dinámicamente servicios con garantías de seguridad utilizando para ello servicios básicos fiables. No está ligado a ningún lenguaje en particular, pero la implementación del prototipo se ha realizado en C++; en cualquier caso, Liane se apoya en un modelo de programación reducido, que permite predecir cómo de flexible puede ser el

entorno dinámico y así los programadores necesitan invertir menos tiempos en hacer análisis de seguridad.

2.3.1.4. Sistema operativo del nodo: NodeOS.

El sistema operativo del nodo, llamado *NodeOS*, es otro de los componentes de un nodo activo. Su función es, como la de los sistemas operativos de PC, hacer de nexo entre los programas (en este caso, los entornos de ejecución) y los recursos físicos del propio nodo necesarios para la transmisión, el procesado y el almacenamiento de los paquetes activos. Además, es el propio *NodeOS* el que aísla a un entorno de ejecución de otro. Realmente no es una técnica moderna, de hecho es la misma que utilizan los sistemas operativos multiusuario para proporcionar acceso a los recursos y aislar a un usuario de otro.

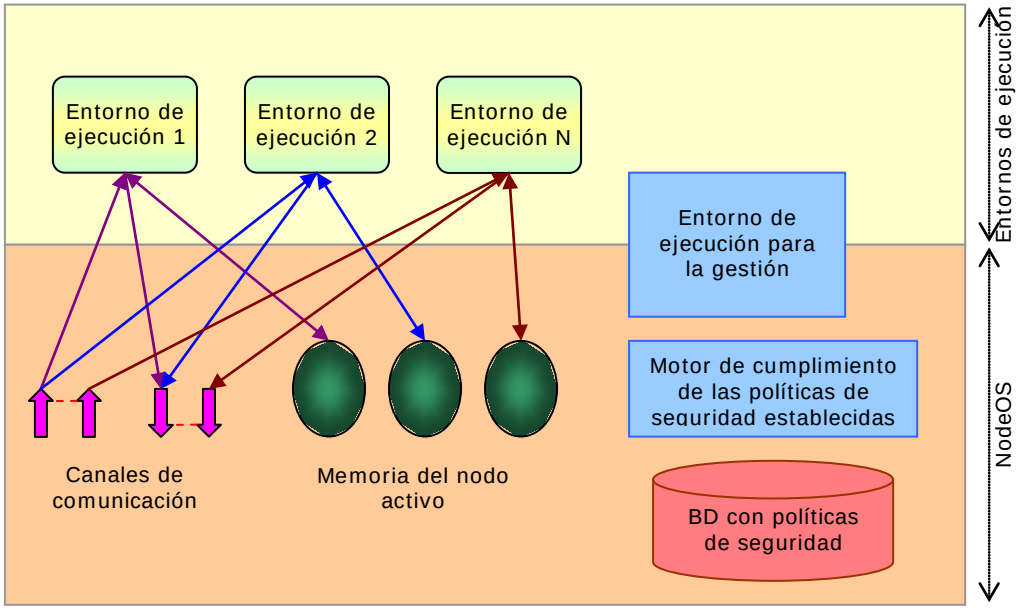
Aunque *NodeOS* se implementó inicialmente en el sistema operativo *Scout*, en la actualidad se están realizando muchas investigaciones sobre el sistema operativo Linux y Java como entorno de ejecución. Por su parte, Linux ofrece seguridad, estabilidad, buena gestión de recursos y, gracias a los núcleos de la serie 2.4, permite capturar de forma sencilla el tráfico de red y pasarlo del entorno del núcleo al entorno de usuario, donde podría fácilmente ser tratado por un programa Java en un entorno de ejecución JVM. Por su parte, Java es uno de los lenguajes preferidos para el desarrollo de nodos activos por su portabilidad y robustez.

En cualquier caso, el sistema operativo que se utilice en el nodo debe proporcionar facilidades para:

- Tratar el acceso a memoria y manejar distintos hilos de ejecución.
- Establecer canales de comunicación para cada flujo de información a conmutar.
- Permitir contabilidad del tráfico que atraviesa el nodo.
- Establecimiento de políticas de seguridad. Restringir qué y qué no puede hacer un programa activo.
- Establecer niveles de autorización para acceso a distintos aspectos del nodo o sus entornos de ejecución.

2.3.1.5. Arquitectura general de una red activa

Cualquier red activa suele seguir la estructura genérica que se muestra en la siguiente figura:



Donde cada uno de los entornos de ejecución existentes en el nodo activo, exporta una API de red que puede ser explotada por los usuarios mediante el envío de paquetes activos a través de los canales adecuados. Esto provocará que el nodo se reconfigure de una u otra manera y los paquetes sean procesador por un programa activo concreto dentro del entorno de ejecución. Todo este proceso puede desembocar en un cambio en el estado interno del nodo, en su memoria; por ello debe existir un motor de seguridad implementado en el sistema operativo del nodo que compruebe constantemente que la política de seguridad está cumpliéndose. Para ello, se utiliza un entorno de ejecución exclusivo para labores de gestión y control el nodo activo que funciona entre los espacios de usuario y del sistema operativo.

Hay que tener presente que la flexibilidad que ofrece un nodo programable, un nodo activo, influye en que las medidas de seguridad deban ser extremas; si no, código malicioso podría afectar al funcionamiento general del nodo.

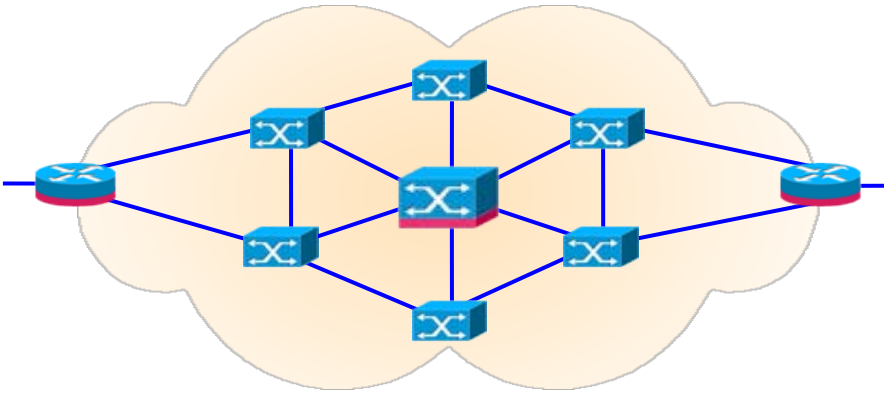
2.3.1.6. Ventajas de las redes activas. Aplicaciones.

Las ventajas principales que aportan las redes activas, se sitúan en torno a los niveles de red y transporte principalmente. Aun siendo gran parte software, las redes activas no se apartan demasiado de la idea de un nodo hardware. Resuelven problemas como encaminamiento, gestión de red, adaptabilidad de la red, soporte inmediato de nuevos protocolos en los nodos activos, control de congestión, etcétera. Por supuesto, otra de las ventajas de las redes activas es la facilidad con la que los investigadores pueden indagar y proponer nuevas ideas, probar nuevos servicios, protocolos, etcétera, en una red en producción. Un par de ejemplos de ellas se explican a continuación.

Adaptación dinámica: La calidad del servicio obtenido por una aplicación se puede degradar debido a las condiciones cambiantes de una red de comunicaciones. La solución tradicional ha sido que el emisor se vaya ajustando a la capacidad de la red, pero esto conlleva un retardo del tiempo que tarda la fuente en enterarse y posteriormente reaccionar. Con nodos activos esta adaptación se puede producir antes gracias a la capacidad del nodo para detectar el estado de la red.

Gestión de red: tradicionalmente la gestión, vía SNMP se hace mediante la consulta de ciertas variables de los nodos. Se hace desde una misma estación de gestión, central. Esto produce cuellos de botella. Los nodos activos, se pueden programar para que ellos mismos monitoricen su estado de forma distribuida y se puedan convertir, en si mismos, en estaciones de gestión. Así la monitorización de la red es distribuida.

Además de todo lo anterior, las redes activas tienen el añadido de que se pueden ir introduciendo paulatinamente en una red sin necesidad de cambiar toda la infraestructura de la misma.



Aspecto de una red activa, con tres nodos activos

2.3.1.7. Problemática de las redes activas

Hemos visto que las redes activas aportan sus beneficios en los niveles de red y de transporte de la red. Sin embargo, no está claro cómo una red activa podría actuar en redes que conmuten vía landas o bien mediante fibras completas. Por ejemplo, redes MPLS o GMPLS trabajan por debajo del nivel de red y no parece sencillo adaptar las redes activas a esos niveles.

Otro problema de las redes activas es la definición de una arquitectura de seguridad. En la mayor parte de los modelos que hemos tratado, los usuarios deben ser autenticados y tener permisos necesarios para ejecutar código en un nodo activo y para acceder a distintas partes de él. El problema es que extender este paradigma a una red como Internet significa tener autenticado a millones de usuarios potenciales, en un modelo de seguridad de escala global. Este es un punto que hay que resolver si se desean implementar redes activas con la seguridad de que código malicioso no puede ser ejecutado, al menos, sin que se sepa quién lo ha hecho.

Por otro lado, el rendimiento de las redes activas es un grave problema. Hay dos claras tendencias en los modelos que hemos comentado en las páginas anteriores. ANTS y otros, siguen la línea de ofrecer mucha flexibilidad en los nodos, proporcionar muchas API y que todo se pueda hacer. Esto permite nodos muy abiertos donde se pueden implantar servicios de todo tipo y los investigadores tienen un gran campo abierto. Sin embargo su rendimiento cae. Por otro lado, por ejemplo *Switchware* sigue la línea de ofrecer poca

flexibilidad, permitir un conjunto reducido de aplicaciones, pero proporcionar un buen rendimiento. En realidad lo deseable sería un nodo que permita mucha flexibilidad y que además obtenga un buen rendimiento. Será una de las cosas que tendrán que resolver los investigadores en los próximos años.

2.3.2. Sistemas multiagente

Los sistemas multiagente permiten una visión mucho más software de la red que las redes activas. Generalmente trabajan a un nivel mucho mayor, de presentación, sesión y aplicación en el modelo OSI, que las redes activas también. Y permiten introducir mucha inteligencia en la red que para ciertas aplicaciones es esencial.

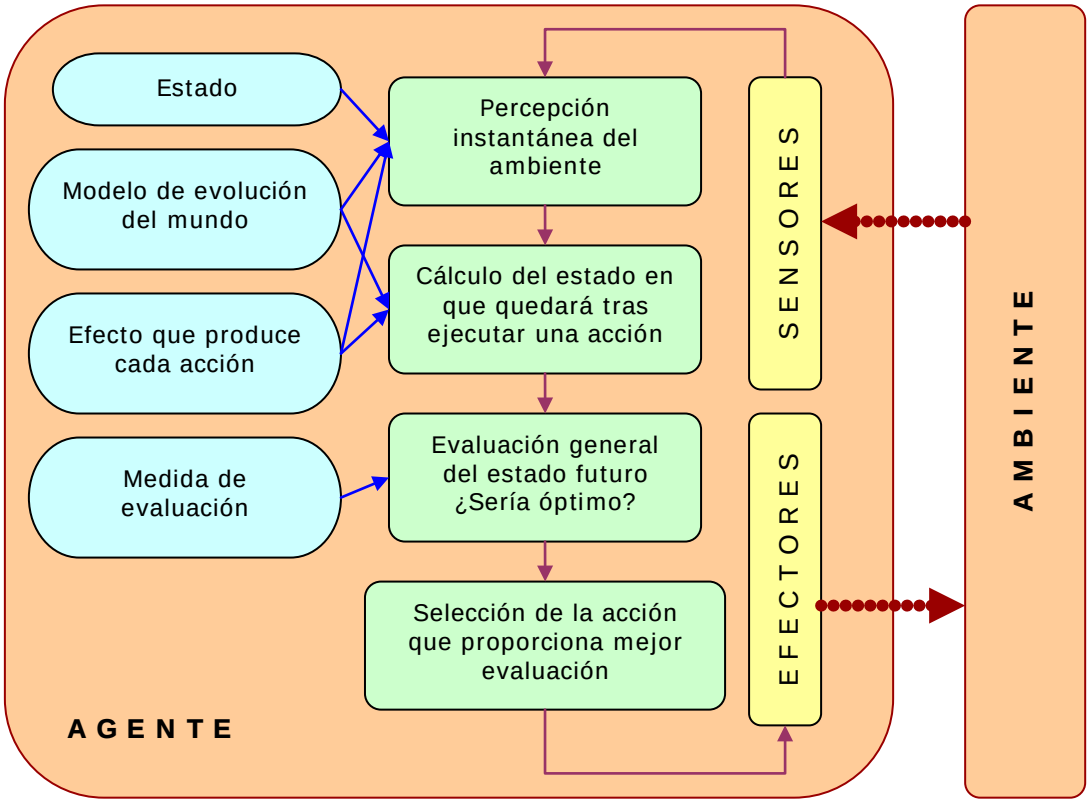
2.3.2.1. Agente inteligente

Hay muchas definiciones de inteligencia artificial, pero una muy clara, aunque genérica es aquella que dice que un sistema inteligente es “un sistema que piensa y actúa como un humano”. Esto es una afirmación un tanto atrevida, pero nos vale para lo que pretendemos.

La definición más sencilla de agente inteligente es la que lo define como un programa que puede percibir las características de su ambiente y actuar en consecuencia de forma inteligente. Se puede filosofar mucho en torno al concepto de inteligencia y al concepto de actuar inteligentemente. Simplemente diremos que un agente actúa inteligentemente cuando tras la percepción de su ambiente lleva a cabo un conjunto de acciones que le encaminan a conseguir su objetivo. El éxito de esta actuación inteligente, de la racionalidad de un agente, dependerá de varios factores:

- De la medida con la que se evalúa el éxito logrado.
- Del conjunto de todas las percepciones que ha tenido el agente.
- Del conocimiento que tenga el agente acerca del medio.
- Del conjunto de acciones que el agente puede llevar a cabo.

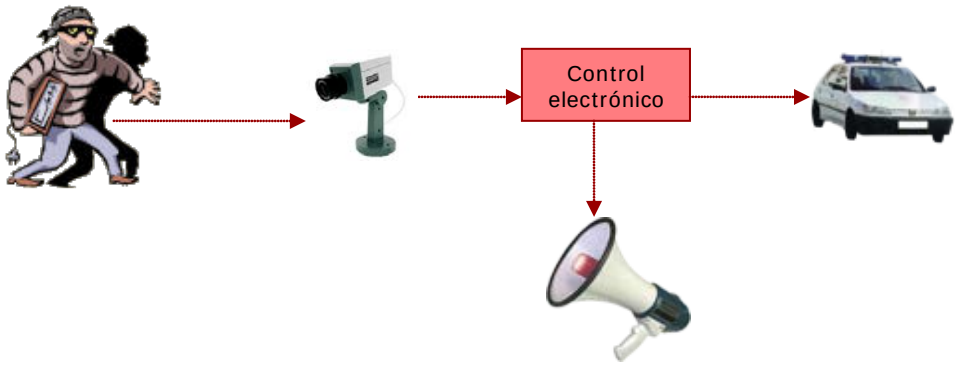
Es indispensable que el comportamiento de un agente inteligente sea autónomo; es decir, que no sólo utilice el conocimiento que el desarrollador introduzca en su interior en el proceso de creación, sino que sea capaz de aprender de sus errores, y utilizar las experiencias previas; incluso que sea capaz de inferir conocimiento a partir de los datos con los que cuenta. Por supuesto, también es deseable que el agente inteligente no sólo escoja las acciones que le permitan llegar a la consecución de una meta sino que escoja aquellas acciones que se lo permitan pero de una manera óptima. *Stuart Russell* describe en su obra *"Inteligencia Artificial: un enfoque moderno"* la siguiente arquitectura para un agente inteligente completo con las características que he expresado.



La teoría de agentes da para escribir muchas páginas, sin embargo todos los agentes tienen en común el hecho de que se crean para poder delegar en ellos ciertas tareas y es esperable que conseguirán realizarlas de la mejor forma posible en un tiempo acotado.

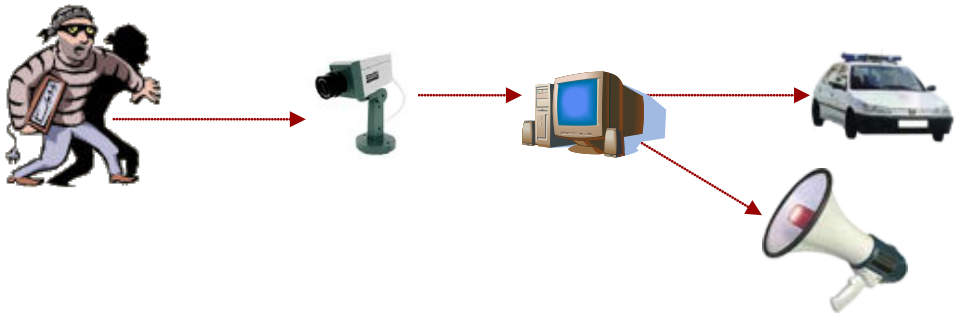
Supongamos un sistema de alarmas de seguridad de un almacén. Consta de una cámara con sensores capaces de detectar cualquier movimiento por mínimo que sea. Una alarma remota conectada directamente con la policía se activa cuando el movimiento es detectado

y además, para ahuyentar al posible ladrón, suena una sirena en el lugar del robo. Sería un esquema como se muestra a continuación.



En este esquema, el control central podría asemejarse a un agente inteligente: es autónomo, sustituye a un vigilante… pero realmente no es así. Este control sólo tiene la inteligencia implícita que el electrónico que lo ha creado ha depositado en él. Así, si un humano hiciese las veces de control, activaría la alarma y llamaría a la policía cuando un ladrón estuviese robando, pero no cuando un gato entrase en el almacén. Con el esquema de la figura anterior, un gato activaría los sensores de movimiento incorporados en la cámara y este "fallo" se repetiría siempre que entrara un gato.

En la siguiente figura, el control centrar se sustituye por una entidad inteligente, que bien podría ser un PC con un agente inteligente que recogiese los datos de la cámara y en base a su experiencia elaborase una respuesta adecuada.



En este caso, el comportamiento del control central si se puede considerar inteligente según la definición que hemos dado puesto que el agente inteligente que corre en el PC, si sigue los esquemas que hemos definido, puede aprender en qué situaciones dar la voz de alarma y en cuales no, según su experiencia.

2.3.2.2. Inteligencia artificial distribuida

Ya sabemos cómo es la estructura de un agente inteligente y los requisitos que deseablemente debería cumplir para que se le pueda llamar así, pero este esquema se puede mejorar para algunas situaciones.

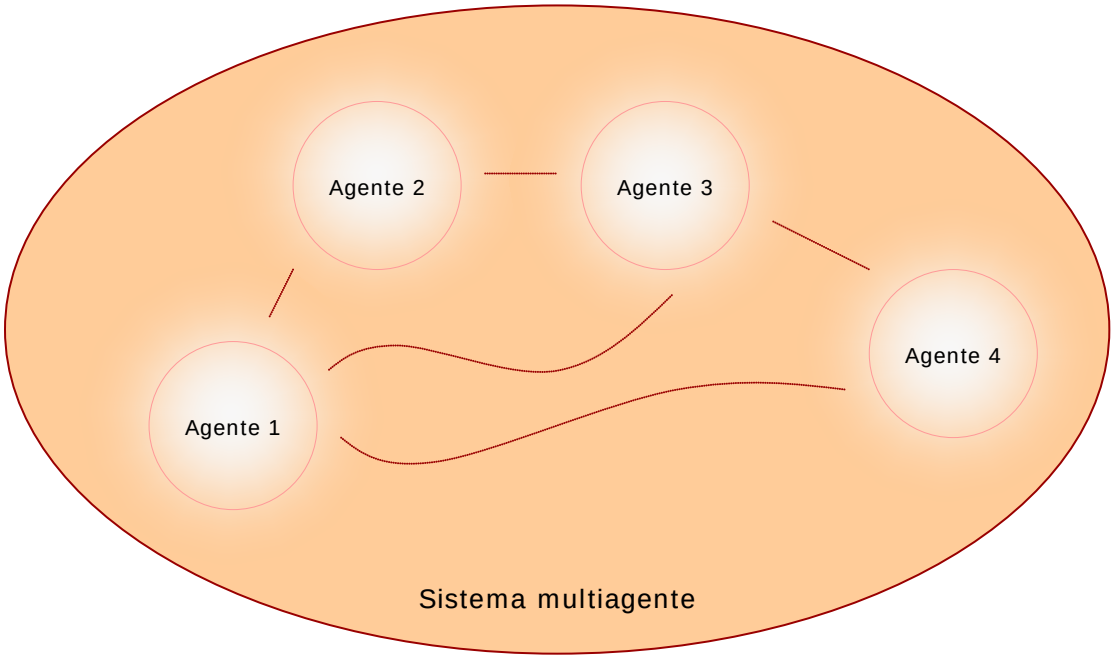
La programación paralela y distribuida ha influenciado en los últimos años todos los aspectos de la informática. El diseño del hardware, las redes, los sistemas de gestión, etcétera, se construyen la mayor parte de las veces de forma que el peso de la gestión no recaiga sobre una parte concreta, formando cuellos de botella y peligrosos puntos únicos de fallo.

En realidad hoy en día se tiende a la redundancia de sistemas, a la duplicación y a repartir el peso de la carga en cualquier escenario. En la inteligencia artificial, no podía ser menos. Así nace lo que se denomina inteligencia artificial distribuida, que es un subcampo de la inteligencia artificial que se centra en los comportamientos inteligentes colectivos que son productos de la cooperación de diversas "entidades", de diversos agentes inteligentes. La inteligencia artificial clásica sólo se centra, en principio, en el comportamiento individual. De este modo, se intenta representar en un modelo informático un hecho que ocurre en la vida real: que un agente no se comporta de igual forma en solitario que en grupo y que el peso del grupo ejerce una gran presión sobre la autonomía del agente como individuo independiente.

Para llevar a buen término este modelo propuesto por la inteligencia artificial distribuida, es necesario que exista algún tipo de *comunicación* que permita entrar en contacto a unos agentes con otros. Basándose en esa comunicación como requisito fundamental, es necesario que además el grupo de agentes, ya comunicados, sea capaza de desarrollar alguna de las siguientes características:

- **Intercambiar información**. Cada agente puede informar al resto de la parte del ambiente que conoce. Esto ahorra tiempo al resto, que confía en él. Se presupone que un agente inteligente no intentará engañar a otros.

- **Preguntar.** Los agentes deberían poderse preguntar por aspectos concretos del ambiente. Así un agente puede proyectarse en sus compañeros y extender su capacidad de percepción más allá de lo que podría trabajando de forma individual. Incluso podría percibir a través de sus vecinos cosas que el mismo no puede, por ejemplo si el vecino de un agente puede percibir la temperatura y él no, él puede percibir la temperatura indirectamente si se la pregunta a su vecino.
- **Responder.** Es el complemento de preguntar y por tanto forma parte de la misma acción.
- **Pedir u ordenar.** Sería bueno que un agente pudiese pedir ayuda para realizar una labor que él no puede realizar solo. Por ejemplo, si hay tres agentes conectados en línea, un podría pedir u ordenar al siguiente que pase un mensaje al tercero, haciendo de enlace. Sin colaboración esto no sería posible. El hecho de que se pueda pedir u ordenar implica que puede haber agentes que tengan más poder que otros, creándose una jerarquía.
- **Negociación.** Un agente puede ayudar a otro en una tarea a cambio de una contraprestación. El hecho de que la negociación exista en el comportamiento de los agentes implica que los agentes pueden tener comportamiento egoísta, avaricioso. Ya veremos que esto es muy importante.



Aspecto genérico de un sistema multiagente

A un conjunto de agentes comunicados que siguen un comportamiento social (colaboran) y que incorporan característica como las comentadas, es a lo que se llama **Sistema Multiagente**. Puede ocurrir que los agentes inteligentes del sistema multiagente, puedan desplazarse, explorando así nuevos ambientes. Un sistema multiagente se puede ver desde otra óptica como una única entidad que ofrece un servicio y donde cada agente por su parte, hace una porción de la tarea. También puede ocurrir que todos hagan la misma tarea pero desde puntos de vista distintos.

2.3.2.3. Tipos de sistemas multiagente

En una clasificación muy vasta, se pueden agrupar los sistemas multiagente en dos grandes grupos: sistemas multiagente donde cada agente busca el bien del grupo y agentes inteligentes donde cada agente busca prioritariamente su propio beneficio y posteriormente, el del grupo.

- **SMA donde cada agente busca su propio beneficio.** En estos sistemas, los agentes sacrifican a menudo el bien colectivo en su propio beneficio, pero no llegan a tener un comportamiento totalmente independiente. En el mundo real se pueden encontrar sistemas de este tipo en el mundo empresarial. Cuando una serie de empresas establecen relaciones de cualquier tipo, cada una de ellas está buscando su beneficio por encima de todo. Sin embargo ese beneficio aparece porque existe relación entre ellas. En realidad a ninguna de ellas le tiene por qué interesar que sus "socias" tengan más o menos beneficios. Lo podemos observar también en asociaciones de algunas empresas de refrescos y alguna ONG, donde se da un porcentaje de las ventas a la ONG. Ambas entidades obtienen recompensa: la ONG, dinero; la empresa de refresco, mayores ventas y una mejora de la imagen por colaborar con una ONG. En realidad, a ninguno de los dos implicados le interesa si el otro obtiene o no beneficios, mientras lo consiga él mismo.

- **SMA donde cada agente busca el beneficio colectivo.** Ocurre, por ejemplo, en una sociedad cooperativa. En este tipo de sociedades, el beneficio individual de cada trabajador depende directamente de los beneficios que tenga la empresa como conjunto; así es necesario buscar por encima de todo que la empresa tenga más

beneficios, pues así todo irá bien. En este tipo de sistemas, ningún componente por separado puede hacer que el beneficio aumente; han de comunicarse y ponerse de acuerdo todos sacrificando, cuando sea necesario, el propio beneficio en pro del beneficio del sistema completo.

En general estos dos tipos de sistemas se encuentran recogidos, ya en el ámbito de la informática, en los sistemas multiagente. Será decisión del ingeniero que diseñe el sistema multiagente que sea, el decidir si debe ser de un tipo u otro y esto estará determinado, en gran medida, por le tipo de problema que se desee solucionar.

2.3.2.4. Sistemas multiagente aplicados a las redes

Los sistemas multiagente ofrecen muchas posibilidades de aplicación en las redes precisamente por su distribución y por necesitar un medio para comunicarse.

- La primera aplicación que surge tras todo lo que hemos comentado en las páginas anteriores es la de dotar de inteligencia a un nodo activo, esto es, superponer el paradigma de las redes activas y de los sistemas multiagente y poder de este modo contar con nodos que además de activos, sean inteligentes. No sería demasiado complicado el crear un servidor de agentes en un entorno de ejecución de un nodo activo y que, en función del código o cápsula que se reciba, el nodo activo entre en juego activando una serie de agentes inteligentes para responder a la petición.

- El concepto de *Bandwidth Broker* como software gestor del ancho de banda de una red es otro de los casos en que los sistemas multiagente pueden funcionar bien. Puede haber diversos agentes colaborativos en la red que decidan entre ellos cómo, cuándo y a quién asignar los recursos de la red.

- El comercio electrónico es otras de las áreas donde los agentes inteligentes y los sistemas multiagente pueden desempeñar un buen papel. Con diversos agentes en la red que colaboren entre si, se pueden implementar servicios de búsqueda de productos, de búsqueda de ofertas, agentes que negocien el precio de un cierto

servicio y con todo ello el usuario puede usar este servicio para realizar sus comprar por medios telemáticos.

- En la gestión de las redes, los sistemas multiagente permiten que la gestión se descentralice, evitando el cuello de botella. Incluso los propios agentes pueden tomar el control de la red en momentos de congestión, caídas, fallos, ataques, etcétera, solucionando el problema por ellos mismos o aconsejando al administrador de la red. De esta forma se libera de trabajo a la estación de gestión y se agiliza la gestión de la red.

Existen muchas otras aplicaciones en estudio como por ejemplo el voto electrónico, la asignación automatizada de landas en redes ópticas, recuperación distribuida de información, implantación de servicios innovadores, trabajo cooperativo (colaboratorios, teletrabajo), monitorización, distribución de información (por ejemplo, noticias de USENET), sistemas de correo antispam, etcétera.

2.3.2.5. Ventajas de los sistemas multiagente.

Teniendo en cuenta la definición en la que un sistema multiagente es un conjunto de agentes en una determinada estructura que conforman el servicio ofrecido, pudiendo cada uno hacer parte del trabajo o haciendo todo el mismo pero desde diferentes enfoques, tenemos dos ventajas principales:

- Cuando se trabaja con un volumen de información importante, se puede distribuir de forma que, si los agentes fueran móviles podrían desplazarse al lugar donde reside la información y procesarla allí, devolviendo los resultados.

- Los sistemas multiagente favorecen la modularidad, favoreciendo la robustez y la reutilización de código. Se pueden diseñar agentes basados en otros agentes, obteniendo cada vez niveles mayores de abstracción.

2.3.2.6. Problemática de los sistemas multiagente

Hay que pensar en un sistema multiagente como una plataforma de agentes donde dichos agentes pueden ser activados, desactivados, donde cuentan con un ciclo de vida, donde se pueden desplazar de un nodo a otro de la red y donde para todas las operaciones hace falta autenticación. Algo así como ocurre con los *applets* de Java y que se encuentra resuleto gracias a la *Sandbox* de SUN y sus pertinentes políticas de acceso y seguridad.

Existen diversos problemas o riesgos con respecto a los sistemas multiagente que se asemejan a los problemas de las redes activas; algunos son los siguientes:

- **Suplantación de identidad.** En este ataque a la seguridad, un agente se hace pasar por otro para contactar con un tercero y poder acceder a recursos e información para los que en realidad no está autorizado. Cuanta más abstracción añadimos a una infraestructura de red y más flexibilidad, más aparecen los tradicionales problemas de cualquier software de red, como este.

- **Denegación de servicio.** Un agente consume recursos del nodo donde se ejecuta; además puede migrar. Con esta fórmula, un agente malicioso puede monopolizar los recursos de un dispositivo impidiendo que éste pueda atender las necesidades del resto de agentes.

- **Puerta trasera.** Un agente inteligente, como software de red que es, podría ofrecer servicios a otros agentes no autorizados para ello, de forma intencionada. El administrador de la red se puede considerar en este caso como un administrador de sistemas que tiene que velar por que esto no pueda ocurrir.

- **Repudio.** Los sistemas multiagente se basan en la colaboración de los agentes que lo componen. Para ciertas transacciones es imprescindible conocer qué agentes han participado en la misma, para tener un control exhaustivo de qué ha pasado, cómo ha pasado y quiénes han estado implicados. Sin control, un agente malicioso podría participar en una transacción y no aceptar dicha participación.

- **Accesos sin autorización.** Un agente puede utilizar los métodos y atributos de otro agente del mismo sistema multiagente. Sin embargo no es aconsejable que este acceso se pueda hacer sin autenticación. Si fuese así, un agente podría reiniciar el estado interno de otro, desbordar su pila de mensajes, etcétera.

La mayoría de estos problemas están detectados y solucionados, pero, al igual que con cualquier otro software, es muy difícil impedir que un código malicioso pueda hacer peligrar la estabilidad de una red. Con las propuestas actuales, es relativamente fácil averiguar quién ha caído un sistema, cuándo y dónde, pero es más difícil impedir dicha caída, al igual que es "fácil" saber quién ataca a un servidor web, pero es más difícil impedir que el ataque tenga efecto.

2.3.3. Tendencias

Tanto las redes activas como los sistemas multiagente siguen un mismo objetivo: hacer de las redes un medio más flexible para la fácil implantación de características nuevas y que esta implantación no sea dramática. Las redes activas ofrecen una visión mucho más "pegada al hardware" que los sistemas multiagente, que ofrecen una visión mucho más software. Las primeras están más enfocadas para proporcionar nuevos protocolos y nuevas características a nivel de red, en el trasporte de los datos, en la conmutación y el encaminamiento, etcétera. Los segundos están situados, conceptualmente, a un nivel de abstracción mayor y se dirigen más a la implantación de servicios sobre la red como por ejemplo un sistema de control de accesos, balanceo de la red, pasarelas de datos y una infinidad de aplicaciones potenciales.

Ya hemos visto que en GMPLS existen problemas a la hora de que una red de conmutación óptica pueda analizar el código que conmuta; en el caso de que la información sea una cápsula o código a ejecutar por un programa activo del nodo, el problema sigue existiendo y habría que ver cómo se puede solucionar. El hecho es que cada vez se tiende a tener redes más puramente hardware, compactando los niveles superiores hasta dejarlos prácticamente en el nivel de red y por otro lado las redes activas y los sistemas multiagente intentan ofrecer soluciones más informáticas, más software, a problemas existentes; Por tanto no es sencillo vaticinar cómo evolucionará este entramado en los próximos años,

Seguramente las redes activas jugaran un papel importante a la hora de gestionar una red y de establecer soluciones flexibles para integrar tecnologías y los sistemas multiagente permitirán ofrecer servicios como gestión del ancho de banda, seguridad, autenticación, etcétera.

2.3.4. Conclusiones

No es incompatible trabajar con redes activas y con sistemas multiagente simultáneamente; es más, sería beneficioso ofrecer cierta flexibilidad e "inteligencia" en los niveles de red y transporte gracias a las redes activas y además contar con servicios distribuidos inteligentes implementados mediante sistemas multiagente en las capas superiores.

Ambos enfoques permiten indagar en aspectos en los que es difícil en la investigación tradicional sobre *Networking* pero, mi impresión a la hora de leer sobre este tema es que no hay una dirección clara y única. No he encontrado fácilmente definiciones de redes activas o sistemas multiagente aplicados a las redes que permitan separar ambos conceptos. No me ha parecido que haya una arquitectura muy ampliamente aceptada sino más bien un buen número de propuestas distintas cada una con sus ventajas e inconvenientes, lo cual es bueno para abrir distintas líneas de investigación, pero es un problema a la hora de que la industria se lance a implantar una de las propuestas.

3. Soporte de garantía de servicio (GoS) sobre MPLS mediante técnicas activas

3.1. Carencias actuales

Actualmente MPLS permite tener redes rápidas que aprovechen las capacidades de las tecnologías ópticas. Esto se hace a costa de fiarse enormemente en que la red no va a fallar.

Así como en los tiempos de X.25 las redes eran bastante toscas y por tanto se debía hacer un control de errores, petición de retransmisiones, etcétera, punto a punto, actualmente con redes que tienen un porcentaje de errores o pérdidas ínfimo parece que no es muy rentable es perder tiempo y recursos comprobando los posibles errores en el interior de la red, puesto que rara vez van a ocurrir. Obviando muchos detalles importantes, esto permite que las redes sean más rápidas y que el canal se aproveche mejor para enviar datos más rápido. El control de errores se encargarán de hacerlo extremo a extremo el emisor y receptor y así los nodos intermedios de la red serán más baratos y eficientes.

El problema surge cuando esa mínima posibilidad de fallo ocurre, pues es el momento en que gran parte de tráfico se perderá; ya se encargarán los protocolos de niveles superiores de solicitar la retransmisión, pero la latencia que esto puede suponer es grande. Para ciertos tipos de tráfico, sensibles a la fiabilidad, MPLS debería poder asegurar que no se verán afectados o que serán afectados en mucha menos medida, pero no lo hace. MPLS tiene dos problemas principales a la hora de poder garantizar a ciertos tráficos que llegarán al destino sin problemas:

- Qué hacer y cómo actuar ante la caída de un camino físico cuando por él transitan paquetes de un flujo que debe ser priorizado.
- Como responder ante la congestión en los nodos cuando los paquetes descartados son también pertenecientes a estos tipos de tráfico.

3.2. Propuesta tecnológica de soluciones

Para intentar paliar las carencias comentadas, este proyecto investiga las distintas opciones o posibilidades de implantar mejoras sobre MPLS, sin quebrar el estándar ni el principio de transparencia que en todo momento debe estar presente.

3.2.1. Garantía de servicio

Antes de pasar a estudiar una arquitectura que permita el soporte de Garantía de Servicio sobre MPLS es necesario definir en unos términos claros qué es concretamente la Garantía de Servicio.

Podemos entender como Garantía de Servicio a la posibilidad de ofrecer a un cierto flujo de tráfico la seguridad de que en condiciones normales será tratado preferentemente con respecto al resto de flujos y que, en caso de la existencia de los problemas comentados anteriormente (pérdida de paquetes y caídas de enlaces), la arquitectura pondrá los medios necesarios para que en cualquier caso el flujo que requiera Garantía de Servicio sea menos perjudicado, tanto menos cuanta más Garantía de Servicio haya sido especificada para ese flujo y siempre en función de las posibilidades de los nodos que atraviese.

Obsérvese que no se define en términos exactos la Garantía de Servicio, sino en términos relativos (preferentemente, en función de las posibilidades…). Esto es debido a que la Garantía de Servicio se puede ver como una aproximación a la calidad de servicio y fiabilidad, pero sobre una tecnología subyacente que no soporta estos parámetros como lo si lo hace ATM, por lo que en ningún caso se podrán asegurar peticiones como “que no sean descartados más del 10% de los paquetes del flujo”, “que el retardo nunca sobrepase x milisegundos” o que “la variabilidad del retardo máxima sea de x”. Es más, ni siquiera se puede garantizar una cadencia del flujo constante, simplemente MPLS, la tecnología que debe soportar GoS, no lo concibe. Aún así es posible garantizar que **el tráfico deseado será tratado de forma privilegiada siempre que sea posible**.

3.2.2. Objetivos

Ya se ha comentado en puntos atrás algunas de las carencias y los problemas de MPLS. Se han repasado el conjunto de tecnologías coetáneas con MPLS y de donde este proyecto beberá para producir una solución. Se ha aclarado la definición de Garantía de Servicio (GoS). El objetivo principal de este proyecto será definir un conjunto de técnicas, basadas en ideas procedentes de las redes activas y los sistemas multiagente, para permitir que MPLS pueda proporcionar garantía de servicio a flujos privilegiados acercando esta tecnología a las ventajas que actualmente está intentando ofrecer GMPLS sobre medios ópticos. Todo ello preservando el principio de transparencia que posibilite la coexistencia entre los nodos activos que se proponen en este PFC y una arquitectura MPLS tradicional.

3.2.3. Decisiones de diseño y alcance de la propuesta

MPLS es muy amplio, además está en pleno estudio con lo que las propuestas de cambio son frecuentes y la aparición de borradores y RFC's es continua. Por ello ha sido ardua la labor de decidir qué partes deben ser estudiadas en detalle y qué partes deben ser tomadas con menos relevancia al no formar aún parte del MPLS "estándar". Además hemos tenido que tomar decisiones sobre las modificaciones que hacer a los distintos protocolos para conseguir el objetivo de este PFC.

3.2.3.1. Elección del protocolo de nivel de red

Aunque MPLS está ideado para funcionar con cualquier protocolo de red, lo cierto es que en la práctica la gran mayoría de las investigaciones y desarrollos están centrados en soportar el protocolo de red IPv4 e IPv6. La extrapolación de todo lo propuesto en este proyecto a cualquier protocolo de red no puede hacerse de forma inmediata puesto que algunas de las propuestas implican la existencia de espacio en las cabeceras de nivel de red e incluso de transporte, para alojar información de control.

Dado que la mayor parte del tráfico de Internet está aún funcionando sobre IPv4, en esta propuesta **nos centraremos en el uso del protocolo de red IPv4** aunque siempre previendo la posibilidad de extrapolar las propuestas vertidas en este documento al

protocolo IPv6, futuro de Internet y hermano mayor de IPv4 con el que comparte algunas similitudes.

3.2.3.2. Marcado de GoS en los paquetes

Como en las redes activas, en este PFC se propone que los propios paquetes incorporen el distintivo que hará que los encaminadores y conmutadores activos los traten de forma diferente al tráfico sin marcar. Como no vamos a usar un sistema de cápsulas ni vamos a usar ANEP porque este PFC no define un servicio de una red activa sino una arquitectura que use técnicas que se aproximen a las redes activas, uno de los mayores problemas es encontrar espacio para marcar los paquetes en todos los niveles de la comunicación y que además sea transparente para los nodos implantados actualmente. IP/MPLS no permite caracterizar el tráfico que fluye a través de un dominio MPLS de una forma suficientemente concisa; es el típico problema de IPv4. Los paquetes son variables, relativamente grandes, el tráfico es a ráfagas… por lo que no pretendemos intentar establecer un sistema tan sumamente fino de parametrización del tráfico o los recursos como por ejemplo permite ATM, directamente diseñado para ello.

Así, tras varias reflexiones consideramos que sería suficiente con establecer cuatro grados o niveles de garantía de servicio distintos. Posiblemente el establecimiento de más niveles en condiciones de tráfico real de Internet, fuese muy difícil de mantener y cumplir. Por otro lado, de alguna forma hemos de marcar los paquetes que deseen contar con algún método para cuando un enlace falle. Los cuatro niveles de GoS se pueden codificar con dos bits y la existencia de LSP de respaldo, con un tercero. En la siguiente tabla se observa mejor cómo se utilizarían tres bits para representar todas las opciones:

LSP	GoS_1	GoS_0	Significado
0	0	0	Paquete no marcado con GoS. A todos los efectos, es un paquete tradicional.
0	0	1	Paquete marcado con nivel de GoS 1 y sin solicitud de LSP de respaldo.
0	1	0	Paquete marcado con nivel de GoS 2 y sin solicitud de LSP de respaldo.
0	1	1	Paquete marcado con nivel de GoS 3 y sin solicitud de LSP de respaldo.
1	0	0	Paquete no marcado con GoS pero con solicitud de LSP

			de respaldo.
1	0	1	Paquete marcado con nivel de GoS 1 y con solicitud de LSP de respaldo.
1	1	0	Paquete marcado con nivel de GoS 2 y con solicitud de LSP de respaldo.
1	1	1	Paquete marcado con nivel de GoS 3 y con solicitud de LSP de respaldo.

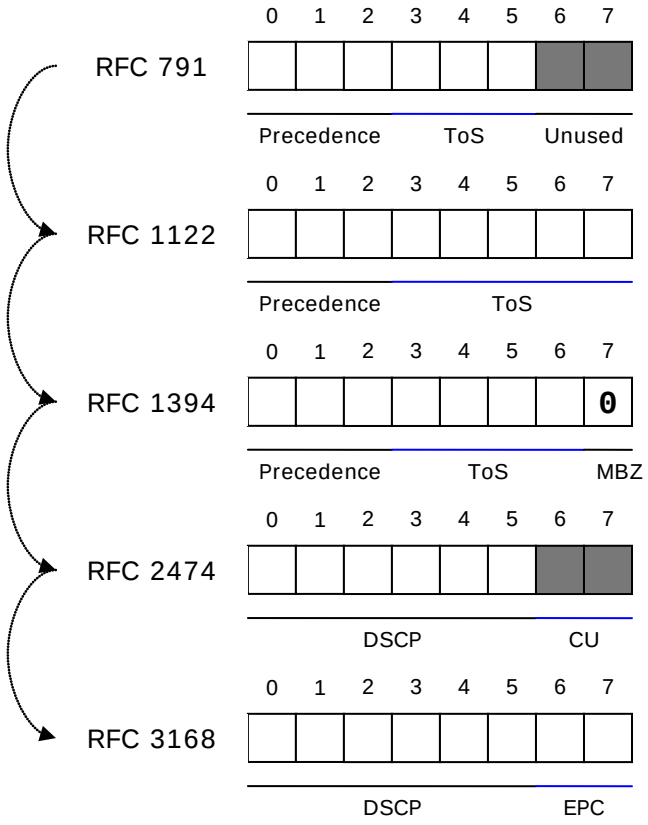
La cuestión es encontrar la forma de especificar esos tres bits necesarios en las cabeceras de nivel de red, transporte y MPLS de los paquetes que lo requieran. Es el primer paso para que la GoS pueda ser ofrecida a estos flujos; asimismo es uno de los problemas principales puesto que la definición de los campos de las PDU en cualquier protocolo no puede hacerse de forma azarosa sino que debe estar bien meditada y debe realizarse con todas las precauciones para seguir cumpliendo los estándares.

Establecer dicha marca en MPLS no es demasiado problemático. En el RFC 3031, donde se especifica MPLS, se define el valor 1 para la etiqueta MPLS como un valor especial. Literalmente extraído de dicho RFC, *"un paquete que lleve esta etiqueta debe ser desviado a un módulo aparte del nodo para ser tratado de forma especial"*. No se especifica nada más en el RFC pero parece muy buena opción el utilizar como etiqueta el valor 1 para denotar que el paquete es especial, que está marcado para obtener garantía de servicio.

El campo EXP, de 3 bits, no se usa para nada en esta etiqueta especial; Casualmente es el número de bits que necesitamos para almacenar a nivel MPLS el grado de Garantía de Servicio requerido por el paquete y la necesidad o no de proteger el LSP por el que éste vaya a circular. De este modo el paquete será retenido y procesado en base a las necesidades que estos bits indiquen. Este marcado a nivel MPLS lo realizará el LER de entrada al dominio MPLS; un LER que debe ser activo y contar con la capacidad de interpretar que el paquete entrante tiene esas necesidades. Obviamente, la información que necesita para saber que valor debe codificar e introducir en el campo EXP debe venir incorporada en el propio paquete IPv4 que desea ingresar al dominio MPLS. Necesitamos marcar también a nivel de red los bits de GoS y LSP de respaldo.

En la cabecera IP no hay tres bits libres al menos de una forma obvia. El primer lugar donde se nos ocurre introducir esos tres bits es en el campo *ToS*, que cuenta con 8 bits y

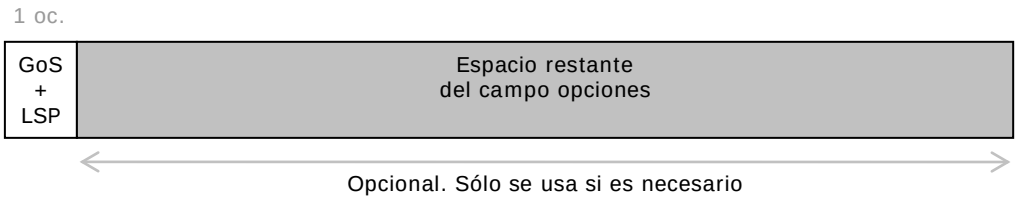
cuyo uso se ha modificado en diversas ocasiones. Sin embargo, un estudio de la evolución de este campo en los distintos RFC's nos muestra lo siguiente:



Es decir, que los bits disponibles (en gris) se agotaron con la especificación realizada en el RFC 3168. Podríamos hacer una reinterpretación del campo *ToS* pero no nos pareció una buena idea por algunas razones; entre ellas porque actualmente se utiliza para especificar niveles para *DiffServ* y para notificar la congestión de los nodos. Es posible que la idea de incorporar servicios diferenciados junto con la propuesta de garantía de servicio que proponemos sea atractiva y por ello no es buena idea tener que seleccionar entre una u otra opción por el mero hecho de que el campo *ToS* sirva para ambas.

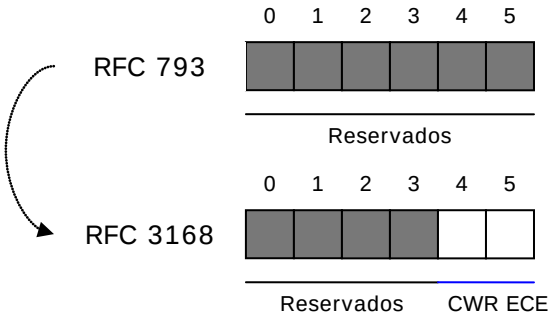
Después de estudiar las posibilidades, parece que la cabecera IPv4 no dispone de 3 bits libres si no es en el campo opciones (opcional, de tamaño variable y máximo de 40 octetos) puede ser un buen sitio para lo que necesitamos. Así que **usaremos el primer octeto del campo opciones de la cabecera IPv4 para colocar en él los 3 bits que especifican los niveles de garantía de servicio y el requerimiento o no de LSP de respaldo**. Con ello tendremos marcados los paquetes en los dos niveles más importantes

para nuestro objetivo, el de red y el nivel MPLS (nivel 2+). En la siguiente figura se muestra cómo quedaría en este momento formateado el campo opciones de IPv4.

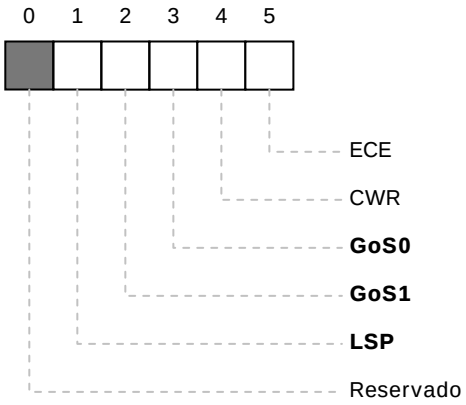


Aunque en este proyecto no subimos más allá del nivel de red puesto que los niveles implicados en la comunicación MPLS son el de red, el de enlace y el nivel 2+ o MPLS, lo cierto es para ser una propuesta realista **debe contemplarse la posibilidad de marcar a nivel de transporte el grado de GoS que se desea para los paquetes privilegiados desde el nivel de aplicación**. Igual que anteriormente hemos decidido utilizar IPv4, ahora decidimos utilizar TCP puesto que parece poco lógico que un paquete que sea transportado sobre UDP pueda requerir GoS. Siguiendo el modelo TCP/IP, los datos se marcarían a nivel de aplicación directamente por el usuario (un usuario podrá ser un nodo de acceso a la red de un ISP) y posteriormente la aplicación de red marcaría los segmentos TCP que al ser encapsulado sobre paquetes IPv4 darían como resultado los paquetes que tratamos en este PFC. Este proceso se realizaría de la siguiente forma:

A nivel de aplicación el usuario decide abrir una sesión para la retransmisión de paquetes con GoS; indica esta operación mediante la elección de un puerto de destino al abrir el *socket* TCP (al acceder al nivel de transporte). Así como por ejemplo para acceder a un servicio de correo se accede al puerto 110 o a un servicio SSH al puerto 22, dedicaremos siete puertos concretos para abrir sesiones TCP con cada uno de los siete niveles GoS disponibles (GoS + LSP de respaldo). Esto provocará que se marque **a nivel de transporte** los tres bits que necesitábamos incluir en este nivel; ¿cómo y dónde? En la cabecera TCP hay un campo de seis bits reservados desde la creación de TCP; durante mucho tiempo ese campo ha permanecido intacto, pero en los últimos años se ha comenzado a hacer uso de algunos de sus bits, concretamente de dos, para poder marcar alguna de las opciones de servicios diferenciados. En la siguiente figura se muestra la evolución de dicho campo en el transcurso del tiempo.



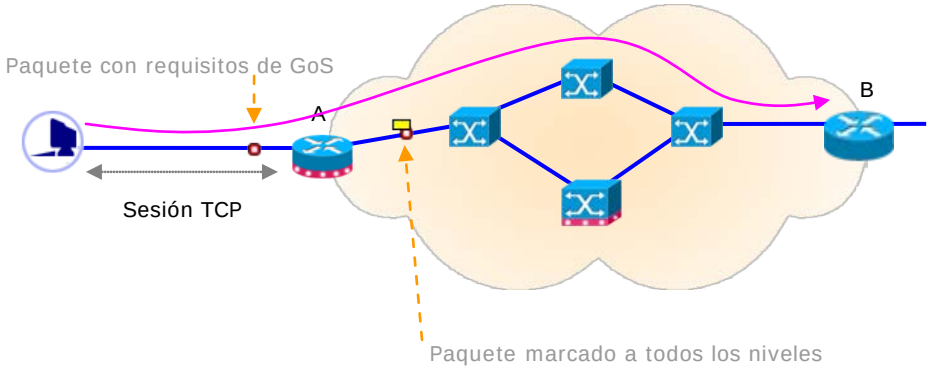
Por lo que tenemos todavía cuatro bits disponibles (en gris) para lo que necesitamos. Nosotros tenemos únicamente que colocar en la cabecera TCP 3 bits por lo que aún quedaría un bit para otros usos. Usaremos los bits 1-3 de este campo y de la siguiente forma:



De esta forma podemos pasar sin problemas la orden de priorizar el paquete, desde el nivel de aplicación hasta el nivel de red pasando por el de transporte. **No se va a implementar esta característica porque el simulador sólo se centra en los niveles IP y MPLS** pero era necesario profundizar en esto para no proponer soluciones a nivel de red que no fuera factible llevar a cabo en niveles superiores.

El proceso de conexión del usuario a través de un puerto concreto para priorizar un tráfico no se realizará extremo a extremo de la comunicación. La aplicación que permite esta operación entrará en contacto desde el usuario directamente hacia un nodo que el ISP disponga para ello y que se encargará de marcar los paquetes a nivel de transporte y red. A partir de ahí el paquete circulará normalmente por la nube del ISP hacia el destino y los nodos activos por lo que atraviese, serán capaces de detectar esta configuración especial

del paquete y emprenderán las acciones que sean necesarias. La figura siguiente muestra este hecho.



Marcado de un paquete y entrada en el dominio MPLS

Después de muchos razonamientos y de sopesar las ventajas y los inconvenientes de muchas posibilidades, conseguimos poder marcar los paquetes que deseamos que obtengan GoS desde el nivel de aplicación, pasando por el de transporte y el de red hasta el nivel MPLS; en todos los casos respetando el funcionamiento estándar de todos los protocolos implicados y proponiendo para ello unos métodos razonables.

3.2.3.3. Identificación global de los paquetes

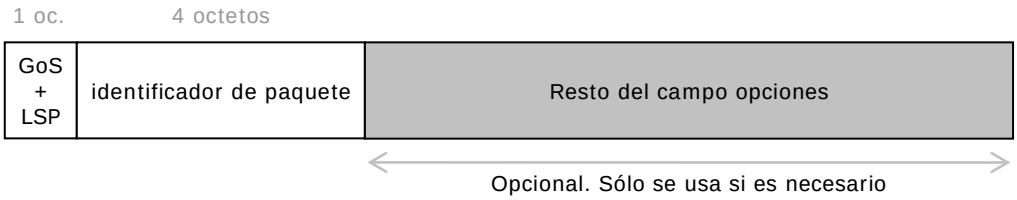
El segundo problemas al que se enfrenta este proyecto consiste en que la propuesta de GoS que se pretende crear incluye, entre otros procedimientos, la capacidad de recuperación de paquetes marcados con GoS que sean descartados en un nodo activo en el interior de un dominio MPLS. Necesitamos por tanto tener una forma de identificar en un momento dado a un paquete concreto en el contexto global del dominio MPLS. Tenemos que las direcciones IP identifican ya de por si globalmente a cada nodo de la topología de una red. Sin embargo no constituye una clave primaria para identificar a todos y cada uno de los paquetes generados por un mismo nodo, de forma independiente. Así pues necesitamos combinar la dirección IPv4 con un identificador único para cada paquete emitido por un generador de tráfico IPv4 y además, por tanto, ese identificador único debe acompañar al paquete IPv4 (si tiene marca de GoS) en todo su trayecto a través del dominio MPLS al que se circunscribe este proyecto.

Por tanto, **consideraremos identificador único de un paquete MPLS en el dominio a el par formado por la dirección IP del emisor del paquete y el identificador único con el que dicho emisor marca a cada uno** de ellos.

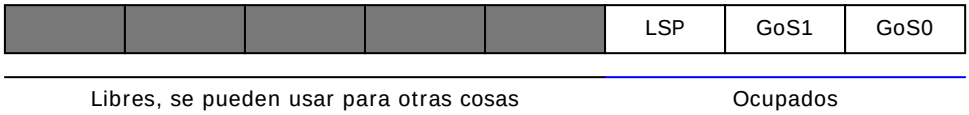
Ahora bien, ¿de qué tamaño debe ser el identificador? Con 16 bits tenemos que podemos tener 65.535 paquetes distintos a la vez en un dominio MPLS de un mismo emisor sin posibilidad de confusión entre ellos. Sin embargo a partir de ese momento el emisor comenzará a generar identificadores desde el inicio y podría darse la situación de que haya dos paquetes con el mismo identificador en la red (generalmente con el mismo destino) y eso podría provocar que se solicitara la retransmisión de un paquete y se retransmitiera otro con el mismo identificador. 16 bits nos parecen pocos más aún si tenemos en cuenta que se trata de una tecnología para troncales donde el volumen de información se prevé grande.

Con 32 bits podemos direccional hasta 4.294.967.296 paquetes sin empezar a repetir la secuencia y nos parece un número adecuado puesto que es improbable de que se repita un identificador sin que el primero de ellos haya ya abandonado el dominio MPLS.

Por tanto **el identificador que usaremos será de 32 bits (4 octetos) que deberá ir, una vez más, en el campo opciones de IPv4**, justo después del octeto destinado a definir los niveles de GoS y antes de que comience la pila de direcciones IPv4 de los nodos activos atravesados. Así, ya de forma definitiva, el campo opciones de IPv4 quedaría formateado para soportar GoS como se muestra en la siguiente figura:



Y los tres bits de GoS y de LSP de respaldo, se ordenan de la siguiente forma dentro del campo GoS+LSP de 8 bits:

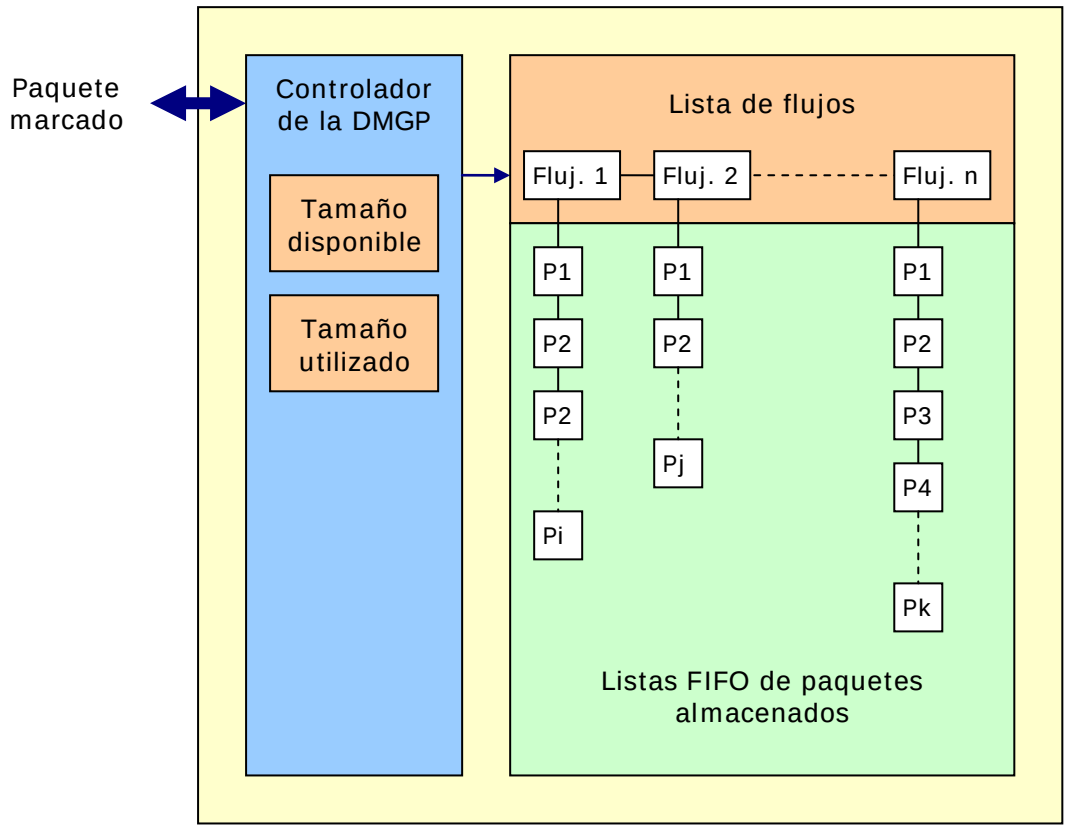


Así que como podemos observar, aún nos quedan libres los 5 primeros bits del primer octeto del campo opciones.

3.2.3.4. Memorias temporales (DMGP)

En este punto hemos conseguido marcar los requerimientos de GoS y de LSP de respaldo en los paquetes que sea necesario a todos los niveles. Además hemos conseguido poder identificar a cada paquete globalmente dentro del dominio MPLS para poder solicitar su retransmisión en caso de que haya un descarte. Sin embargo parece claro habrá que almacenar en algún lugar dentro de los nodos activos los paquetes marcados con GoS que lo vayan atravesando; así se podrá buscar un paquete cuando el nodo activo reciba de otro nodo activo una solicitud de retransmisión.

La propuesta de este PFC es la memoria DMGP (Dynamic Memory for GoS PDU) o Memoria Dinámica para PDU's con GoS, que tiene la siguiente estructura interna.



En ella, los paquetes marcados con GoS son almacenados. Un paquete se envía a la DMGP. Su controlador comprueba si existe una entrada de flujo para ese paquete; si no es así, comprueba si tiene reservas de memoria para satisfacer la demanda requerida por el paquete en base a su grado de GoS. Si tiene, le reserva un tamaño fijo a ese flujo, crea la entrada e inserta el paquete en DMGP, asociándolo a dicha entrada de flujo que se acaba de crear.

Siguientes paquetes del mismo flujo no necesitarán que se les cree una entrada de flujo. Se insertarán en la memoria DMGP si queda espacio necesario para él del total que se reservó a ese flujo. Si no queda, se comienzan a eliminar paquetes de la cola de paquetes de ese flujo, en orden FIFO, hasta que quede espacio suficiente para que pueda ser insertado el nuevo paquete. De este funcionamiento se derivan dos conclusiones:

– Cuanto más tiempo pasa es más improbable que un paquete que llegó hace tiempo esté en DMGP; habrá cedido el espacio a paquetes más nuevos.

– La reserva de tamaño para un flujo es exclusivamente para ese flujo. Ningún flujo, que llegue después a la DMGP influirá en aquellos que ya tienen su espacio reservado.

Si un paquete llega a la DMGP y pertenece a un flujo que no tiene entrada en la DMGP y se comprueba que la DMGP ha reservado ya todo el espacio del que disponía, para ese flujo no se reserva memoria; el nodo actual no podrá retransmitir paquetes de ese flujo. Aún así, este nodo todavía podrá ofrecer el resto de servicios a ese tráfico que tiene requisitos de GoS, por ejemplo solicitar su retransmisión a otros nodos.

El tamaño reservado para cada flujo es constante, pero no tiene que ser el mismo para cada DMGP de cada nodo. El tamaño se asigna por porcentajes del total de la DMGP y siempre según el nivel de GoS incorporado en el paquete, como se muestra a continuación:

		Algunos ejemplos		
Grado de GoS	% DMGP asignado	DMGP = 1 KB	DMGP = 100 KB.	DMGP = 1000 KB.
1	4	41 Bytes por flujo.	4,1 KB. Por flujo	41 KB. Por flujo
2	8	82 Bytes por flujo.	8,2 KB. Por flujo	82 KB. Por flujo
3	12	123 Bytes por flujo.	12.3 KB. Por flujo	123 KB. Por flujo

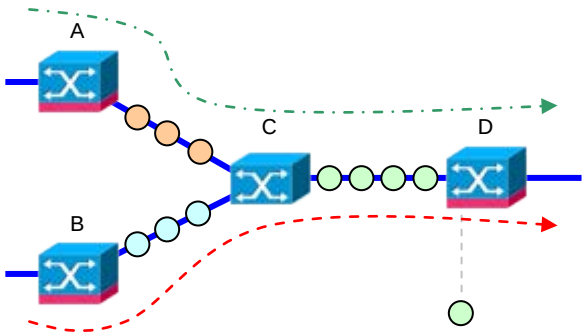
Donde se ve que claramente a mayor DMGP mayor número de paquetes se reservarán para cada flujo. Número que será fijo mientras no se aumente la DMGP.

Esta forma de repartir el espacio de la DMGP implica que tras un número determinado de entradas de flujos que hayan reservado espacio, la memoria DMGP no podrá dar servicio a más flujos. Con el tipo de reparto actual, se permiten un máximo de 25 flujos con GoS 1, si son todos de este tipo, 12-13 flujos con GoS 2, si son todos de este tipo, o bien 8 flujos con GoS 3, si son todos de este tipo. Si son combinaciones, el número de flujos soportados por la DMGP variará. En cualquier caso, se presupone que el número de flujos con requerimientos de GoS en el dominio MPLS debería ser anecdótico en comparación con el número de flujos sin estos requerimientos; Además de esto, para que hubiese problemas los flujos tendrían que coincidir en el tiempo porque cuando un flujo termina, la DMGP elimina su entrada y vuelve a quedar espacio sin reservar, aprovechable para otro flujo.

El tamaño de la memoria DMGP depende en gran medida del tráfico que circule por la red, de los requerimientos de los usuarios finales, del tipo de aplicación, etcétera. En última instancia debería ser el ingeniero de red encargado del mantenimiento de la troncal MPLS el que hiciese un estudio para modelar de forma adecuada el tráfico existente en la red y, conforme a ello, establecer el tamaño apropiado de las memorias DMGP, igual que se puede modificar las unidades de asignación (clusters) en los sistemas de ficheros de los sistemas operativos para mejorar las prestaciones en base al tamaño medio de los ficheros del sistema, el uso del *host*, etcétera.

3.2.3.5. Marcado de la ruta seguida, para retransmisiones

Supongamos la siguiente situación:



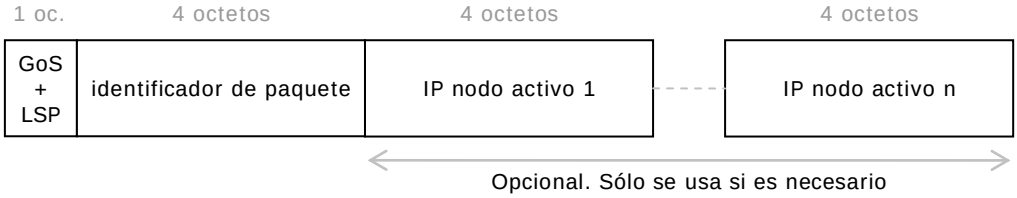
A, C y D son nodos activos. C es un nodo MPLS tradicional. A D llegan paquetes que provienen tanto de A como de B, pero son indistinguibles para él puesto que sólo conoce la etiqueta y el puerto por el que llegan dichos paquetes. Y sólo sabe que C es quien se los envía. Podría distinguir por la etiqueta la procedencia pero tampoco sería algo fiable puesto que C podría incorporar mecanismos de agregación que fusionaran en uno solo flujo los dos provenientes de A y B. D pierde un paquete por saturación y entonces debe saber a quién solicitar la retransmisión. A C no, porque C no es activo y no entendería la petición.

Necesitamos poder almacenar en algún lugar de los paquetes los nodos activos por los que ha ido pasando un paquete marcado con GoS para que, en caso de pérdida del paquete, se pueda solicitar a los mismos que lo retransmitan. Para ello los nodos activos deben romper la filosofía MPLS de que un LSR no debe observar más allá de la cabecera MPLS. Los nodos activos tendrán que marcar a nivel IP sobre el paquete sus direcciones de red. Sin

embargo no se invalidaría el estándar MPLS que ya se anticipó a esta situación estableciendo el mecanismo de "desvío a un módulo especial" de los paquetes con etiqueta = 1.

El hacerlo sobre el nivel IP es una decisión que hemos tomado porque en la cabecera MPLS no hay espacio alguno y, aunque se podrían tomar algunos bits de la propia etiqueta MPLS (que tiene 20, y por tanto se le podrían tomar algunos) esto rompería el principio que perseguimos de que los nodos no activos existentes en una red no tengan problema alguno al tratar tráfico marcado con GoS, es decir, transparencia de cara a MPLS estándar.

Aprovechando que hemos hecho uso del **campo opciones de IPv4, pensamos también transformar dicho campo en una pila de direcciones IP donde almacenar las direcciones de los nodos activos que el paquete atraviese**, circularmente, de forma que siempre se almacenen los *n* últimos nodos atravesados (dadas las limitaciones espaciales de este campo). Podrían ser (40 – 1 - 4) / 4 = 8 direcciones de nodos activos lo cual es más que suficiente para un dominio MPLS puerto que no se está proponiendo una sustitución de los nodos MPLS sino la incorporación de algunos nodos MPLS activos. Además retroceder hasta 8 nodos activos para solicitar una retransmisión es una buena cifra.



Así que con todo lo visto en los apartados anteriores, tenemos los paquetes marcados con el nivel GoS necesario, en todos los niveles de la comunicación. Tenemos identificados globalmente en el dominio MPLS todos los paquetes con requerimientos de GoS, tenemos un lugar organizado donde almacenar, dentro de los nodos activos, los paquetes que tengan ese requerimiento y tenemos también la posibilidad de conocer hasta las últimas 8 direcciones IP de nodos activos por los que ha pasado el paquete, para poder solicitarles la retransmisión del mismo. Además en ningún momento se han sobrepasado los estándares de los protocolos implicados. Las bases están sentadas. Ahora es necesario describir el

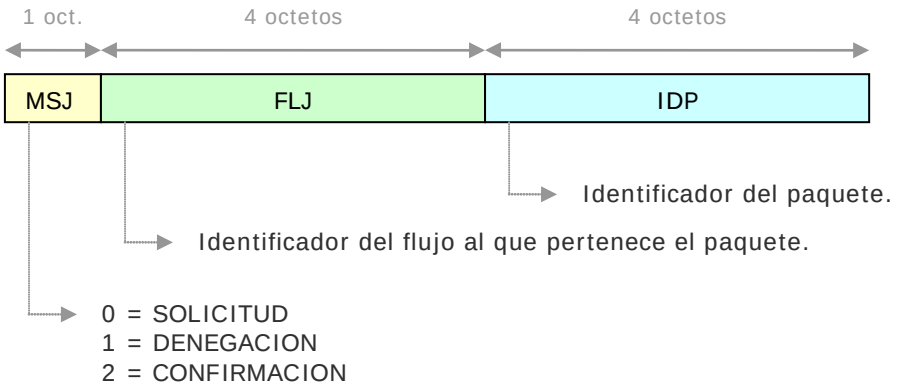
conjunto de protocolos y la arquitectura de los nodos activos para aprovechar estas nuevas ventajas y ofrecer GoS y LSP de respaldo sobre MPLS.

3.2.3.6. Protocolo para retransmisiones (GPSRP)

Necesitamos un protocolo que sirva para controlar el proceso de almacenamiento, búsqueda y retransmisión de paquetes marcados con garantía de servicio. Es deseable que dicho protocolo funcione vía IPv4 puesto que ya contamos en esta propuesta con que va a ser el protocolo de nivel de red que va a estar disponible; **A la propuesta formulada en este proyecto la hemos llamado *GPSRP (GoS PDU Store and Retransmit Protocol)***, o Protocolo de Almacenamiento y Retransmisión de PDU's con GoS.

A la hora de crear este protocolo se ha pensado en él contexto en el que será utilizado y el propósito del mismo. GPSRP entrará en funcionamiento cuando un nodo activo esté congestionado y entre los paquetes que descarte, se encuentren paquetes marcados con algún nivel de Garantía de Servicio. El propósito es realizar una recuperación local que sea más rápida que la retransmisión extremo a extremo realizada por los protocolos de niveles superiores; más aún, la retransmisión debería hacerse efectiva antes de que los *Timeouts* de los extremos expiren pues una vez iniciada la petición de retransmisión por parte de éstos, no tiene sentido seguir con el proceso de recuperación local.

Así pues, GPSRP debe ser rápido; y debe evitar producir más congestión de la que ya habrá en ese momento en el nodo activo y sus alrededores. Al final la decisión ha sido crear un protocolo muy escueto con sólo tres tipos de mensaje, cuyos paquetes sean muy reducidos, de tamaño fijo y que sean transportados por IP. El formato de los paquetes GPSRP es el siguiente:



El primer campo del paquete ocupa 1 octetos y sirve para especificar el tipo de paquete GPSRP; el segundo sirve para especificar el flujo al que pertenece el paquete y el tercero, el identificador global del paquete. El funcionamiento general de este protocolo se muestra en los siguientes diagramas.

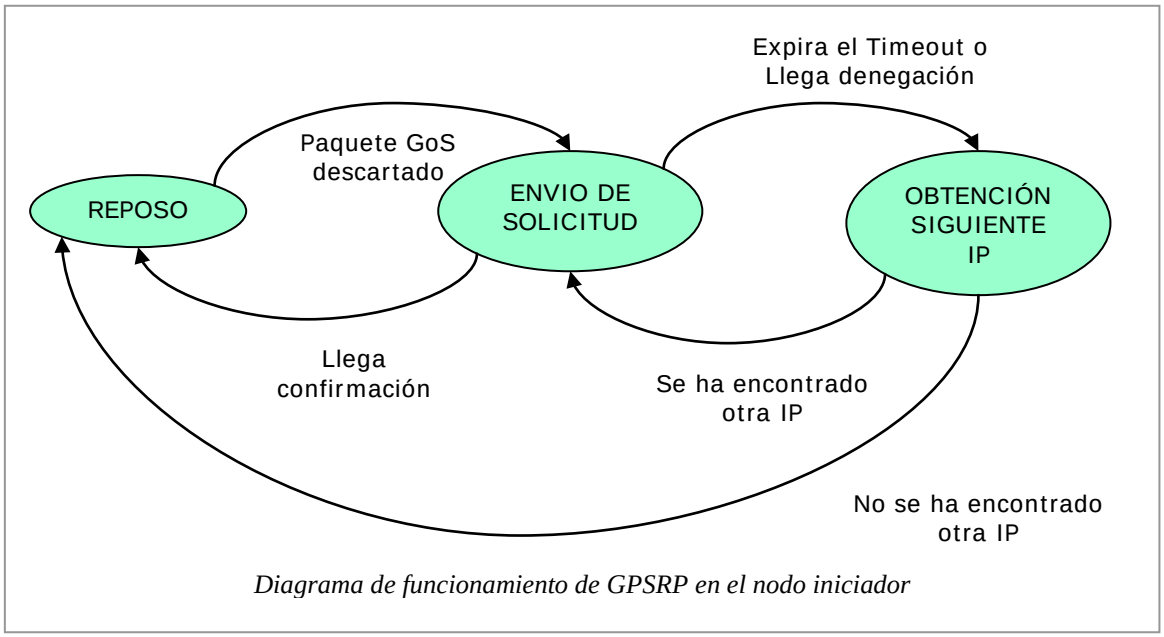


Diagrama de funcionamiento de GPSRP en el nodo iniciador

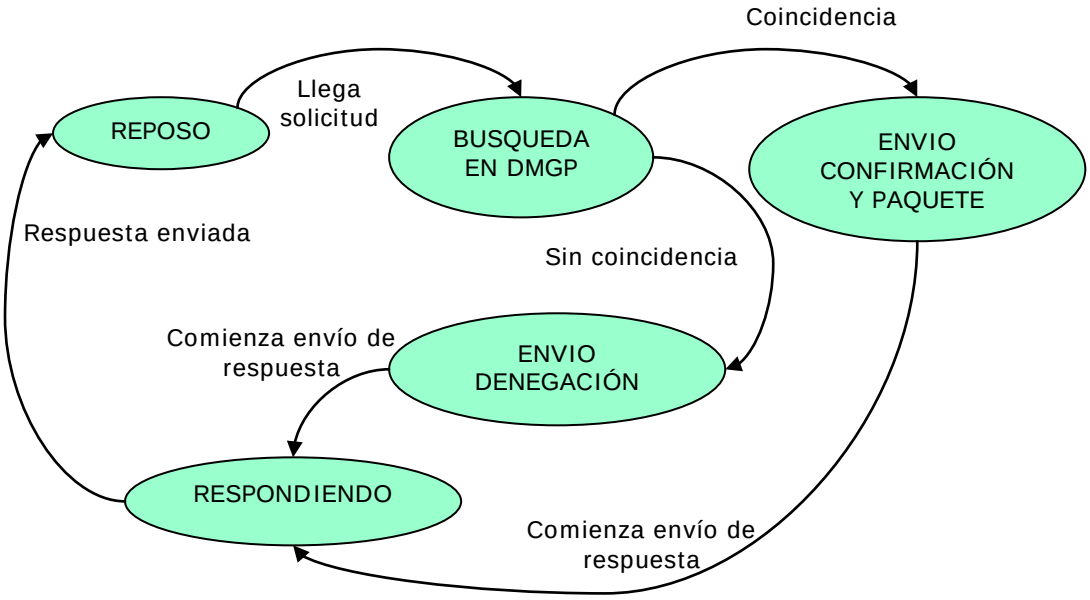


Diagrama de funcionamiento de GPSRP en el nodo no iniciador

La explicación de los diagramas anteriores es la siguiente:

Cuando en un nodo activo se descarta un paquete marcado con GoS, éste tiene la información necesaria para poder solicitar su retransmisión (gracias al algoritmo EPC). Obtiene el campo opciones de la cabecera IP del paquete y comprueba si ha pasado por algún nodo activo antes de llegar al nodo actual. Si no es así, no se intenta recuperar el paquete. Si ha atravesado nodos activos, obtiene la IP del último nodo activo atravesado (será el mas cercano al actual) y le envía una petición de retransmisión del paquete perdido. Si tras un tiempo el nodo restante no contesta o contesta con una negación (ya no tiene el paquete), obtiene la siguiente IP y reintenta la operación y así sucesivamente hasta que no queden IP de nodos activos a los que solicitar la retransmisión. Si en algún momento el nodo remoto responde con una afirmación, esto significa que además de la afirmación, el nodo remoto ha procedido a la retransmisión y por tanto el proceso termina. Si el paquete retransmitido se volviese a descartar, el proceso comenzaría de nuevo, como si hubiese sido un paquete que llegase por primera vez al nodo.

Del lado opuesto, cuando el nodo remoto obtiene una petición de retransmisión, en la misma petición se identifica el flujo y el paquete requerido. Comprueba si lo tiene en la

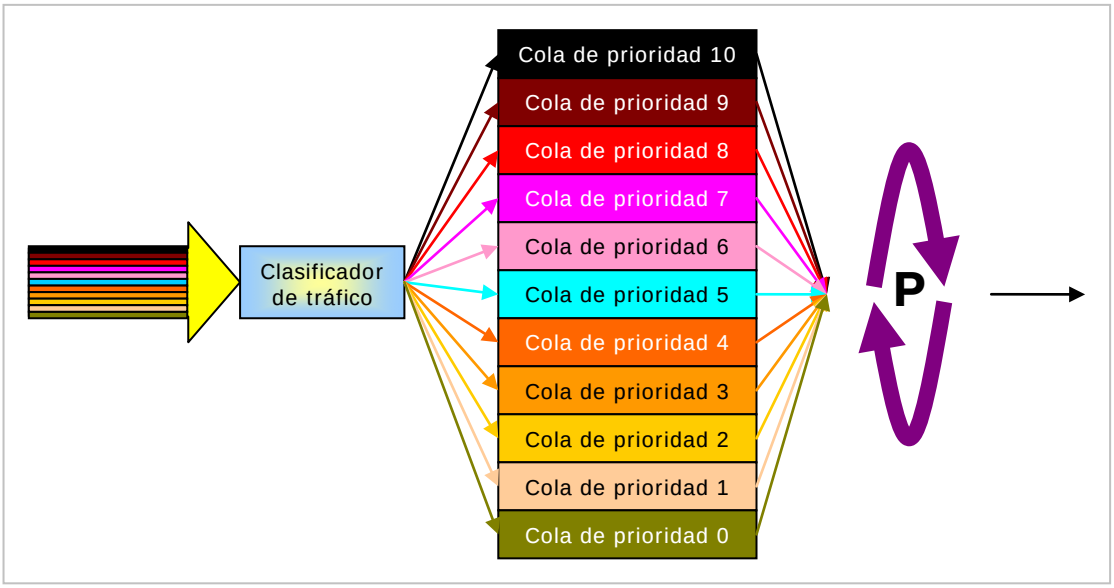
DMGP y si es así lo indica enviando una confirmación, seguida de la retransmisión del paquete MPLS solicitado. Si no el paquete está en la DMGP, contesta con una denegación.

De este modo, con un protocolo sencillo se consigue mantener un sistema de recuperaciones coherente. Se puede dar el caso de que una expiración del *Timeout* venga precedida por la contestación afirmativa (aunque fuera de tiempo) de retransmisión y podría ocurrir, aunque de forma muy ocasional, que más de un nodo retransmitiese el mismo paquete. Los extremos se encargarán de descartar el paquete duplicado, pero en cualquier caso la recuperación se habrá llevado a efecto.

3.2.3.7. Arquitectura de los búferes activos

Ante el funcionamiento del protocolo GPSRP hay un problema; en los búferes generalmente se almacena el tráfico recibido en colas FIFO lo que, en ocasiones de congestión, como es el caso que nos ocupa, pude producir que aunque una solicitud de retransmisión se envíe lo más rápidamente posible, no sea realmente procesada por el nodo destino de la misma hasta mucho tiempo después, cuando le llegue el turno. Para evitar este problema, este proyecto propone una arquitectura para los puertos de los nodos activos, que se han denominado “puertos activos” y que se explica a continuación.

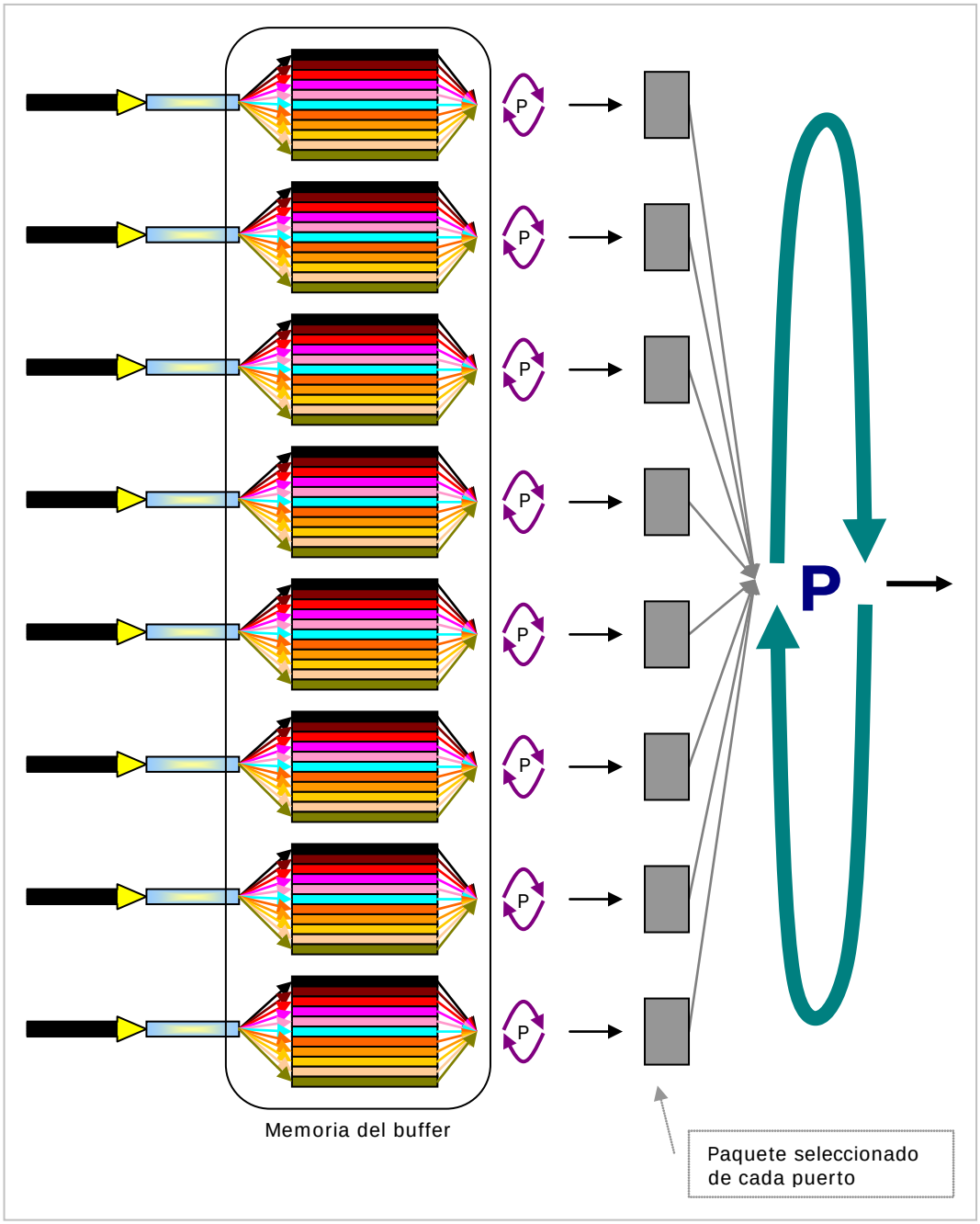
El primer paso de un nodo activo en la búsqueda de la garantía de servicio se da en el buffer de sus puertos. Mientras que en nodos tradicionales los paquetes no son identificados y priorizados sino que generalmente se tratan en el orden de llegada (en Open SimMPLS 1.0 es así), en los nodos activos los puertos identifican el tipo de tráfico y asignan prioridades para su tratamiento. La siguiente figura muestra esta arquitectura para un solo puerto:



Al puerto llega tráfico de muy diversos tipos; en el momento de la llegada un clasificador se encarga de averiguar qué tipo de tráfico es para saber que nivel de prioridad debe tener dentro del puerto. Según esta prioridad, cada paquete irá a una cola distinta (hay 11 colas). Las colas son lógicas, no físicas, y por tanto pueden tener distintos tamaño, incluso pueden llegar a ocupar la totalidad del buffer del nodo al que pertenece el puerto, si así se requiriese. Cada cola contendrá pues, tráfico con la misma prioridad. La política de gestión de cada una de estas colas es FIFO, o sea, el tráfico es ordenado en el puerto primero por prioridades y posteriormente paquetes de la misma prioridad son ordenados por el momento de llegada.

Un algoritmo de Round Robin con prioridades lee en turno circular y siempre que sea posible, más paquetes de la cola con prioridad 10 y menos paquetes a medida que la prioridad de la cola decae. En condiciones donde las colas tengan tráfico suficiente, el algoritmo de selección de paquetes leerá 11 paquetes de la cola con prioridad 10, 10 de la cola con prioridad 9, 9 de la cola con prioridad 8, y así hasta leer sólo uno de la cola con prioridad 0. Tras esto, el ciclo se repite. De este modo se está priorizando claramente el tráfico de las colas con prioridad más alta que el de las colas con menor prioridad; o sea, se procesa más cantidad de paquetes más prioritarios que de los menos prioritarios.

Pero este mecanismo es para un solo puerto del nodo. El esquema general de un nodo con, por ejemplo, 8 nodos, sería como muestra la figura siguiente.



Es decir, cada puerto siempre tiene las 11 colas lógicas de prioridad y tiene un paquete preseleccionado para ser leído. Como resultado, en el ejemplo, siempre (si hay tráfico suficiente) habrá 8 paquetes elegidos, acorde cada uno a las prioridad concreta de los flujos de cada puerto. Entonces, entre esos ocho paquetes disponibles, uno por puerto, un algoritmo de Round Robin, por prioridades también, irá leyendo en turno circular once paquetes de prioridad 10, 10 paquetes de prioridad 9, 9 paquetes de prioridad 8, etcétera. Así, de forma global se estarán atendiendo los paquetes más prioritarios con mayor atención que los paquetes menos prioritarios.

Obsérvese que este modo de trabajo no implica que los paquetes con más prioridad serán descartados con una probabilidad más baja. Esta técnica no afecta a la probabilidad de que un paquete sea descartado o no sino en la probabilidad de que, una vez que el paquete ha ingresado en el búfer, sea atendido con mayor rapidez (se atienden en mayor proporción).

En cualquier caso, el doble algoritmo Round Robin con prioridades atiende los paquetes de cada tipo de una forma tal que se puede asegurar que tras un número concreto de paquetes, el paquete recibido será atendido. Esto supone una cota a la relegación del tráfico en los puertos.

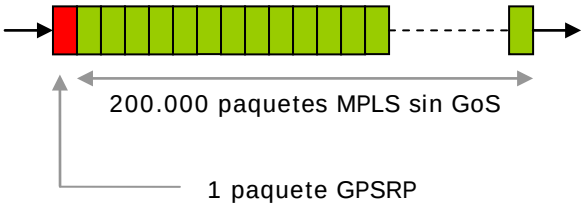
En un ciclo completo de Round Robin con prioridades, con todos los tráficos existentes y con el número suficiente de cada uno de ellos, es decir, en la peor situación, el Round Robin por prioridades atenderá paquetes de la siguiente forma:

Tipo de tráfico	Número de paquetes
TLDP	11
GPRSP	10
RLPRP	9
GoS 3 + LSP	8
GoS 3	7
GoS 2 + LSP	6
GoS 2	5
GoS 1 + LSP	4
GoS 1	3
GoS 0 + LSP	2
Sin GoS	1
Paquetes totales	66 paquetes

Por lo que, independientemente del número de paquetes que haya en el búfer el nodo o del tipo de tráfico de que se trate, al menos un paquete de cada tipo se conmutará cada 66 paquetes; en general no será normal que haya todo tipo de tráficos y con cantidad suficiente de cada uno de ellos por lo que esta cota la mayor parte del tiempo será menor.

Realmente el algoritmo que trabaja en esta propuesta en los puertos activos es una variante de SFQ (Stochastic Fair Queuing), de *MacKenney* con algunos tintes de CBQ (Class Based Queuing).

Supongamos que tenemos el siguiente búfer en un puerto tradicional FIFO.



Hay 200.000 paquetes de MPLS tradicional (sin GoS) en el búfer cuando de repente llega un paquete GPSRP solicitando una retransmisión. Tendrá que esperar a que se conmuten 200.000 paquetes antes de que le llegue el turno; en ese momento será demasiado tarde para realizar la recuperación de paquetes. Con los puertos activos, hubiese tenido que esperar a que se conmutara un solo paquete MPLS sin GoS y se le hubiese cedido el turno al paquete GPSRP; esta diferencia de tiempo es lo que hace que esta propuesta de puerto activo sea ideal para complementar al protocolo GPSRP.

Las prioridades definidas en los puertos son como se muestran a continuación, dependiendo del tipo de tráfico existente en el búfer.

Prioridad 10 →	Paquete TLDP
Prioridad 9 →	Paquete GPSRP
Prioridad 8 →	Paquete RLPRP
Prioridad 7 →	Paquete con nivel GoS 3 y LDP de respaldo activado
Prioridad 6 →	Paquete con nivel GoS 3 y LDP de respaldo desactivado
Prioridad 5 →	Paquete con nivel GoS 2 y LDP de respaldo activado
Prioridad 4 →	Paquete con nivel GoS 2 y LDP de respaldo desactivado
Prioridad 3 →	Paquete con nivel GoS 1 y LDP de respaldo activado
Prioridad 2 →	Paquete con nivel GoS 1 y LDP de respaldo desactivado
Prioridad 1 →	Paquete sin GoS pero LDP de respaldo activado
Prioridad 0 →	Paquete tradicional

Las prioridades las hemos asignado de acuerdo a la importancia que supondría la pérdida de un paquete de dicho tipo para el global de la comunicación o para el correcto funcionamiento de la red. **Deben entenderse las prioridades como la probabilidad de que cuando un nodo esté saturado, un paquete sea procesado**. Con este esquema de prioridades, es mucho más probable que en caso de congestión del nodo, si un paquete se tiene que ser procesado, sea un paquete con mayor prioridad, mientras que si hay un

paquete con menor prioridad en el buffer, será relegado, aunque no indefinidamente a favor de paquetes más prioritarios.

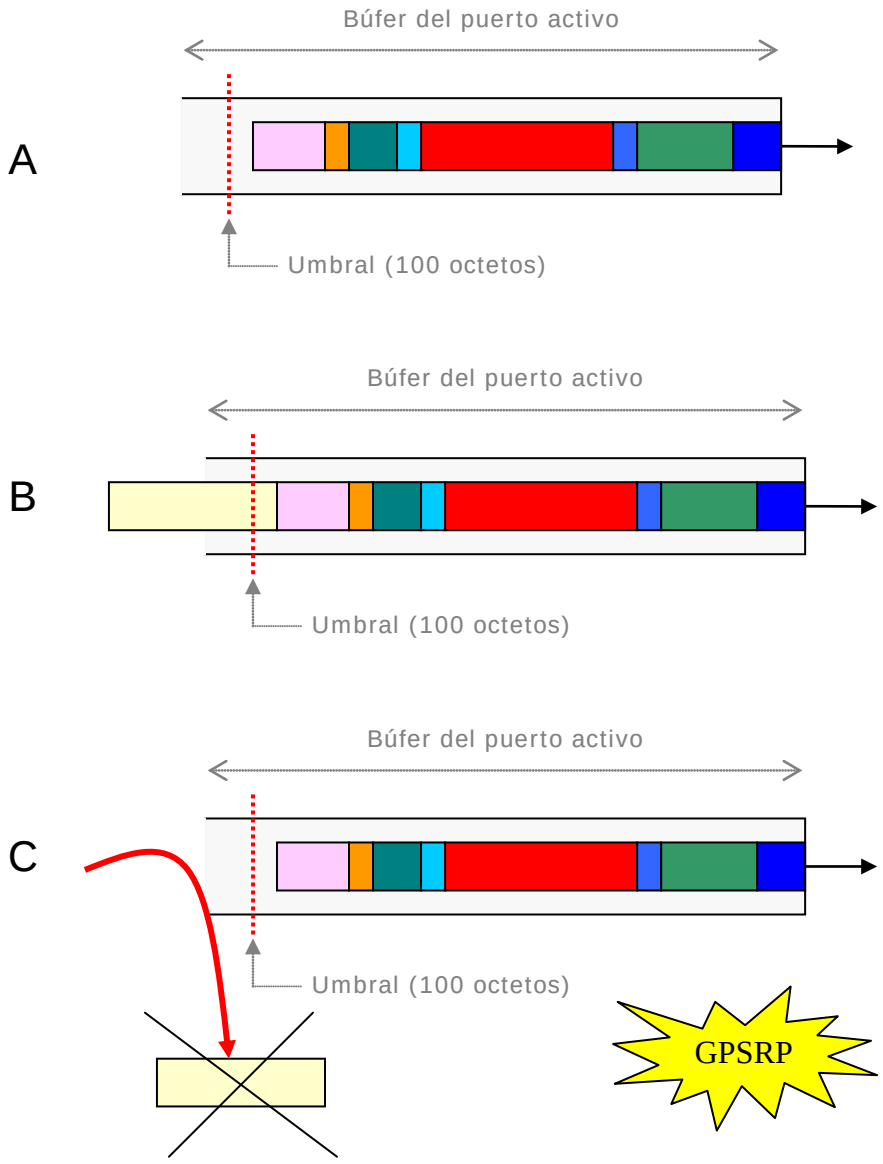
Este esquema de prioridades favorece como puede verse a aquellos paquetes marcados con niveles de garantía de servicio frente a los que no y a los que más nivel tienen frente a los que tienen menos.

> *¡Nota! En este PFC no existen los paquetes RLPRP. La técnica o protocolo RLPRP se ha constituido como una técnica intrínseca, interna y exclusiva de los nodos activos y no como un protocolo de comunicaciones. Sin embargo para un futuro está planeado implementar RLPRP como un protocolo de comunicaciones entre nodos con adyacencia activa y para cuando llegue ese momento, los puertos activos ya implementan la prioridad en los búferes para ese tipo de paquetes.*

3.2.3.8. Gestión de búferes en los puertos (EPCD)

Ya se ha definido casi por completo el modo en que GPSRP acomete la recuperación de paquetes con GoS descartados; sin embargo, **para poder solicitar la retransmisión de un paquete** descartado por congestión en los búferes de un nodo activo, **es necesario por recuperar al menos la cabecera IPv4 de dicho paquete** donde se almacenan los datos identificativos del mismo (IP-identificador) y el conjunto de los últimos nodos activos que se han atravesado y que será a los que tendremos que solicitar dicha retransmisión.

Para todo esto necesitamos un algoritmo de gestión en los búferes que permita que al caer el paquete al menos parte del mismo se haya podido capturar, la parte que necesitamos. Hemos pensado en un algoritmo que mantenga un umbral siempre disponible de tal forma que todo paquete recibido que supere el límite de ese umbral será descartado pero dicho algoritmo permita "morder" parte de él. Algoritmos de este tipo hay y nos basaremos en algunos de ellos como *RED (Random Early Discard)* o *EPDR (Early Packet Discard and Relay)*. **La propuesta de este PFC al respecto es el algoritmo *EPCD (Early Packet Catch and Discard)*** o Captura y Descarte Anticipado de Paquetes. La siguiente secuencia de figuras ilustra el procedimiento, que entronca con el protocolo GPSRP definido páginas atrás.

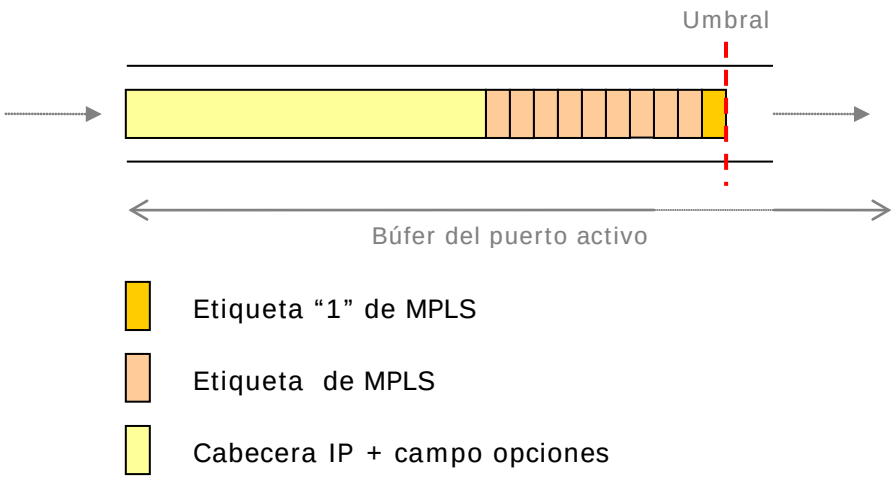


En la figura (A) se muestra un búfer que comienza a estar lleno. De repente (B) llega un paquete que produce que el búfer se termine de congestionar, sobrepasando el umbral permitido por EPCD. Entonces EPCD captura toda la información posible sobre ese paquete que, al menos, serán los 100 octetos que el umbral permite, quizás más, y descarta el paquete (C). Es esperable que dentro de ese fragmento recogido por EPCD se haya conseguido capturar todas las etiquetas MPLS además de la cabecera IP con su campo opciones. Si con esos datos EPCD logra determinar que el paquete está marcado con GoS y consigue obtener el campo opciones de IP de ese paquete, notificará a GPSRP para que proceda a la recuperación de ese paquete. Si no consiguiese averiguar algo sobre el paquete o el paquete no estuviese marcado con GoS, se obviaría cualquier intento de recuperación,

pero se habría preservado libre el umbral necesario para poder capturar esta información si otro paquete es descartado.

¿Por qué 100 octetos de umbral? EPCD necesita la cabecera IP del paquete con su campo opciones se es que se ha utilizado. Además, deberá capturar todas las etiquetas que se encuentren con anterioridad a ésta. La cabecera IP al completo en el peor de los casos ocupará 60 octetos (20 fijos + 40 máximos del campo opciones) con lo que, en un espacio de 100 octetos, que es lo que define el umbral, quedarían 40 octetos que podrían ser ocupados por etiquetas MPLS. Esto supone, a 4 octetos por cabecera MPLS, 10 cabeceras MPLS, lo que equivaldría a 9 dominios concéntricos y una cabecera con etiqueta 1 (paquete con GoS).

Estamos hablando de una jerarquía de dominios MPLS de hasta 9 dominios concéntricos, lo cual es muy infrecuente (lo normal son 3 ó 4 como mucho); en casi el 100% de los caso, con esos 100 octetos hay espacio suficiente para poder decodificar qué paquete se ha descartado y obtener los datos necesarios para proceder a su recuperación vía GPSRP.



En la figura anterior se observa el trozo de paquete capturado por EPCD gracias al umbral de 100 octetos en el caso más desfavorable descrito en líneas atrás.

3.2.3.9. Algoritmo de encaminamiento (RABAN)

El primer paso para que en una red la pérdida de paquetes sea baja es evitar que se produzcan; este punto de vista es casi tan importante como conseguir su recuperación una vez que se han perdido. Los protocolos de encaminamiento generalmente son estáticos y siguen una métrica definida para cada enlace de tal forma que encaminan todo el tráfico que se dirige a un mismo destino por el mismo camino, sin tener en cuenta más factores. Esto acaba en muchas ocasiones desequilibrando la red, produciendo congestiones en los nodos y la consecuente pérdida de paquetes. En muchas ocasiones seleccionar un camino basándose en una métrica estática definida tiene el inconveniente de que lo que en principio parece la mejor decisión, acaba provocando descarte de paquetes. Sin embargo en muchas de estas ocasiones existían caminos alternativos, con un poco más de retardo quizás, pero sin problemas de congestiones. Una recuperación local en casi todas las ocasiones es más rápida que una recuperación extremo a extremo; pero una no recuperación a costa de ir por un camino un poco más lento es con casi toda seguridad, más rápida que una recuperación local.

Por tanto, un buen algoritmo de encaminamiento que permita balancear el tráfico no dirigiendo todo el tráfico por el mismo área de la red sino distribuyendo los flujos por las partes infrautilizadas, provocaría menos congestiones y pérdidas de paquetes; con respecto a los flujos marcados con GoS, haría más difícil que varios flujos con GoS coincidiesen en el mismo nodo activo, aumentando la capacidad de éste de atender los requerimientos de GoS de una forma mejor.

Este PFC propone un algoritmo de encaminamiento llamado ***RABAN (Routing Algorithm for Balanced Active Networks)*** o Protocolo de Encaminamiento para Redes Activas Balanceadas. **Este algoritmo consistirá en un Floyd donde los pesos de los caminos serán un cálculo dinámico ponderado** donde influirán:

- El retardo del enlace.
- El número de LSP que sostiene el enlace.
- El número de LSP's de respaldo que están abiertos para ese enlace.
- El estado de saturación de los dos nodos que une el enlace, así como el número de entradas de su matriz de conmutación/encaminamiento.
- Una estimación de los paquetes en tránsito que circulan por el enlace.

La ponderación de dichos valores se ha llevado a cabo mediante multitud de pruebas bajo muchas circunstancias distintas en el simulador Open SimMPLS 1.0 y el resultado final queda como se muestra a continuación.

$$PesoEnlace = (0'5 \cdot PesoEstatico) + (0'5 \cdot PesoDinamico)$$

$$PesoEstatico = delay$$

$$PesoDinamico = PesoExtremo1 + PesoExtremo2 + PesoLSP + PesoLSPBackup + PesoOnFly$$

$$PesoExtremo1 = (delay \cdot 0'1) \cdot \left[(0'7 \cdot congestionBufferE1) + (0'3 \cdot 10 \cdot NumEntradasMCE1)\right]$$

$$PesoExtremo2 = (delay \cdot 0'1) \cdot \left[(0'7 \cdot congestionBufferE2) + (0'3 \cdot 10 \cdot NumEntradasMCE2)\right]$$

$$PesoLSP = (delay \cdot 0'05) \cdot NumLSP$$

$$PesoLSPBackup = (delay \cdot 0'05) \cdot NumLSPBackup$$

$$PesoOnFly = (delay \cdot 0'1) \cdot PaquetesEnVueloEstimados$$

Donde:

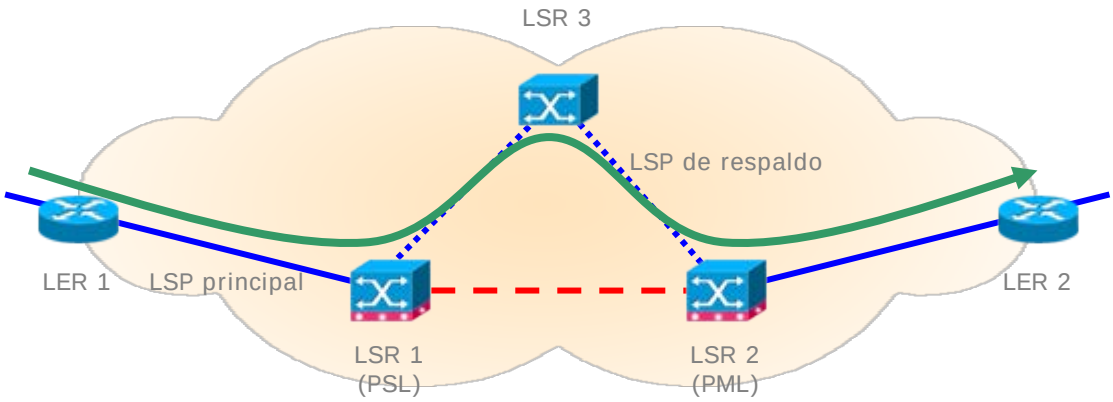
- **Delay:** es el retardo del enlace.
- **CongestionBufferE1:** es el porcentaje de congestión del buffer del nodo que es extremo 1 del enlace.
- **NumEntradasMCE1:** es el número de entradas que están definidas en la matriz de conmutación/encaminamiento del nodo que es extremo 1 del enlace.
- **CongestionBufferE2:** es el porcentaje de congestión del buffer del nodo que es extremo 2 del enlace.
- **NumEntradasMCE2:** es el número de entradas que están definidas en la matriz de conmutación/encaminamiento del nodo que es extremo 2 del enlace.
- **NumLSP:** es el número de LSP que están establecidos sobre el enlace.
- **NumLSPBackup:** es el número de LSP de respaldo que hay establecidos sobre el enlace.
- **PaquetesEnVueloEstimado:** es la estimación del número de paquetes que hay en tránsito por el enlace.

3.2.3.10. Técnica de creación de LSP de respaldo (RLPRP)

Cuando los paquetes de un flujo están marcados para requerir un LSP de respaldo, significa que desean un mecanismo para que cuando un LSP por el que viajan resulte dañado, se pierda el menor tráfico posible y se minimice el tiempo de reestructuración normal de los caminos virtuales para llevar el tráfico al destino.

Existen muchas y muy buenas propuestas en este sentido la mayor parte de ellas definen el modo en que un LER de entrada precalcula o establece al unísono el LSP principal para un flujo y un LSP de respaldo, desde el origen al destino. Asimismo, además del cálculo o establecimiento previo del LSP de respaldo, existen diversas propuestas sobre cómo gestionar el tráfico que en ese momento se encuentra "en vuelo" por los enlaces por los que fluye el LSP.

Una de las propuestas más elaboradas que he podido consultar se encuentra expresada brevemente en un artículo titulado *"QoS Online Routing and MPLS Multilevel Protection: A Survey"* de José L. Marzo y Eusebi Calle, de la Universidad de Gerona. En este artículo se comenta, entre otras cosas, la idea de recuperación local de LSP con la implicación de unos nodos especiales (PML y PSL) capaces crear LSP de respaldo y de conmutar el flujo de datos de un LSP al LSP de respaldo local que hace un *by-pass* evitando el enlace caído, como se muestra en la siguiente figura, reproducida directamente de dicho artículo.



A fin de cuentas es lógico que tanto el origen como el destino del LSP de respaldo tengan que tener ciertas capacidades especiales porque si no, por definición, un LSR no tiene la posibilidad de iniciar la creación de un LSP de de ser extremo de la misma.

Esta propuesta está especificada para cumplir con requisitos de QoS, para trabajar con *DiffServ*, y con algoritmos de encaminamientos basados en restricciones (ancho de banda, menor interferencia, etcétera). Además difiere del planteamiento que se desea plasmar en este proyecto; la idea de técnica activa donde sea el propio paquete el que sugiere (mediante la marca de GoS) cómo debe ser tratado por el nodo. Incluso por su profundidad, esta propuesta excede los límites de un Proyecto Final de Carrera.

Sin embargo, el planteamiento de un sistema de recuperación local de LSP es precisamente lo que deseamos para completar la propuesta de este proyecto sobre GoS en entornos locales; y parece claro que deben ser los nodos activos los que de alguna manera implementen esa capacidad. Y debe ser todo transparente a MPLS.

Para dar solución a este problemas en este proyecto se ha desarrollado una propuesta llamada ***RLPRP (Resilient Local Path Recovery Protocol)***, o Protocolo Flexible de Recuperación Local de Caminos, cuyo funcionamiento se explica a continuación.

Partimos de la base de que el paquete MPLS que venga marcado desde el origen con requerimientos de LSP de respaldo es quien debe activar los mecanismos dentro de los nodos activos para que lleven a cabo su creación. Además, posibles modificaciones en el protocolo de señalización (TLDP en este caso; LDP o RSVP genéricamente) pueden dificultar la compatibilidad con los nodos que no sean activos, lo que deseamos evitar. Y por último, modificaciones en estos protocolos estarían haciendo que los nodos estableciesen LSP de respaldo como respuesta a peticiones y no como respuesta a la recepción de un paquete “activo”.

RLPRP no es un protocolo de comunicaciones sino más bien un conjunto de reglas y procedimientos para la creación y el mantenimiento de LSP de respaldo que se activa con la llegada al nodo activo de paquetes con ese requerimiento. Antes de nada, para poder almacenar información sobre LSP de respaldo, nuestra propuesta añade dos campos más a la matriz de conmutación/encaminamiento formada por ILM (Incoming Label Map), FTN

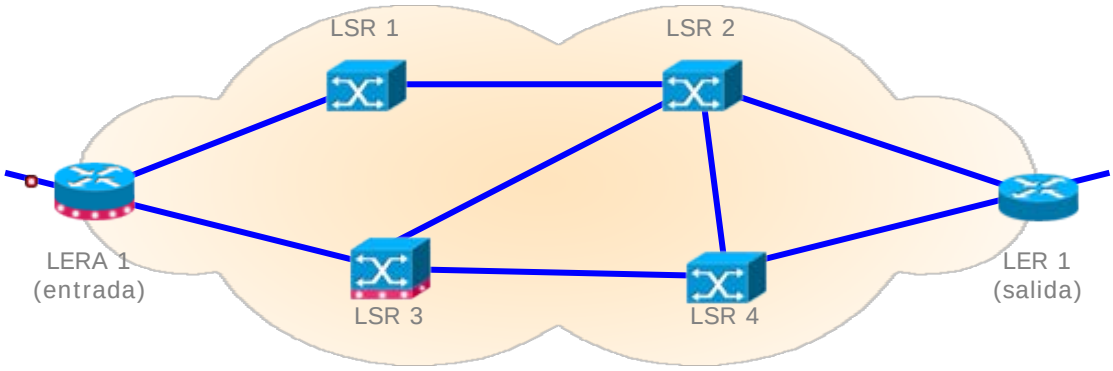
(FEC to Next Hop Label Forwarding Entry) y NHLFE (Next Hop Label Forwarding Entry); concretamente los dos campos irían a NHLFE y son los siguientes:

- **PuertoSalidaBackup:** que almacenaría el puerto al cual hay que dirigir el tráfico si se desea hacer uso del LSP de respaldo.
- **EtiquetaSalidaBackup:** que almacenaría la etiqueta que deben llevar los paquetes cuando deban ir por el LSP de respaldo.

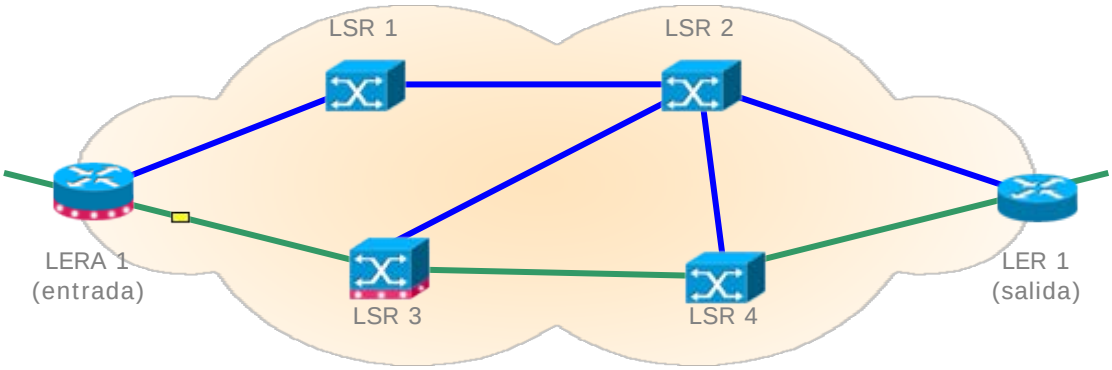
Etiquetas similares se encuentran ya definidas por el IETF para la tabla NHLFE para los LSP tradicionales. Hace falta duplicar esta información para el LSP de respaldo en nuestra propuesta, obligatoriamente; pero sólo para los nodos activos.

Cuando un paquete TLDP o bien tráfico externo al dominio MPLS llega al nodo activo, este establece un LSP como tradicionalmente se ha hecho y el LSP queda establecido desde el origen al destino. Además, con nuestra propuesta, una vez que comienzan a llegar paquetes etiquetados y con GoS al nodo activo, como deben ser analizados aparte (etiqueta = 1) se comprueba si entre sus requerimientos se encuentra la necesidad de LSP de respaldo. Si no es así, se continúa como en MPLS tradicional, pero si es así, se procede a la creación de un LSP de backup, mediante TLDP en nuestro caso, siguiendo el procedimiento natural, pero enviando la petición por otro puerto. Este puerto será calculado por el protocolo de encaminamiento RABAN, que permite obtener el "segundo mejor salto" para llegar al destino.

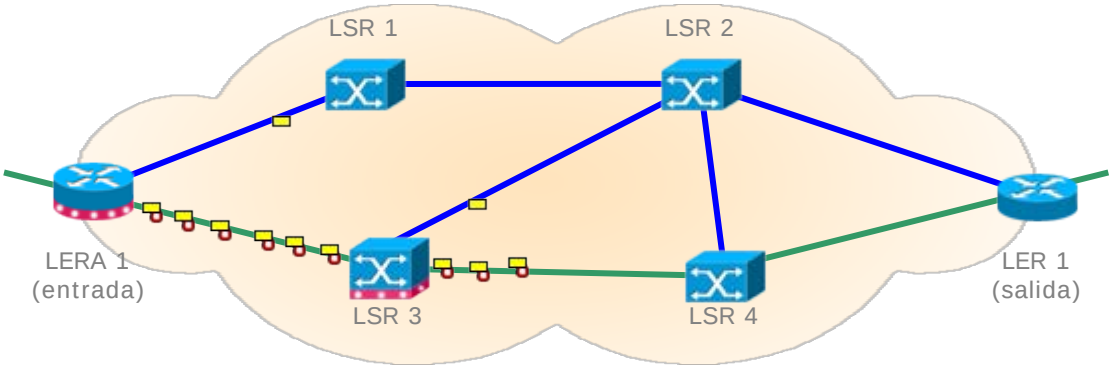
En las siguientes figuras se muestra el proceso completo que se acaba de explicar.



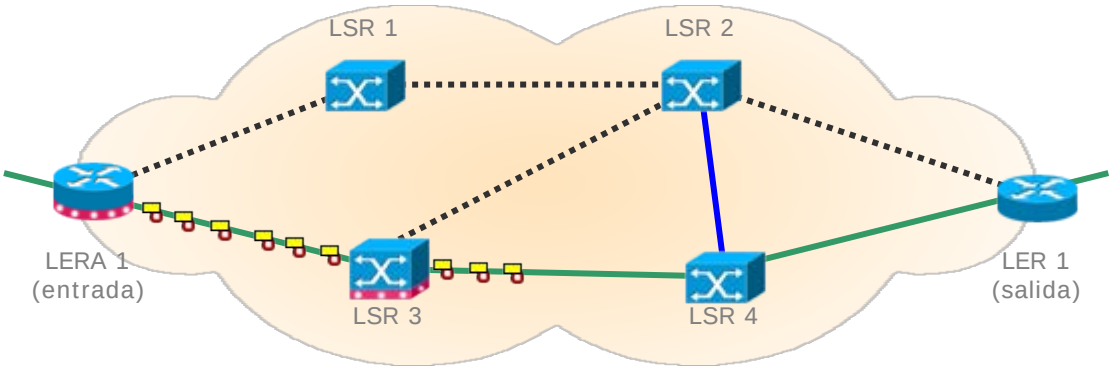
Paso 1: llega paquete con requerimientos de LSP de respaldo



Paso 2: se establece el LSP, de forma tradicional

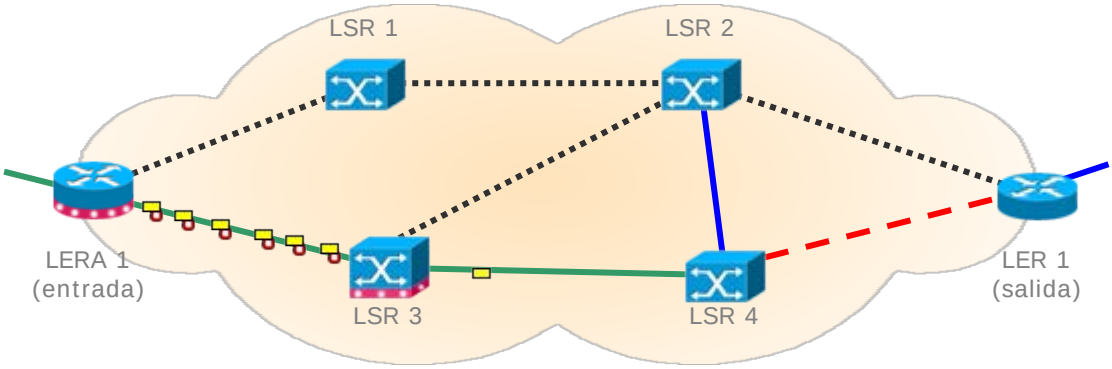


Paso 3: los nodos activos reciben tráfico con requisitos de LSP de respaldo y actúan



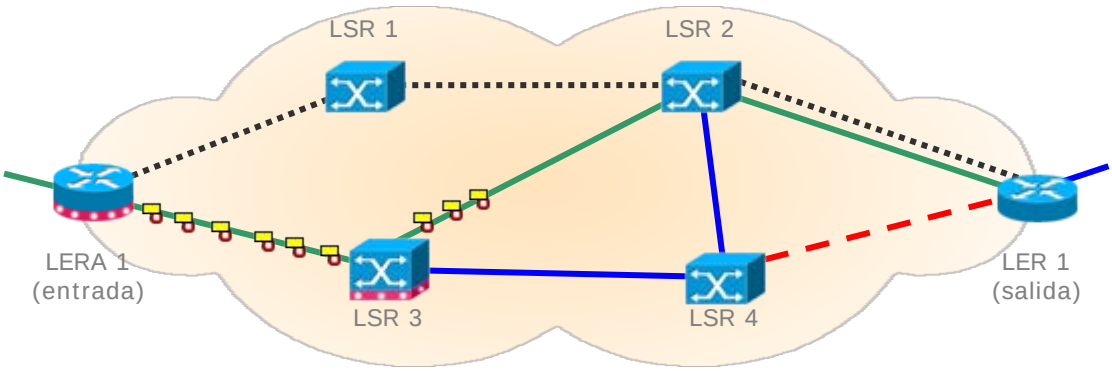
Paso 4: quedan establecidos los LSP de respaldo

Una vez que todos los LSP, el principal y los de respaldo que se hayan podido crear estén establecidos, todo sigue su curso de forma natural. Si un enlace del LSP se rompe, comienza la señalización típica para deshacer los LSP hacia atrás.



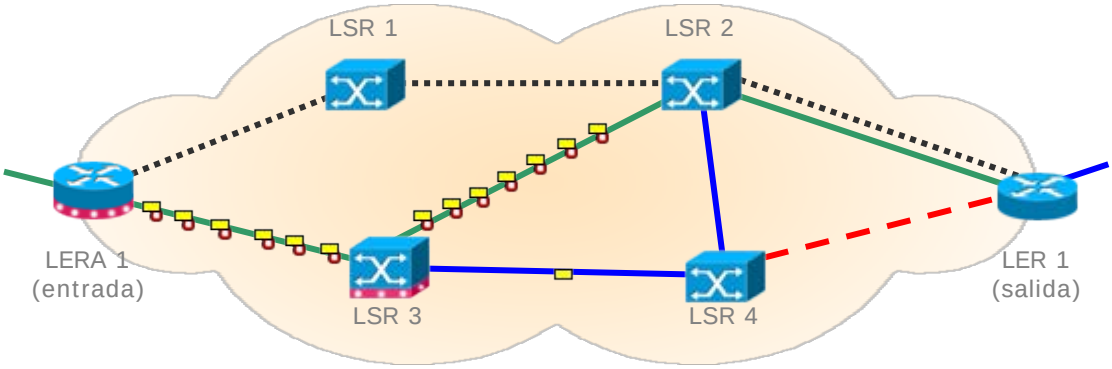
Paso 5: LSP principal cae. Comienza a deshacerse hacia atrás.

Y al llegar a LSR 3, éste comprueba si tiene establecido un LSP de respaldo para poder conmutar. Como es así, modifica la tabla NHLFE correctamente para que se conmute directamente al LSP de respaldo, que ahora será principal. En un LSR normal, la desformalización del LSP hubiese continuado hasta el origen, que habría tenido que emprender la reconstrucción del LSP; mucho tráfico se hubiese perdido y hubiese hecho falta bastante tiempo para rehacer el LSP.



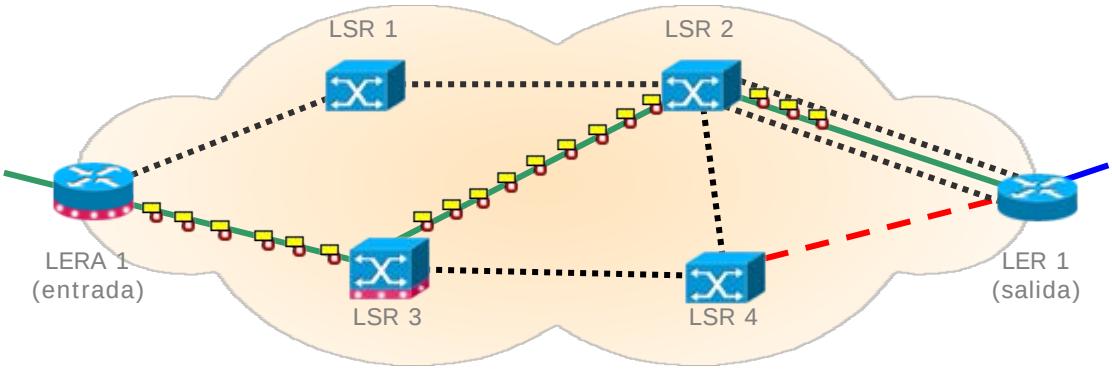
Paso 6: LSR 3 conmuta al LSP de respaldo.

Si el fallo se hubiese producido en el enlace LERA1-LSR3, se habría usado el LSP de respaldo de LERA1-LSR1, de igual forma. En cualquier caso, ahora LSR 3 detecta que le está llegando tráfico con requerimiento de GoS y que él no tiene un LSP de backup establecido así que vuelve a intentar la creación de uno, si es posible.



Paso 7: LSR 3 intenta establecer un nuevo LSP de respaldo.

Por último, se establece de nuevo el LSP de respaldo solicitado por LSR 3 y el tráfico continúa fluyendo como si no hubiese ocurrido nada.



Paso 8: LSR 3 establece por fin un nuevo LSP de backup.

Como vemos la propuesta de RLPRP siempre que pueda recupera en un entorno cercano el LSP. Como principal desventaja de este método está el hecho de que se crean LSP de respaldo en cascada, desde el propio nodo activo que atraviesa el LSP principal, hasta el LER de salida; pero es necesario; así es como se asegura la localidad de la reparación del LSP. Esto sólo ocurre con el LSP principal puesto que es cuando un paquete activo llega al nodo con ese requerimiento, cuando se produce el establecimiento del LSP de respaldo. Y sólo cuando ese LSP de respaldo pasa a ser principal (en una conmutación de LSP) y sus nodos activos reciben tráfico activo, emprenderán la misma acción.

Como comentamos anteriormente, es previsible que el número de flujos con requerimientos de GoS sea mínimo. Sirva como ejemplo el análisis de tráfico semanal de

la red Abilene donde se muestra la distribución de todo el tráfico *DiffServ* que circula por ella.

Tipo de tráfico DiffServ según DSCP (Red Abilene)	Porcentaje de paquetes
Best effort [DSCP=0]	95,01%
Scavenger [DSCP=8]	0,44%
EF [DSCP=46]	0,37%
Otros	4,18%
Total	100%

Es decir, incluso existiendo la posibilidad de seleccionar tráfico con distintas propiedades, más del 95% de los paquetes (según esta estadística, el 93,61% si hablamos en octetos) se configuran como *best effort*. Así que el restante, un 4,99% de los paquetes tienen algún requerimiento especial.

Si trasladamos estas estadísticas a nuestra propuesta, parece razonable pensar que sólo ese 5% requeriría GoS y quizás sólo alguno de ellos necesite LSP de respaldo, lo cual no debería suponer demasiados flujos con este requerimiento. Así pues, no parece, de momento, que el hecho del establecimiento de diversos LSP de respaldo local (segmentos de éste, realmente) para un único LSP principal, tal como propone este documento, pueda ocasionar demasiados problemas; más aún porque en este proyecto no se está hablando en ningún momento de reserva de recursos para dichos LSP de respaldo sino exclusivamente del establecimiento de los mismos a nivel de señalización etiquetas y entradas en las tablas de encaminamiento/conmutación). Es más, ni siquiera circulará tráfico por ellos si no es necesario.

Además, lo más probable es que existan pocos nodos activos en relación con el número total de nodos; y sólo estos pueden emprender la acción de crear un LSP de respaldo.

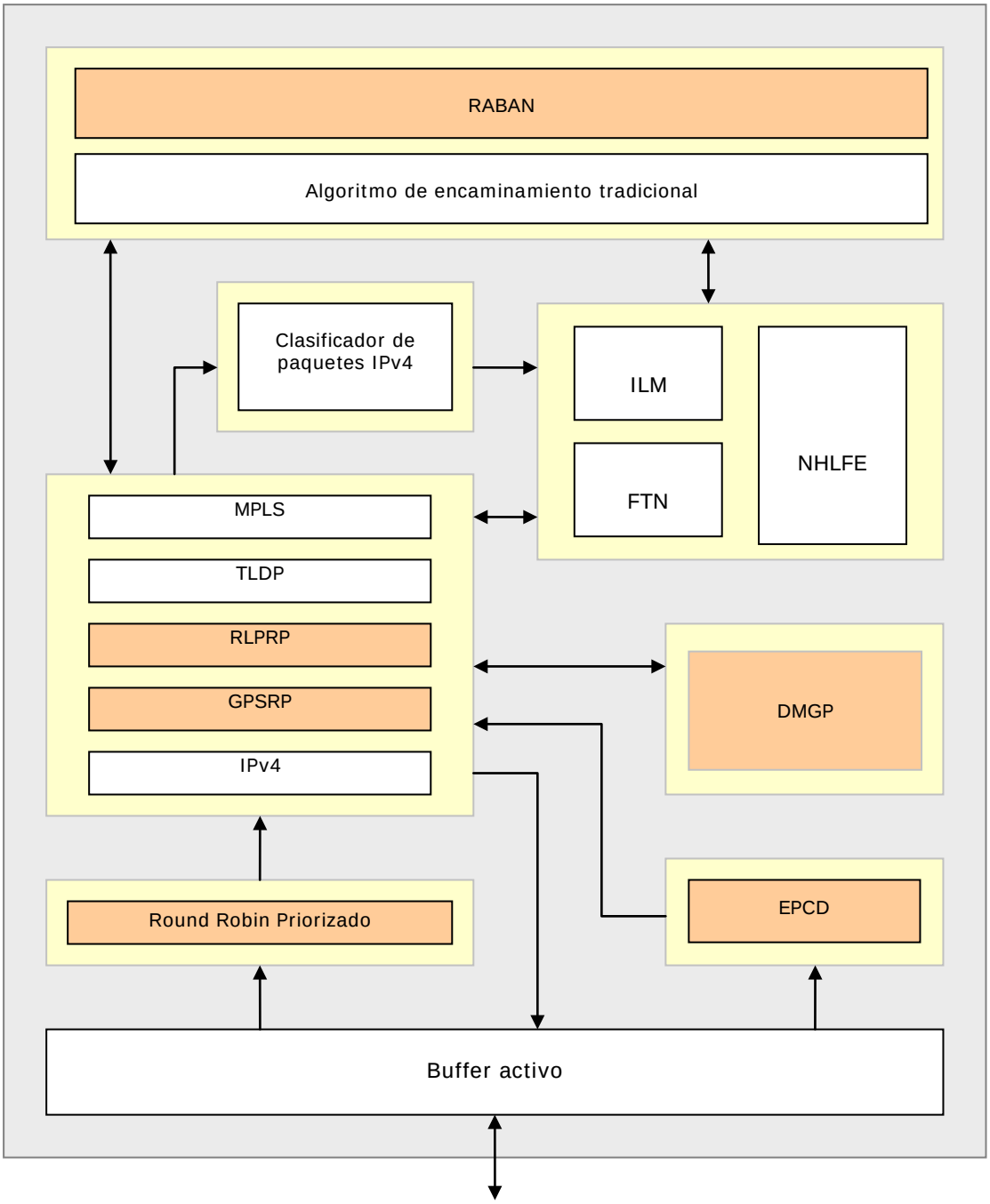
3.2.3.11. Arquitectura general de un nodo activo

De una forma genérica y con un alto nivel de abstracción, podemos definir la arquitectura de un nodo activo, de un LER activo en este caso, como se muestra en la siguiente figura.

En ella, los búferes del nodo aceptan el tráfico entrante, que irremediablemente debe ser servido por el algoritmo Round Robin Priorizado; de este modo nos aseguramos automáticamente que el tráfico más importante será atendido más deprisa, según la escala de prioridad definida páginas atrás, independientemente del momento de su llegada al búfer. Tráfico de igual tipo, será atendido en orden de llegada en el turno circular Round Robin tradicional mientras que no aparezca tráfico más prioritario.

Cuando el paquete es obtenido del búfer, pasa automáticamente a ser atendido por el módulo de protocolo correspondiente. Si el paquete es TLDP, el módulo TLDP lo atenderá y, como es un paquete de señalización, eventualmente modificará los valores de la matriz de conmutación (formada por ILM, FTN y NHLFE) si así es requerido. Si el paquete es GPSRP, encargado de las retransmisiones de paquetes con GoS, el correspondiente módulo lo atenderá y para ello debe acceder a la DMGP donde se almacenan los paquetes que están marcados con algún nivel de GoS. GPSRP entra en funcionamiento también cuando EPCD, que siempre está monitorizando el buffer de recepción, le notifica que se ha descartado un paquete marcado con GoS. En este caso, además de la notificación, entrega al módulo GPSRP la cabecera MPLS/IP del paquete en cuestión para poder realizar la solicitud de retransmisión. Si el paquete es MPLS, el módulo MPLS buscará una entrada en la matriz de conmutación, acorde la etiqueta el protocolo entrante; si no existe, TLDP entrará en juego, solicitando una y el paquete pasará de nuevo al buffer hasta que se tenga respuesta del nodo adyacente. Por último, si el paquete entrante es IPv4, se clasifica se comprueba si hay algún FEC acorde para dicho paquete en la matriz de conmutación. Si no es así se pedirá una etiqueta para este paquete, se depositará de nuevo en al buffer y se esperará hasta tener respuesta.

En cualquier caso, en todo momento los protocolos de *routing* estarán disponibles para los protocolos que lo necesiten, ayudando para formar una matriz de conmutación acorde a la política de encaminamiento y ayudando a los protocolos que viajarán sobre IP a seleccionar la ruta adecuada para dirigirse al destino. El algoritmo de encaminamiento que se utilizará será RABAN, que intentará elegir una ruta no sólo con poco retardo, sino con poco tráfico, con poca congestión, con pocos LSP establecidos, etcétera.



Arquitectura general de un nodo activo con capacidad de encaminamiento

3.2.4. Conclusiones

La investigación llevada a cabo en este proyecto ha intentado en todo momento ser lo más realista posible para conseguir una propuesta que pudiese ser implementada en nodos reales. No sería una aspiración descabellada dado que se ha intentado también que esté

íntimamente ligada a las redes activas cuyo principal objetivo, como se ha comentado en este mismo documento, es hacer las redes mucho más flexibles, facilitando el despliegue de nuevos servicios y protocolos basándose, eso si, en que la información de cómo debe ser tratado un paquete debe viajar con el propio paquete. Por otro lado, se ha conseguido implementar un conjunto de técnicas, protocolos y estructuras que en conjunto permiten que el tratamiento del tráfico marcado con GoS sea privilegiado, más cuanto más nivel de GoS se haya asignado, que era el objetivo inicial del proyecto.

Debido a lo anterior creo que al menos el desarrollo teórico de este proyecto ha sido satisfactorio; en los siguientes apartados, con la construcción de un simulador específico, intentaré constatar si también en la práctica resulta ser una buena propuesta.

3.2.5. Ampliaciones futuras

Este proyecto constituye un acercamiento de MPLS, las redes activos y el concepto de Garantía de Servicio (GoS). En futuros trabajos sería interesante:

- Implementar RLPRP además de cómo una técnica interna al nodo activo, como un protocolo de comunicaciones; incorporar el concepto de "adyacencia activa" para la resolución de los LSP de respaldo en un entorno aún más local de lo que propone este proyecto.
- Mejorar el Round Robin por Prioridades que trabaja en los puertos del nodo activo, para que no reparta el tiempo de conmutación para conseguir el mismo número de paquetes conmutados por puerto, sino el mismo número de octetos; con tráfico de tamaño variable, sería lo más justo. Realmente el algoritmo que trabaja en esta propuesta en los puertos activos es una variante de SFQ (Stochastic Fair Queuing), de *MacKenney* con tintes de CBQ (Class Based Queuing). En su lugar se debería implementar un algoritmo que fuera una mezcla de DRR (Deficit Round Robin) y CBQ (Class Based Queuing) ya que DRR tiene una complejidad constante y además realiza la repartición por octetos y no por paquete.
- Es necesario portar todas las propuestas a IPv6, aunque realmente el único problema existente es encontrar en la cabecera IPv6 algún espacio para almacenar

lo que almacenamos en el campo “opciones” de IPv4, teniendo en cuenta además el tamaño de las nuevas direcciones.

- Ir abandonando TLDP (LDP, es lo mismo) y adoptando en su lugar RSVP puesto que durante el transcurso de este PFC la comunidad científica sí lo ha ido haciendo (y continúan).
- Entroncar los grados de GoS con la construcción de LSP con reserva de recursos para que además de conseguir LSP de respaldo, conseguir que sean LSP de respaldo de distinta “categoría” para cada uno de los tipos de tráfico según su grado de GoS.

3.3. Simulador Open SimMPLS

La propuesta teórica que se presenta en este proyecto para dar soporte de Garantía de Servicio (GoS) a flujos privilegiados de información requiere de pruebas empíricas para ratificarse como una propuesta sólida y válida. Sin embargo al tratarse de una tecnología desarrollada exclusivamente en el ámbito ese este PFC, no hay herramientas que permitan la simulación, por lo que es necesario crear un simulador, que en este caso se llamará Open SimMPLS 1.0.

3.3.1. Especificación

Una vez definido el objetivo del desarrollo software que será necesario emprender y tras múltiples reuniones con el director del proyecto, al que se entiende como “potencial cliente” al que va dirigido el software, se han recopilado multitud de requisitos que el simulador deberá cumplir:

- **Deberá funcionar como un simulador de MPLS tradicional**. Así se podrán comparar las características nuevas propias de la arquitectura propuesta, con las de MPLS tradicional.
- **Deberá implementar todas y cada unas de las propuestas desarrolladas en este proyecto** para la consecución del objetivo de poder ofrecer GoS a flujos privilegiados. Por la misma razón que antes.

- **Deberá ser lo suficientemente visual para poder ser utilizado como herramienta docente**. Pues es previsible su utilización en asignaturas del área de ingeniería telemática y/o en cursos de doctorado.
- **Deberá ser multilingüe.** Al menos debería estar traducido al inglés. Esto es así porque es previsible que pueda resultar de utilidad a futuras investigaciones nacionales o internacionales. El idioma inglés es el universal en el mundo de la investigación.
- **Deberá ser portable.** Por el mismo motivo que el punto anterior; la cantidad de sistemas en los que los investigadores y el personal docente de las universidades trabaja es elevada. El simulador debería estar disponible para la mayor parte de ellas; al menos Windows y Linux.
- **Deberá contar con un proceso sencillo de instalación y configuración.** El simulador se debe poder usar esporádicamente para la obtención de resultados en pruebas concretas. Si necesitase un laborioso proceso de configuración e instalación sería un problema.
- **Deberá facilitar la obtención de gráficas** de los resultados de las pruebas que se desean realizar en él.
- **Deberá permitir la configuración de los parámetros** importantes tanto MPLS como de las propuestas definidas en este proyecto.
- **Deberá estar bien documentado** puesto que se prevé su uso posterior a la presentación del simulador/proyecto y su modificación o ampliación en trabajos posteriores por parte de personas no involucradas en su desarrollo inicial.
- **Sería recomendable que contase con una web** desde donde se pudiese descargar tanto el trabajo teórico como el simulador y su documentación, para dar difusión y para facilitar la retroalimentación asociada al mismo.
- **Como requisito personal (compartido por el director del PFC), deberá ser software libre;** además toda la documentación del PFC debe ser documentación libre, aunque se respeten ciertos plazos para la publicación y presentación del proyecto, el simulador y sus conclusiones en los foros nacionales e internacionales que se consideren oportunos.

La idea preliminar para el desarrollo del simulador es la utilización de librerías Open Source que permita minimizar el tiempo de desarrollo al tiempo que maximice la calidad del software final, siempre y cuando dichas librerías no resuelvan por sí mismas un

porcentaje elevado del proyecto. Usar un sistema de internacionalización para traducción instantánea del software dependiendo del lenguaje instalado en el sistema del usuario, usar un lenguaje robusto que permita la ejecución en distintas plataformas (quizás Java) y que sea lo suficientemente conocido para permitir a cualquier persona colaborar en el proyecto una vez presentado. Además, implementar un sistema que permita la configuración sencilla del simulador para las ocasiones en que el usuario sea novel y permitir la configuración más concreta de todos los parámetros para personas que deseen obtener el máximo partido el simulador. Con respecto a la documentación, toda será distribuida bien mediante HTML si es online o bien en formato PDF (*Portable Document Format*) para asegurar su posible consulta al mayor número de personas.

3.3.2. Estudio de viabilidad

Antes de proceder al desarrollo del simulador, se han realizado investigaciones encaminadas a prever si dicho desarrollo será factible tanto económicamente como operacionalmente y técnicamente.

Viabilidad técnica.

Tras un estudio se ha comprobado que todo lo que se desea implementar tiene solución desde un punto de vista técnico.

Como lenguaje de programación se utilizará Java, resuelve los problemas de internacionalización, robustez y portabilidad. Java está disponible gratuitamente vía Internet; proporciona API's mejoradas para la programación de los aspectos visuales del simulador y es multihilo.

Como librería *opensource* basada en Java para la generación de gráficas estadísticas se utilizará una renombrada y gratuita librería llamada JFreeChart que entre otras cosas permite la exportación de las gráficas a disco en formato de ficheros gráficos y la impresión de las mismas desde la propia aplicación que la use.

El resto de herramientas para creación y generación de documentación, retoque de imágenes, etcétera, están también proporcionados por el mundo del software libre. Por otro lado se dispone de las siguientes plataformas para la prueba de desarrollo:

- Linux sobre Sun UltraSpark (64 bits), que dispone de Java.
- Linux sobre Intel, que dispone de java.
- Windows sobre Intel, que dispone de Java.

Así que aseguramos la prueba en la mayoría de plataformas, como era requerido.

Viabilidad económica.

Está asegurada. Como proyecto final de carrera, el desarrollo será llevado a cabo por un estudiante a tiempo completo sin cobrar nada. Aún así, este proyecto ha conseguido parte de la financiación del Ayuntamiento de Zafra en unión con la Universidad de Extremadura en forma de premio-beca.

Viabilidad operacional.

El simulador, una vez terminado, no necesitará de personal encargado de su mantenimiento nada más que para aquellas modificaciones que en un futuro se quisiesen hacer. Además no necesitará de una gestión y/o configuración continuada puesto que el simulador no incorpora ningún servidor externo que se deba mantener. La viabilidad operacional por tanto también se cumple.

3.3.2.1. Análisis de costes y beneficios

Aunque como se ha comentado en el punto anterior, el alumno encargado del desarrollo del sistema no cuenta con un sueldo, se va a hacer una estimación aproximada de lo que costaría la implementación real de este simulador. En cuanto a los beneficios, no se pueden cuantificar económicamente puesto que el simulador no va dirigido a una organización con ánimo de lucro sino a una entidad educativa; no va a ahorrar costes monetarios.

Los beneficios obtenidos por este proyecto son:

- Capacidad de probar nuevas tecnologías y servir de base a nuevas líneas de investigación.
- Facilitar el desarrollo docente en las clases específicas sobre MPLS.
- Facilitar las investigaciones sobre MPLS en relación a GoS.
- Propiciar la presentación de artículos nacionales e internacionales.
- Propiciar la asistencia a congresos y simposios sobre simulación, redes, informática aplicada, etcétera.
- En caso de implementarse realmente el proyecto (en un conmutador), se obtendrían los beneficios propios de la aportación de GoS al tráfico privilegiado.

Y el análisis de los costes que derivarían del desarrollo del simulador en otro ámbito que no fuese el de un PFC son los que se detallan en los siguientes apartados.

Nota, se supondrá que una sola persona desarrolla el proyecto con un sueldo desarrollando todas las actividades necesarias y con un sueldo medio de 30 € por hora, costes sociales y margen de beneficios incluidos. La jornada laboral será de 35 horas semanales, de lunes a viernes.

3.3.2.1.1. Plan del desarrollo. Estimación de costes

Se establecerá como **fecha inicio del proyecto de desarrollo el 1 de Enero de 2004**. Realmente este proyecto final de carrera ha comenzado el 1 de Julio de 2003, pero se ha de considerar como un proyecto aparte el "proyecto de investigación" que ha propiciado la definición de los métodos para obtener GoS en MPLS y por otro lado, el proyecto puramente software que implica la creación del simulador Open SimMPLS 1.0, el cual ha comenzado en la fecha indicada al comienzo de este párrafo.

El plan de desarrollo (estimación) está expresado en la tabla junto con los datos finales (real) tanto económicos como temporales, que se han obtenido una vez finalizado el PFC.

Fase	**Análisis del problema**
Inicio	1 de Enero de 2004

Fin	31 de Enero de 2004		
	Estimación	**Real**	**Diferencia respecto la estimación**
Duración	22*7 = 154 h.	17*7 = 119 h.	-35 h.
Coste	154*30 = 4.620 €	119*30 = 3.750 €	-870 €

Fase	**Diseño del simulador**		
Inicio	1 de Febrero de 2004		
Fin	14 de Marzo de 2004		
	Estimación	**Real**	**Diferencia respecto la estimación**
Duración	30*7 = 210 h.	26*7 = 182 h.	-28 h.
Coste	210*30 = 6.300 €	182*30 = 5460 €	-840 €

Fase	**Diseño detallado del simulador**		
Inicio	15 de Marzo de 2004		
Fin	30 de Abril de 2004		
	Estimación	**Real**	**Diferencia respecto la estimación**
Duración	35*7 = 245 h.	44*7 = 308 h.	63 h.
Coste	245*30 = 7.350 €	308*30 = 9.240 €	1.890 €

Fase	**Implementación del simulador**		
Inicio	1 de Mayo de 2004		
Fin	30 de Octubre de 2004		
	Estimación	**Real**	**Diferencia respecto la estimación**
Duración	129*7 = 903 h.	129*7 = 903 h.	0 h.
Coste	903*30 = 27.090 €	903*30 = 27.090 €	0 €.

Fase	**Pruebas**		
Inicio	1 de Noviembre de 2004		
Fin	21 de Noviembre		
	Estimación	**Real**	**Diferencia respecto la estimación**
Duración	15*7 = 105 h.	10*70=70 h.	-35 h.
Coste	105*30 = 3.150 €	70*30 = 2.100 €	- 1050 €

Fase	**Documentación**		
Inicio	22 de Noviembre de 2004		
Fin	5 de Diciembre de 2004		
	Estimación	**Real**	**Diferencia respecto la estimación**
Duración	10*7 = 70 h.	20*10 = 200	130
Coste	70*30 = 2.100 €	200*30 = 6.000 €	3.900 €

Así que el coste y duración finales estimados y reales del proyecto, de haberse llevado a cabo en un ámbito empresarial y no universitario, hubiese sido el siguiente:

Inicio	1 de Enero de 2004		
Fin	5 de Diciembre de 2004		
	Estimación	**Real**	**Diferencia respecto la estimación**
Duración total	1.687 h.	1.782 h.	95 h.
Coste total	50.610 h.	53.460 €	2.850 €

Es decir, el proyecto se finaliza unos días más tarde de los estimado y por tanto, a un coste un poco superior. Este coste habrá que descontarlo del margen de beneficio que implícitamente se ha incluido en el salario por hora de trabajo, puesto que el coste del proyecto final para el cliente no varía.

3.3.3. Manual de referencia. Guía del programador.

Este manual constituye una ayuda para el buen entendimiento de cómo funciona el simulador. La referencia completa no ha podido ser incluida en esta documentación por problemas de espacio puesto que tiene una extensión de más de 2.500 páginas. Toda persona interesada en profundizar en la implementación del simulador debería consultar dicha referencia completa (se encuentra en el CD de entrega del PFC).

En la web del proyecto http://patanegra.unex.es/opensimmpls se puede descargar la referencia completa en formato PDF o bien consultarla online puesto que está disponible también en HTML navegable.

En esta guía del programador se va a explicar el funcionamiento interno del simulador; a veces con diagramas y otras veces comentando el funcionamiento de alguna clase o método concreto. Al finalizar este capítulo, el lector estará en condiciones de abordar el manual de referencia completo que se ha comentado, sabiendo en todo momento la organización interna de simulador.

3.3.3.1. Paquetes y clases

Aunque como se ha comentado no se va a copiar aquí el manual de referencia que se encuentra en formato electrónico, se va a hacer un listado de todos los paquetes del simulador y una breve descripción de cada una de las clases de los mismos, para tener unas nociones mínimas que permitan entender todo con claridad.

3.3.3.1.1. simMPLS.electronica.dmgp

En este paquete se encuentran las clases necesarias para implementar a memoria DMGP de los nodos activos, donde se almacenarán los paquetes con GoS para su posible retransmisión.

Las clases de este paquete son las siguientes.

- **TDMGP:** esta clase implementa una memoria DMGP.
- **TEntradaDMGP:** esta clase implementa una de las entradas de la memoria DMGP.
- **TEntradaFlujoDMGP:** esta clase implementa una entrada de identificación para un flujo dentro de una memoria DMGP.
- **TEntradaPeticionesGPSRP:** esta clase implementa una de las entradas de la tabla donde se almacenan las peticiones pendientes de respuesta del protocolo GPSRP.
- **TMatrizPeticionesGPSRP:** esta clase implementa la tabla donde se almacenan las peticiones pendientes de respuesta del protocolo GPSRP.

3.3.3.1.2. simMPLS.electronica.puertos

En este paquete se encuentran las clases necesarias para la implementación de los puertos de los nodos, activos o no, sus búferes, etcétera.

Las clases de este paquete son las siguientes.

- **TEntradaPuertoActivo:** esta clase implementa una entrada de un puerto activo.
- **TPuerto:** esta clase implementa un puerto abstracto.

- **TPuertoActivo:** esta clase implementa un puerto activo, con su algoritmo de gestión y el protocolo EPCD.
- **TPuertoNormal:** esta clase implementa un puerto tradicional, con su algoritmo de gestión.
- **TPuertosNodo:** esta clase implementa un conjunto abstracto de puertos de un nodo.
- **TPuertosNodoActivo:** esta clase implementa un conjunto de puertos activos para un nodo activo. Con su algoritmo de gestión del conjunto y su búfer.
- **TPuertosNodoNormal:** esta clase implementa un conjunto de puertos tradicionales para un nodo no activo. Con su algoritmo de gestión del conjunto y su búfer.

3.3.3.1.3. simMPLS.electronica.recolectorsimulacion

En este paquete se encuentra definida la clase necesaria para recolectar todos los elementos de simulación y mostrarlos en el panel correspondiente del simulador.

Las clases de este paquete son las siguientes.

- **IEventoSimulacionListener:** esta clase define una interfaz que deben implementar todos los objetos que deseen recibir eventos de simulación.
- **TRecolectorSimulacion:** esta clase implementa un recolector de simulación, que será el suscriptor único en Open SimMPLS de eventos de simulación. Todos los eventos de simulación de la aplicación llegan a una instancia de esta clase.

3.3.3.1.4. simMPLS.electronica.reloj

En este paquete se implementan las clases necesarias para la implementación de un reloj del sistema capaz de sincronizar todos los elementos de simulación y para llevar a cabo la temporización de la misma.

Las clases de este paquete son las siguientes.

- **EProgresoUnSoloSuscriptor:** esta clase implementa una excepción que se generará cuando más de un suscriptor se quiera suscribir para recibir eventos de progresión.
- **IEventoProgresionListener:** esta clase implementa una interfaz que deben implementar todos aquellos objetos que deseen recibir eventos de progresión.
- **IEventoRelojListener:** esta clase implementa una interfaz que deben implementar todos aquellos objetos que deseen recibir eventos de reloj.
- **TEventoProgresion:** esta clase define un evento de progresión, que informa sobre el porcentaje de ejecución de la simulación.
- **TEventoReloj:** esta clase define un eventos de reloj, que será enviado a ciertos nodos y que les informa del tic de reloj actual (inicio y fin).
- **TMarcaTiempo:** esta clase define una marca de tiempo, un *timestamp* virtual para la simulación.
- **TReloj:** esta clase define un reloj del sistema. Envía eventos de reloj con cada tic a todos los elementos que lo requieran y posteriormente espera que todos lo hayan consumido antes de proceder a enviar el siguiente.

3.3.3.1.5. simMPLS.electronica.tldp

En este método se encuentran definidas todas las clases que son necesarias para implementar la matriz de conmutación/encaminamiento de un nodo, es decir, ILM, FTN y NHLFE, además de los añadidos propios de la propuesta de este PFC.

Las clases de este paquete son las siguientes.

- **TEntradaMatrizConmutacion:** esta clase implementa una entrada completa en la matriz de conmutación, formado por ILM, FTN y NHLFE además de los campos requeridos por RLPRP.
- **TMatrizConmutacion:** esta clase implementa una tabla de conmutación/encaminamiento completa para MPLS.

3.3.3.1.6. simMPLS.entradasalida.osm

En este paquete se encuentran las clases necesarias para almacenar y cargar de/a disco los escenarios.

Las clases de este paquete son las siguientes.

- **TAlmacenadorOSM:** esta clase implementa un almacenador capaz de almacenar un escenario completo existente en memoria a disco, en un formato de texto legible para el humano.
- **TCargadorOSM:** esta clase implementa un cargador, capaz de cargar desde un fichero de texto del disco, un escenario a memoria.

3.3.3.1.7. simMPLS.entradasalida.red

En este paquete se encuentran las clases necesarias para enviar un correo electrónico desde la propia aplicación.

Las clases de este paquete son las siguientes.

- **TSMTP:** esta clase implementa un enviador de mensajes vía SMTP.

3.3.3.1.8. simMPLS.escenario

En este paquete se encuentran muchas de las clases más importantes. Se encuentran las implementaciones de todos los elementos de la topología, de la topología en si, de escenarios, estadísticas, eventos de simulación, etcétera.

Las clases de este paquete son las siguientes.

- **ESimulacionUnSoloSuscriptor:** esta clase define una excepción que se activará cuando más de un objeto se intenta suscribir para recibir eventos de simulación.
- **TConfigEnlace:** esta clase define un objeto encargado de almacenar la configuración de un nodo.

- **TElementoTopologia:** esta clase define un elemento abstracto de la topología. Sirve para hacer polimorfismo y herencia.
- **TEnlaceExterno:** esta clase implementa un enlace externo al dominio MPLS.
- **TEnlaceInterno:** esta clase implementa un enlace interno al dominio MPLS.
- **TEnlaceTopologia:** esta clase implementa un enlace abstracto de la topología. Se usa para realizar polimorfismo y herencia.
- **TEntradaBufferEnlace:** esta clase implementa una entrada de un búfer que tiene el enlace a efectos de poder simular correctamente el tránsito de los paquetes por él.
- **TEscenario:** esta clase implementa el objeto más grande de todo el simulador. Un escenario completo y funcional en memoria.
- **TESEnlaceCaido:** esta clase implementa un evento que informará a quien lo reciba de que un enlace activo ha fallado.
- **TESEnlaceRecuperado:** esta clase implementa un evento que informará a quien lo reciba de que un enlace caído se ha recuperado.
- **TESEtiquetaAsignada:** esta clase implementa un evento que informará a quien lo reciba de que se ha asignado una etiqueta MPLS.
- **TESEtiquetaDenegada:** esta clase implementa un evento que informará a quien lo reciba de que se ha denegado una etiqueta MPLS.
- **TESEtiquetaEliminada:** esta clase implementa un evento que informará a quien lo reciba de que se ha eliminado una etiqueta MPLS.
- **TESEtiquetaRecibida:** esta clase implementa un evento que informará a quien lo reciba de que se ha recibido una etiqueta MPLS.
- **TESEtiquetaSolicitada:** esta clase implementa un evento que informará a quien lo reciba de que se ha solicitado una etiqueta MPLS.
- **TESLSPEliminado:** esta clase implementa un evento que informará a quien lo reciba de que se ha eliminado un LSP.
- **TESLSPEstablecido:** esta clase implementa un evento que informará a quien lo reciba de que se ha establecido un LSP.
- **TESNodoCongestionado:** esta clase implementa un evento que informará a quien lo reciba del porcentaje de congestión de un nodo.
- **TESPaqueteConmutado:** esta clase implementa un evento que informará a quien lo reciba de que se ha conmutado un paquete.
- **TESPaqueteDescartado:** esta clase implementa un evento que informará a quien lo reciba de que se ha descartado un paquete.

- **TESPaqueteEncaminado:** esta clase implementa un evento que informará a quien lo reciba de que se ha encaminado un paquete.
- **TESPaqueteEnTransito:** esta clase implementa un evento que informará a quien lo reciba de que hay un paquete en tránsito por un enlace.
- **TESPaqueteEnviado:** esta clase implementa un evento que informará a quien lo reciba de que se ha enviado un paquete.
- **TESPaqueteGenerado:** esta clase implementa un evento que informará a quien lo reciba de que se ha generado un paquete.
- **TESPaqueteRecibido:** esta clase implementa un evento que informará a quien lo reciba de que se ha recibido un paquete.
- **TEstadisticas:** esta clase implementa unas estadísticas abstractas para un nodo de la topología. Se usa para permitir herencia y polimorfismo.
- **TEstadisticasEmisor:** esta clase implementa un sistema de recolección de datos estadísticos de un nodo emisor.
- **TEstadisticasLER:** esta clase implementa un sistema de recolección de datos estadísticos de un nodo LER.
- **TEstadisticasLERA:** esta clase implementa un sistema de recolección de datos estadísticos de un nodo LERA.
- **TEstadisticasLSR:** esta clase implementa un sistema de recolección de datos estadísticos de un nodo LSR.
- **TEstadisticasLSRA:** esta clase implementa un sistema de recolección de datos estadísticos de un nodo LSRA.
- **TEstadisticasReceptor:** esta clase implementa un sistema de recolección de datos estadísticos de un nodo receptor.
- **TEventoSimulacion:** esta clase implementa un evento de simulación abstracto. Se usa para permitir herencia y polimorfismo.
- **TNodoEmisor:** esta clase implementa un nodo emisor, con todos sus componentes, protocolos, técnicas y modos de operación.
- **TNodoLER:** esta clase implementa un nodo LER, con todos sus componentes, protocolos, técnicas y modos de operación.
- **TNodoLERA:** esta clase implementa un nodo LERA, con todos sus componentes, protocolos, técnicas y modos de operación.

- **TNodoLSR:** esta clase implementa un nodo LSR, con todos sus componentes, protocolos, técnicas y modos de operación.
- **TNodoLSRA:** esta clase implementa un nodo LSRA, con todos sus componentes, protocolos, técnicas y modos de operación.
- **TNodoReceptor:** esta clase implementa un nodo receptor, con todos sus componentes, protocolos, técnicas y modos de operación.
- **TNodoTopologia:** esta clase implementa un nodo abstracto, que será usado para permitir herencia y polimorfismo.
- **TSimulacion:** esta clase implementa un objeto que contendrá todos los datos de configuración de la simulación.
- **TTopologia:** esta clase implementa una topología completa, en forma de grafo y permite realizar operaciones sobre dicha topología, como por ejemplo recorridos Floyd o recorridos RABAN, que están implementados aquí.

3.3.3.1.9. simMPLS.interfaz.dialogos

En esta clase se encuentran las clases de todos los cuadros de diálogo de la aplicación; ventanas de confirmación, de error, de advertencia, etcétera. Todo cuadro de diálogo se encuentra aquí.

Las clases de este paquete son las siguientes.

- **JLicencia:** esta clase implementa una ventana que muestra la licencia GPL bajo la cual está liberado el simulador.
- **JPanelCoordenadas:** esta clase implementa un panel que es capaz de crear una miniatura del escenario en curso. Se usa para seleccionar en él la posición donde colocar un nuevo elemento.
- **JSobre:** esta clase implementa una ventana que muestra los créditos del simulador.
- **JVentanaAdvertencia:** esta clase implementa una ventana de uso general para mostrar advertencias.
- **JVentanaAyuda:** esta clase implementa una ventana que muestra la ayuda del simulador.

- **JVentanaBooleana:** esta clase implementa una ventana de uso genérico que permite obtener respuestas a preguntas SI/NO. Muy útil en multitud de ocasiones.
- **JVentanaComentario:** esta clase implementa la ventana desde la cual se puede enviar un correo electrónico a los desarrolladores de la aplicación.
- **JVentanaEmisor:** esta clase implementa una ventana que permite configurar un nodo emisor.
- **JVentanaEnlace:** esta clase implementa una ventana que permite configurar un enlace interno o externo.
- **JVentanaError:** esta clase implementa una ventana genérica que se usa para mostrar al usuario ventanas de error.
- **JVentanaLER:** esta clase implementa una ventana que permite configurar un nodo LER.
- **JVentanaLERA:** esta clase implementa una ventana que permite configurar un nodo LERA.
- **JVentanaLSR:** esta clase implementa una ventana que permite configurar un nodo LSR.
- **JVentanaLSRA:** esta clase implementa una ventana que permite configurar un nodo LSRA.
- **JVentanaReceptor:** esta clase implementa una ventana que permite configurar un nodo receptor.

3.3.3.1.10. simMPLS.interfaz.simulador

Este paquete contiene las clases relacionadas con la interfaz de usuario necesarias para el simulador: ventanas de escenarios, la interfaz de la propia aplicación, etcétera.

Las clases de este paquete son las siguientes.

- **JPanelDisenio:** esta clase implementa el panel sobre el cual se pueden crear y editar un escenario.
- **JPanelSimulacion:** esta clase implementa un panel sobre el cual se observa la simulación en curso, los paquetes, la congestión, la caída de enlace, etcétera.

- **JSimulador:** en esta clase se implementa la interfaz principal del simulador. La ventana donde se pueden abrir escenarios existentes o crear escenarios nuevos.
- **JVentanaHija:** esta clase implementa una ventana donde se mostrará todo lo referente a un escenario completo. Se permite la simulación, la edición y la configuración de los elementos de dicho escenario.

3.3.3.1.11. simMPLS.interfaz.splash

Este paquete almacena la clase necesaria para mostrar la imagen *splash* de la aplicación. Como no es un cuadro de diálogo en sí, no se encuentra en el paquete descrito un poco más arriba.

Las clases de este paquete son las siguientes.

- **JSplash:** esta clase implementa una pantalla/imagen que aparece al comienzo de la ejecución del programa, mostrando los créditos y mensajes sobre las tareas previas al arranque del simulador.

3.3.3.1.12. simMPLS.interfaz.utiles

En este paquete se encuentra una clase que permite la carga anticipada y la distribución de las imágenes en toda la aplicación, ahorrando memoria y tiempo de proceso.

Las clases de este paquete son las siguientes.

- **TDispensadorDeImagenes:** esta clase implementa un objeto que carga todas las imágenes utilizadas en el simulador en memoria. Así se ahorra mucho tiempo en carga de imágenes durante la simulación, se ahorra memoria porque no se cargan dos instancias distintas para una misma imagen y se permite la fácil modificación de una imagen en el conjunto del simulador.

3.3.3.1.13. simMPLS.principal

En este paquete se encuentra la clase principal del sistema. Aquí se encuentra el método main(…) y es esta clase la que se debe invocar para ejecutar el simulador.

Las clases de este paquete son las siguientes.

- **openSimMPLS:** esta clase implementa el método main(…) que se debe invocar para que comience la ejecución del simulador.

3.3.3.1.14. simMPLS.protocolo

En este paquete se implementan una gran cantidad de clases necesarias para la constitución de PDU's de los diferentes protocolos soportados por el simulador.

Las clases de este paquete son las siguientes.

- **TCabeceraIPv4:** esta clase implementa la cabecera de un paquete IPv4, con todos sus campos.
- **TCampoOpcionesIPv4:** esta clase implementa el campo opciones de un paquete IPv4 en la forma que se ha definido para el soporte de GoS. De esta forma el acceso a estos datos e mucho más cómodo.
- **TDatosGPSRP:** esta clase implementa la cabecera de un paquete GPSRP.
- **TDatosTCP:** esta clase implementa un objeto que simula ser la carga útil de un paquete TCP. Útil para simular paquetes de distinto tamaño.
- **TDatosTLDP:** esta clase implementa la cabecera de un paquete TLDP.
- **TEtiquetaMPLS:** esta clase implementa una etiqueta MPS.
- **TPDU:** esta clase implementa un paquete genérico. Se usa para permitir la herencia y el polimorfismo.
- **TPDUGPSRP:** esta clase implementa un paquete GPSRP.
- **TPDUIPv4:** esta clase implementa un paquete IPv4.
- **TPDUMPLS:** esta clase implementa un paquete MPLS.
- **TPDUTLDP:** esta clase implementa un paquete TLDP.

- **TPilaEtiquetasMPLS:** esta clase implementa una pila de etiquetas MPLS con todas las operaciones necesarias para trabajar fácilmente con ella.

3.3.3.1.15. simMPLS.utiles

En este paquete se encuentran un conjunto de clases de uso general que podrán ser reutilizadas así tal cual en cualquier otro proyecto.

Las clases de este paquete son las siguientes.

- **EDesbordeDeIP:** esta clase implementa una excepción que se disparará cuando el generador de direcciones IP únicas de un escenario se quede sin direcciones.
- **EDesbordeDelIdentificador:** esta clase implementa una excepción que se disparará cuando el generador de identificadores únicos de un escenario se quede sin direcciones.
- **JFiltroOSM:** esta clase implementa un filtro que permite a los cuadros de diálogo mostrar exclusivamente los fichero *.osm de escenario.
- **TActualizadorDeProgreso:** esta clase implementa un objeto que se encarga de actualizar la barra de progreso de simulación cuando es necesario hacerlo.
- **TEventoSimMPLS:** esta clase implementa un evento abstracto, es el superevento de este simulador. Se usa para proporcionar herencia y polimorfismo.
- **TGeneradorDeIP:** esta clase implementa un generador de direcciones IP únicas para asignar, a modo de DHCP, cuando se requiere para un nodo nuevo del escenario.
- **TIdentificador:** esta clase implementa un generador de identificadores únicos que se utiliza en todos aquellos lugares donde es necesario el uso de una clave primaria.
- **TIdentificadorLargo:** esta clase implementa un generador de identificadores únicos largos que se utiliza en todos aquellos lugares donde es necesario el uso de una clave primaria y el volumen de objetos a identificar es muy elevado.
- **TIdentificadorRotativo:** esta clase implementa un generador de identificadores únicos que se utiliza en todos aquellos lugares donde es necesario el uso de una clave primaria. Este generador se reinicia al valor inicial automáticamente cuando llega a su límite superior.

- **TMonitor:** esta clase implementa un monitor genérico, usado para sincronizar hilos y para la creación de secciones críticas en los lugares donde no debe estar permitida la concurrencia.

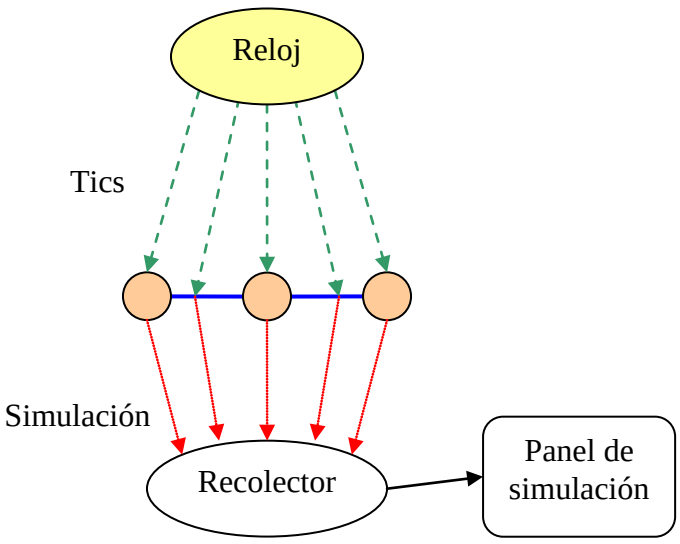
3.3.3.2. La clase principal

La clase principal del sistema, llamada **openSimMPLS**, inicia la ejecución del simulador.

El método `main(…),` que se encuentra en esta clase, Crea un objeto de tipo **TDispensadorDeImagenes** que cargará todas las imágenes necesarias en la aplicación y que posteriormente será pasado como parámetro en el constructor de cualquier elemento referente a la interfaz. Posteriormente se crea un objeto de tipo **JSimulador** que es la interfaz principal de Open SimMPLS 1.0; A partir de este momento la ejecución del simulador dejará de ser secuencial y en su lugar atenderá a los eventos generados por el usuario en la interfaz: clic en los botones, selección de opciones de menú, etcétera.

3.3.3.3. Visión global del escenario

La siguiente imagen es una abstracción de alto nivel del funcionamiento general de un escenario en plena simulación.



En la figura anterior se ve cómo un elemento principal, un reloj, envía a los elementos de la topología (enlaces y nodos, líneas azules y círculos naranjas) unos eventos de temporización. El reloj es un elemento que se configura con dos valores: por un lado, la duración de la simulación; por otro lado, el tic de reloj. El reloj, que se ejecuta en un hilo propio, avanzará desde cero hasta la máxima duración definida para la simulación en tics de reloj de una determinada duración y cada tic será mandado a todos los elementos de la topología; cuando llegue al máximo de duración de la simulación, su hilo se detendrá y se finalizará la simulación.

Cuando un elemento de la topología recibe un tic, implícitamente está recibiendo la duración del mismo, o sea, unos nanosegundos. Automáticamente cada elemento activa un hilo de ejecución distinto, existiendo por tanto concurrencia, y comienza a hacer su operación predeterminada, por ejemplo conmutar, transportar paquetes, recibir tráfico… El hilo de ejecución se detendrá cuando el nodo no pueda hacer más operaciones porque el tic se le ha agotado. Cuando todos los elementos han agotado su tic, el reloj lo detecta y genera el siguiente tic, con lo que la operación se vuelve a repetir.

Durante el tiempo que el hilo propio de cada elemento está en funcionamiento ocurren multitud de cosas que deben ser recogidas y que una vez representadas serán la simulación visual de la misma; este proceso de recolección lo realiza un recolector global al escenario, como el reloj, al que todos los elementos notifican qué están haciendo durante el tiempo que están en funcionamiento.

Repitiendo este proceso reiteradas veces se consigue una simulación fluida que parece continua aunque en realidad es discreta.

La clase que implementa el reloj del sistema es **TReloj**. Incorpora una lista interna de todos los elementos a los que debe enviar eventos de temporización.

Todos los elementos de la topología, que son **TNodoEmisor**, **TNodoReceptor**, **TNodoLER**, **TNodoLSR**, **TNodoLERA**, **TNodoLSRA**, **TEnlaceInterno** y **TEnlaceExterno** implementan obligatoriamente un par de métodos que permitan realizar lo que se ha comentado:

- **capturarEventoReloj()**, que permite al reloj enviar tics de temporización al nodo, lo cual a su vez provoca en el nodo que su hilo de ejecución se active.
- **esperarFinalizacion()**, que permite saber al reloj cuándo un nodo ha consumido su tic; de esta forma cuando todos los elementos han consumido su tic, el reloj procede a generar uno nuevo.

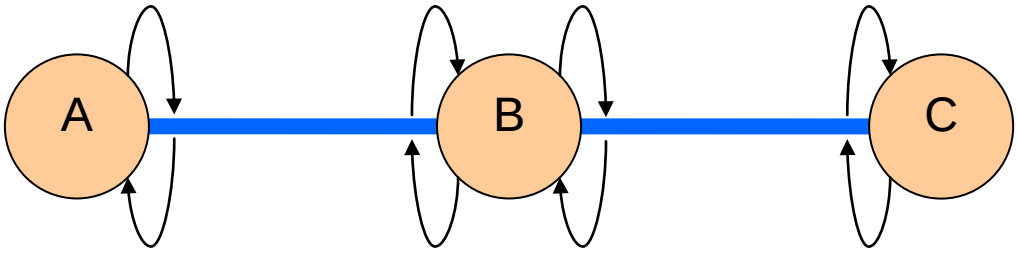
Todos los elementos de la topología deben incorporar un atributo interno que almacene una referencia al recolector al que deben enviar los eventos.

La clase que implementa el recolector de eventos de simulación se llama **TRecolectorSimulacion**. Implementa un importante método:

- **capturarEventoSimulacion()**, que permite que los elementos de la topología puedan enviarle los eventos de simulación que van generando durante los tiempos en que sus hilos están en ejecución.

3.3.3.4. Comunicación entre elementos

El proceso no es tan sencillo como se ha explicado en el punto anterior. En realidad cada elemento además de las operaciones que realiza mientras dispone de tiempo asignado, también mantiene comunicación con algunos elementos de la red. Veámoslo más de cerca:



Se observa que realmente en cada tic de reloj, independientemente de lo que cada elemento realice, existe una comunicación entre elementos. Esta comunicación entre elementos se realiza mediante los puertos de conexión de los nodos. Los nodos tienen unos puertos de

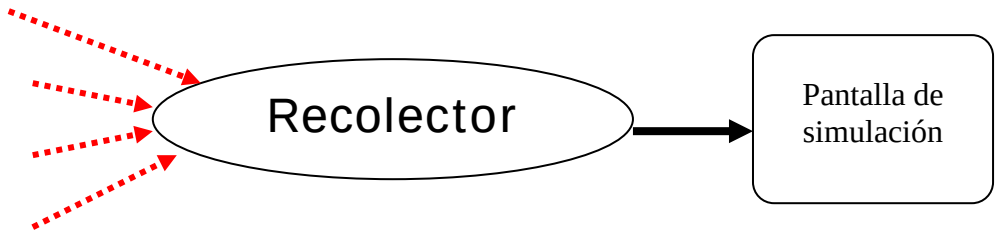
conexión. En principio de tipo **TPuertosNodo**. Cada objeto de este tipo tiene un número determinado y fijo de puertos individuales de tipo **TPuerto.** Como un puerto conecta tanto a un enlace como a un nodo, debería permitir al enlace acceder al nodo y al nodo acceder al enlace. Así cuando un nodo debe enviar tráfico a otro nodo, puede obtener un puerto, acceder al enlace que lo conecta con otro nodo y depositar ahí el paquete correspondiente; ya llegará al destino. Asimismo, cuando un paquete que viaja por un enlace llegue al destino, el enlace podrá solicitar al puerto que le de acceso al nodo y depositará en el nodo el paquete.

Esto es así. El conjunto de puertos de un nodo implementa obligatoriamente un par de métodos:

- **obtenerEnlaceDePuerto(),** que permite al nodo acceder al enlace al que está unido.
- **obtenerNodo()**, que permite al enlace acceder al nodo al que está unido.

3.3.3.5. Visualización de la simulación

En la visión global, aparece cómo el recolector de eventos de simulación recoge los eventos; incluso se explica el método que utiliza. Pero el recolector sólo recoge, no muestra los eventos. Para ello ha de utilizar los servicios de un componente gráfico que permita la visualización. La figura siguiente lo muestra más cerca:



El concepto es que el recolector se dedique a recoger eventos y la pantalla de simulación se dedique a simularlos, ha realizar la parte visual. Por tanto parece lógico que el recolector deba tener un atributo interno que almacene una referencia a la pantalla de simulación. En efecto, es así; y es la primera unión con la interfaz de usuario.

Por otro lado, la pantalla de simulación está implementada en la clase **JPanelSimulación**, que realiza todas las operaciones de refresco, simulación visual… es decir, interpreta todos los eventos que le llegan al recolector y los muestra de una forma amigable para el usuario.

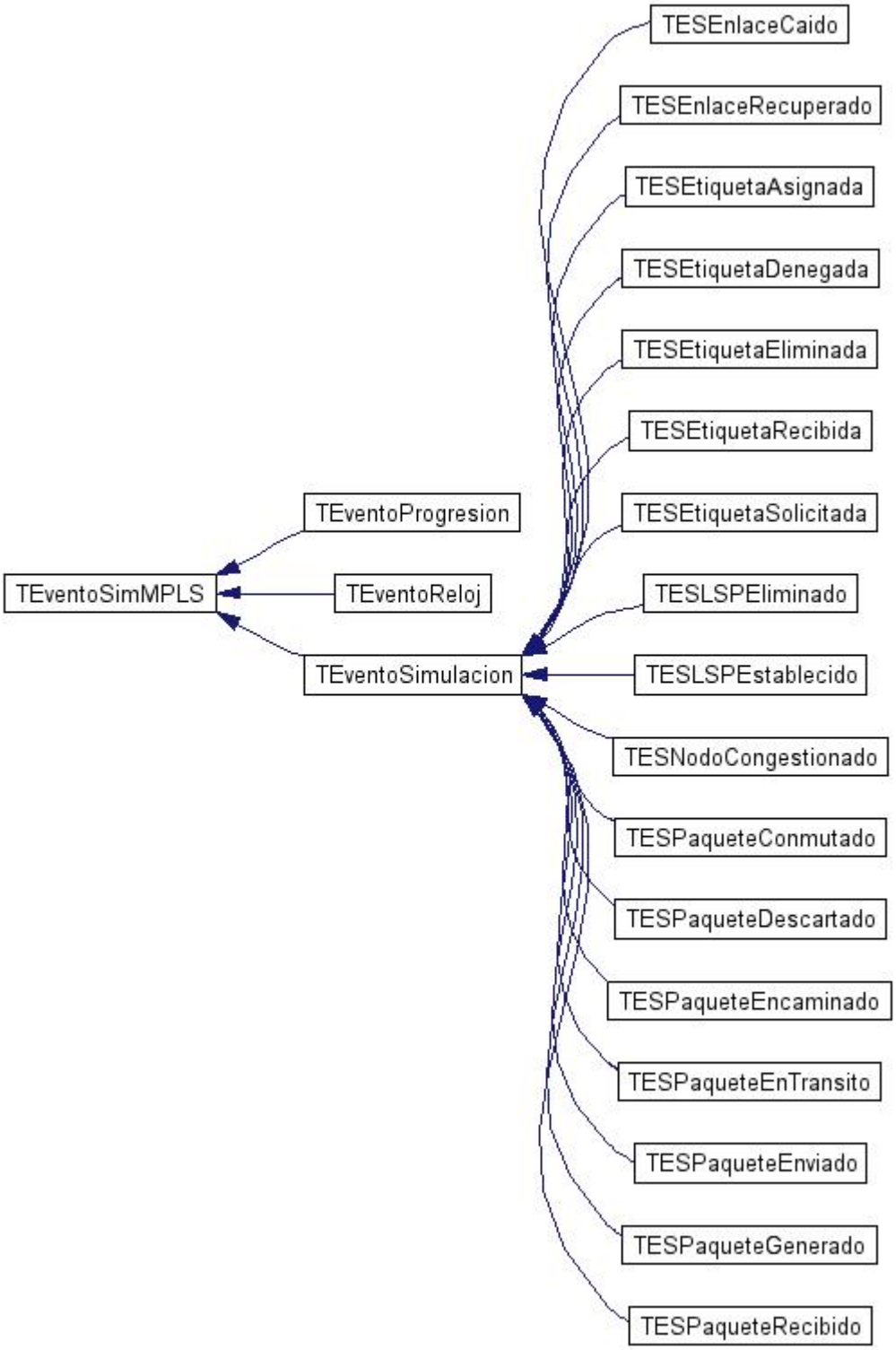
El panel de simulación implementa el siguiente método:

- **ponerEvento()**, que es el método usado por el recolector para enviarle los eventos al panel de simulación.

Tanto el recolector de simulación como el panel de simulación funcionan sin hilos propios. Se activa cuando le llega un evento que procesar, lo procesa y lo envía al panel de simulación que lo representará.

3.3.3.6. Jerarquía de eventos

Existen muchos eventos distintos de simulación; en realidad todos los eventos del simulador forman una jerarquía y por comodidad, las interfaces de los métodos que usan eventos se han definido para aceptar eventos de la superclase más alta o de un nivel más bajo, pero no se han creado métodos que trabajen directamente con los eventos de las clases más específicas. La jerarquía de eventos es la siguiente:



Todos los eventos incorporan varios atributos imprescindibles:

- Una **referencia al elemento que los generó**; así el objeto que recibe los eventos puede acceder también a al objeto que los generó y a sus métodos.
- Un **identificador único** y global para todo el escenario.

- Un atributo que define de qué **tipo de evento** se trata.

El hecho de que cada evento lleve un identificador único significa que debe haber un objeto en memoria que genere dichos identificadores y que además sea el mismo para todos los elementos. Así se asegura que no hay eventos confundibles.

Dicho objeto se encuentra definido en la clase **TIdentificadorLargo** y su método más interesante es **obtenerNuevo()**, que devuelve un identificador único y actualiza los valores internos para que la siguiente petición genere también un identificador único.

Este generador de identificadores pertenece al escenario y se pasa por parámetros a todos los elementos; así que además de todo lo comentado anteriormente, un elemento cualquiera contiene un atributo que es una referencia a este generador de identificadores.

3.3.3.7. La topología

La topología es un objeto que almacena todos los elementos del escenario y que se encarga de conectar o interconectar enlaces con nodos y de establecer las asociaciones entre los elementos y el reloj o los elementos y el recolector de eventos de simulación. Se encuentra implementada en la clase **TTopologia**.

Para realizar estas operaciones debe tener bien identificados todos los elementos por lo cual cada elemento debe llevar un identificador, como ocurría con los eventos; en este caso, la topología es la que posee un generador de identificadores que en este caso no permite obtener identificadores largos puesto que no es necesario debido al bajo número de elementos que contendrá la topología. Concretamente la topología tiene tres atributos muy importantes:

- **Generador de identificadores**, para poder asignar un identificador distinto a cada elemento que se inserte en la topología.
- **Generador de direcciones IP**, para asignar una IP distinta a cada nodo que se inserte en la topología. Puesto que este simulador trabajará con IP, cada nodo debe

tener una y este generador de IP ofrece una distinta para cada nodo, actuando como si se tratase de un servidor DHCP.

- **Reloj del sistema,** que será el encargado de hacer comenzar la simulación enviando tics de reloj a los elementos.

- El generador de identificadores es de tipo **TIdentificador** y su método más usado es **obtenerNuevo()** que devuelve un identificador distinto cada vez que se invoca y nunca repetidos.

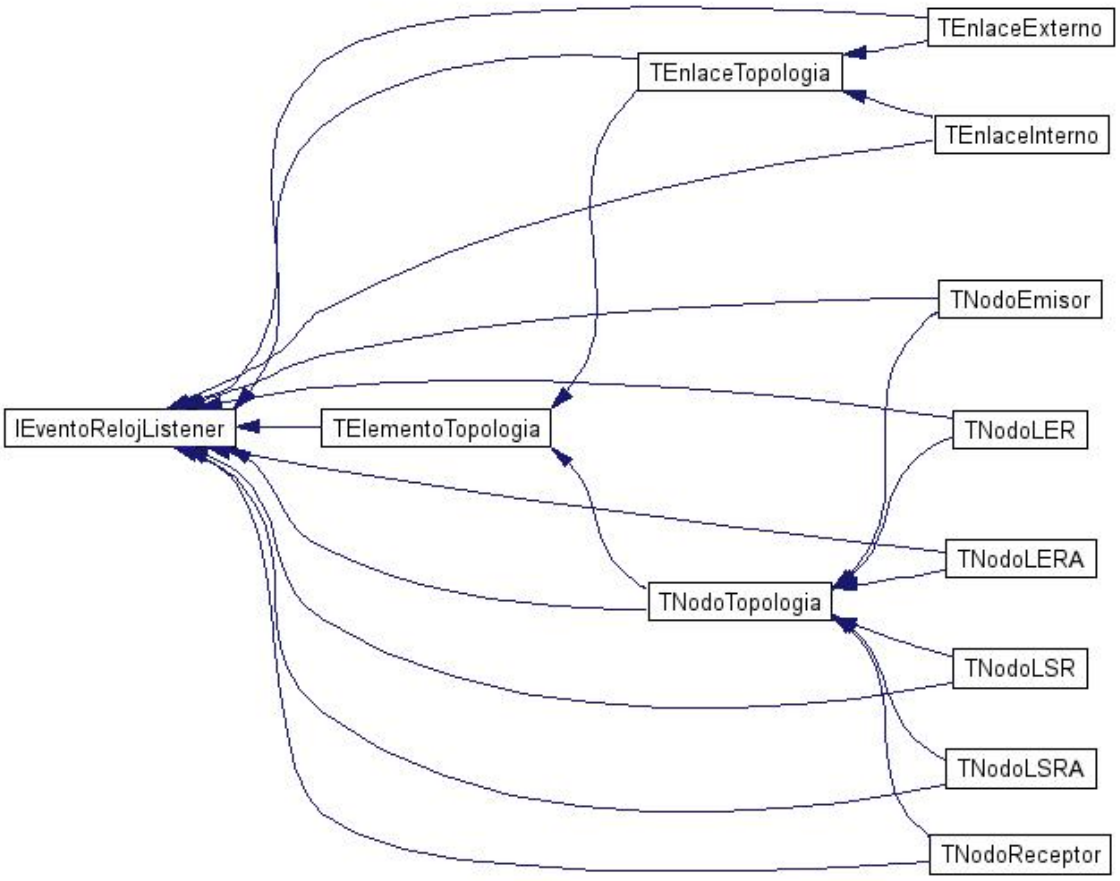
- El generador de direcciones IP es de tipo **TGeneradorDeIP** y su método más usado es **obtenerIP()** que devuelve una IP distinta cada vez que se invoca, dentro del rango 10.0.0.1 – 10.255.255.254.

Pero probablemente lo más importante de la topología sea los métodos que implementa; son muchos y algunos muy importantes, pero cabe destacar tres que es donde se implementan los algoritmos de encaminamiento:

- **obtenerIPSalto(),** este método recibe como parámetros la IP origen de un paquete y la IP destino a donde se desea llegar y devuelve la IP del siguiente nodo, partiendo del origen, por el que hay que seguir. El cálculo se realiza mediante un algoritmo de Floyd tradicional.
- **obtenerIPSaltoRABAN(),** este método recibe como parámetros la IP origen de un paquete y la IP destino a donde se desea llegar y devuelve la IP del siguiente nodo, partiendo del origen, por el que hay que seguir. El cálculo se realiza mediante un algoritmo de Floyd pero con los pesos especificados por RABAN, que se explicaron en la propuesta de este PFC.
- **obtenerIPSaltoRABAN(),** realiza la misma labor que el anterior, pero además admite otro parámetro que es la IP de un nodo adyacente al origen por le cual no se desea pasar. Así que a efectos prácticos este método devuelve el segundo mejor salto posible para llegar del origen al destino.

3.3.3.8. Jerarquía de elementos

Como en el caso de los eventos, se ha creado una jerarquía de elementos para que los métodos que tienen como parámetro o bien devuelve un elemento, puedan hacer uso del polimorfismo y aceptar o devolver elementos de la superclase. Así, la jerarquía de elementos sería:



Donde se ve claro que además de ser una jerarquía que hereda de **TElementoTopologia**, todos los elementos implementan la interfaz **IEventoRelojListener**, que es la que les permitirá que puedan recibir eventos de temporización por parte del reloj.

3.3.3.9. El escenario

El escenario es un objeto que contiene todo lo referente a un escenario completo de simulación. Todos los elementos que se pueden comentar: reloj, recolector, topología, generadores de identificadores y de IP, elementos… se encuentra en un objeto de este tipo

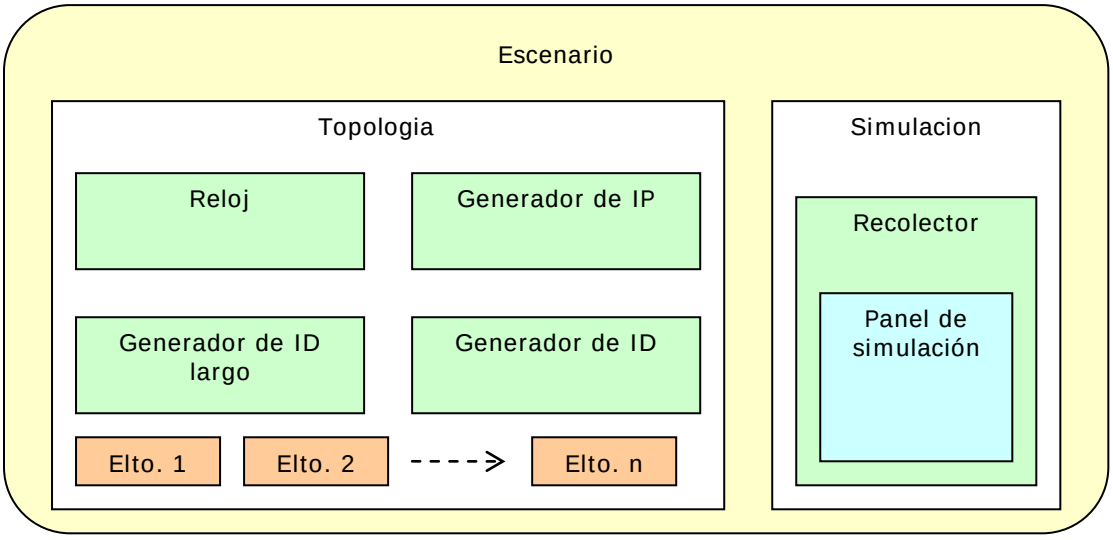
de tal forma que es un solo objeto el que contiene en la memoria todos los elementos de cada escenario; y además es autónomo; puede realizarse una simulación sin conexión con una interfaz de usuario.

Un objeto de escenario es de tipo **TEscenario** y entre sus atributos más importantes podemos encontrar:

- **Una topología**, de tipo **TTopologia**, donde se almacenan todos los elementos y donde se realizan todas las conexiones entre ellos, como se ha visto antes.
- **Un objeto de simulación**, de tipo **TSimulacion**, que es donde se encuentra el recolector de eventos de simulación.

El método más importante del escenario es **generarSimulacion()**, que ponen en funcionamiento el reloj de la topología y de este modo la simulación comienza a funcionar. Realmente el tipo **TEscenario** contiene todo lo necesario para funcionar aunque no haya una interfaz de usuario conectada directamente.

Para ir centrando un poco todos los conceptos, veamos una gráfica de cómo está organizado por dentro, con gran nivel de abstracción, el objeto de tipo **TEscenario**.

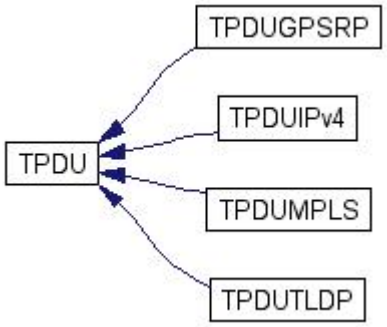


Además, para facilitar muchas operaciones, todos los elementos del escenario tienen acceso mediante una referencia, al objeto que los engloba. Así los elementos tiene una referencia a la topología, lo que significa que “la topología está embebida” dentro de cada

elemento. La topología tiene una referencia al escenario; igual ocurre con la simulación. De este modo, todos los objetos importantes pueden acceder al resto de objetos importantes por medio de referencias.

3.3.3.10. Jerarquía de paquetes

Antes de comenzar a detallar el funcionamiento de los nodos, es necesario hacer un inciso en cómo está construida la jerarquía de paquetes. De nuevo, se ha realizado una jerarquía de objetos para poder aprovechar la herencia y el polimorfismo que se consiguen. La jerarquía de paquetes es la siguiente.



Todos los paquetes, heredan de TPDU que es una superclase abstracta que debe ser redefinida por las clases que hereden de ella. Hay cinco atributos que tiene todo paquete independientemente del tipo que sea:

- **La IP de origen**, que es un **String** que expresa la dirección del nodo que originó el paquete.
- **La IP de destino**, que es un **String** que expresa la dirección del nodo al que va dirigido el paquete.
- Un **identificador único** para que cuando los paquetes son almacenados en los búferes, puedan ser identificadores inequívocamente.
- **Tipo** de paquete que se trata: IP, MPLS… A este atributo se le asignará alguna de las constantes que están definidas en esta clase para ese propósito.

- **Cabecera IPv4,** de tipo **TCabeceraIPv4.** En Open SimMPLS 1.0 todos los paquetes o viajan sobre IP o bien contienen transportan un paquete IP por lo que este dato es necesario en todos los casos.

TPDUIPv4 implementa un paquete del protocolo IPv4; **TPDUGPSRP** implementa un paquete del protocolo GPSRP, **TPDUMPLS** implementa un paquete del protocolo MPLS y **TPDUTLDP** implementa un paquete del protocolo de señalización TDLP. Cada uno de estas clases tiene métodos propios que básicamente permiten acceder a los campos propios de ese tipo de tráfico.

Sin embargo **TPDUMPLS** tiene un método que es necesario comentar con más detalle:

- **obtenerCopia()**, que crea un paquete nuevo e independiente pero idéntico al actual. Es una clonación. Este método sólo lo tiene esta clase.

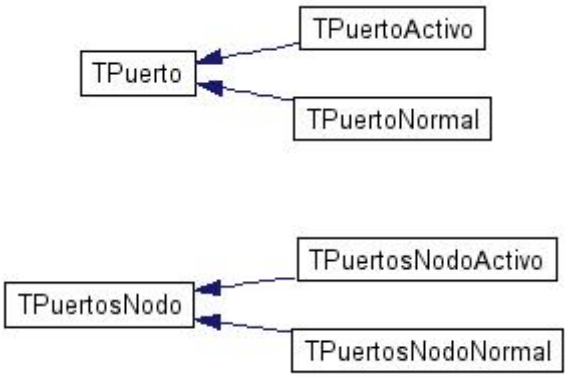
En todo momento en el simulador un paquete se crea una vez y en todo el transcurso de la simulación lo que se utiliza es una referencia a ese paquete. Así se ahorra en memoria y tiempo de ejecución y hay menos fallos; no hay problemas porque el paquete cuando entra en un elemento deja de estar en el anterior y cuando llega al destino, se destruye.

Esto funciona bien en todas las ocasiones excepto cuando un paquete MPLS tiene que ser almacenado por un tiempo indefinido en las memorias DMGP para su posible retransmisión. Al estar simulándose todo con una única referencia al paquete, si un paquete es almacenado en la DMGP por un lado y por otra sigue circulando y cambiando en la topología, se estaría cambiando la misma instancia. Así que para almacenar paquetes en la DMGP y recuperarlos de ella se obtiene antes un clon del paquete; a todos los efectos de simulación y funcionamiento exactamente igual, pero sin ese problema.

Es el único objeto que requiere clonación en todo el simulador; Esto es así por decisión expresa puesto que la clonación de objetos hizo caer el rendimiento del simulador en las pruebas que se hicieron y por tanto se ha organizado todo para trabajar con referencias a un mismo paquete.

3.3.3.11. Jerarquía de puertos y conjuntos de puertos

Se comentó anteriormente que los puertos eran de tipo **TPuerto** y los conjuntos de puertos eran de tipo **TPuertosNodo**; en realidad es cierto y no lo es. Estas dos clases son superclases abstractas de sendas jerarquías de objetos que de nuevo se ha establecido para aprovechar herencia y polimorfismo. Las jerarquías, que son pequeñas, son las siguientes:



Donde se ve que hay puertos activo (**TPuertoActivo**) y normales (**TPuertoNormal**), y conjuntos de puertos activos (**TPuertosNodoActivo**) y conjuntos de puertos normales (**TPuertosNodoNormal**)

Es necesaria la separación porque los puertos y los conjuntos de puertos de los nodos activos y los no activos son diferentes. Mientras en los no activos se implementan puertos con búferes FIFO y extracción de paquetes por Round Robin y los activos tienen una estructura más compleja con colas de prioridad y Round Robin Priorizado para la extracción de paquetes de dichas colas.

En cualquier caso, todo puerto heredado de **TPuerto** implementa tres métodos importantes para la simulación.

- **obtenerPaquete()**, que extrae del búfer del puerto un paquete, acorde a la política que se haya implementado (distinta en puertos activos y no activos, como se ha comentado).

- **ponerPaqueteEnEnlace(…)**, que permite a un nodo poner un paquete en el enlace al que está unido por el puerto, directamente.
- **puedoConmutarPaquete(…)**, que permite averiguar si con el número de nanosegundos de los que dispone el nodo el nodo (obtenidos por los tics enviados por el reloj) se puede conmutar el siguiente paquete que se debe leer del puerto; por ejemplo, si se dispone de nanosegundos para conmutar 100 octetos pero el paquete que debe ser conmutado ocupa 200, este método devolverá false.

Generalmente se usa el método **ponerPaqueteEnEnlace()** directamente del puerto y no del conjunto de puertos, porque es algo individual que se sabe con certeza que sólo se puede hacer a través de un puerto concreto. **ObtenerPaquete()** y **puedoConmutarPaquete()** no suelen usarse directamente por el usuario/nodo porque la elección del paquete es algo más global que concierne a todo el conjunto de puertos. Estos métodos suelen ser usados exclusivamente por objetos de tipo **TPuertosNodoNormal** y **TPuertosNodoActivo**, para su planificación global.

En su lugar, todo conjunto de puertos implementa dos métodos que si serán utilizados directamente por el nodo:

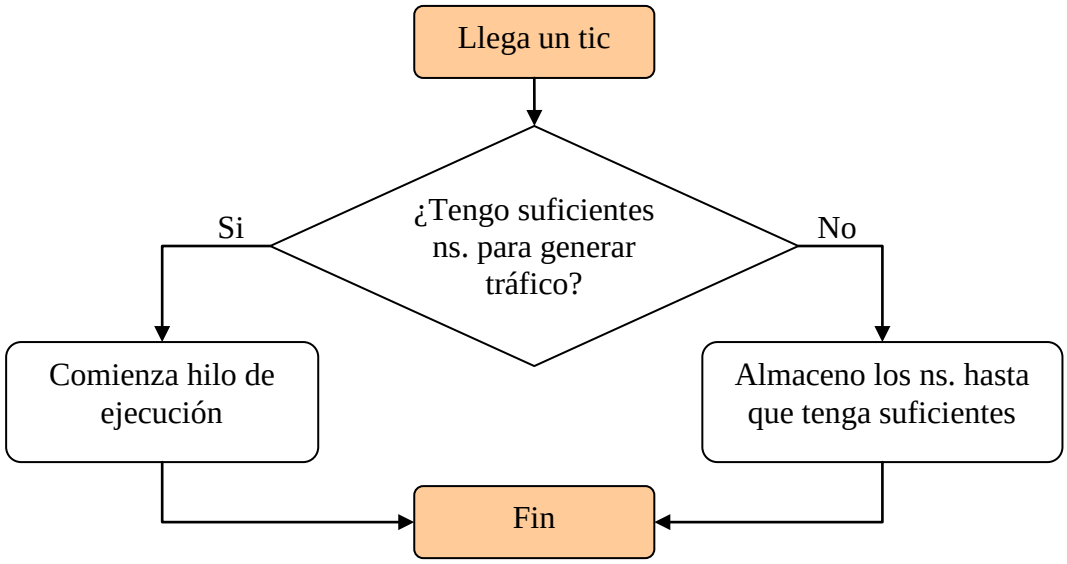
- **leerPaquete()**, que lee un paquete del puerto que deba leer en cada momento y acorde a la política de gestión del puerto y de búfer que se haya implementado (distinta en nodos activo que normales).
- **puedoConmutarPaquete()**, comprueba si con los nanosegundos de que dispone el nodo “padre”, se puede conmutar el paquete devuelto por el método **leerPaquete()**.

Es muy importante saber que en la clase **TPuertoActivo** y sólo en esa clase, se implementa un método llamada **protocoloEPCD()** cuya utilidad es mantener un umbral de 100 octetos libres siempre en el búfer y además que cuando un paquete va a ser descartado, si está marcado con GoS, se inicie un proceso de recuperación, que llevará a cabo el nodo pero que iniciará este método.

3.3.3.12. Funcionamiento del nodo emisor

Ya hemos visto los componentes más importantes del escenario; ahora toca comenzar a ver los elementos importantes a nivel de nodo; en este caso, de un nodo emisor.

Cuando el reloj del simulador se ponen en marcha y llegan tics de reloj al nodo emisor, se pone en funcionamiento un proceso complejo por su densidad, pero sencillo de entender; el siguiente diagrama lo explica bien.



El proceso de llegar un tic ocurre cuando el reloj del simulador invoca al método **capturarEventoReloj()** del nodo, como ya se explicó. Este método actualiza un atributo interno que almacena el montante de nanosegundos de los que dispone el nodo y llama al método **iniciar()** que ponen en funcionamiento el hilo del nodo. Esto hace que se ejecute automáticamente el método **run()** que todo elemento tiene y es por fin en este método donde se codifica toda la funcionalidad del emisor.

El método **run()** realiza las siguientes operaciones en un bucle que finaliza cuando no quedan nanosegundos suficientes para operar:

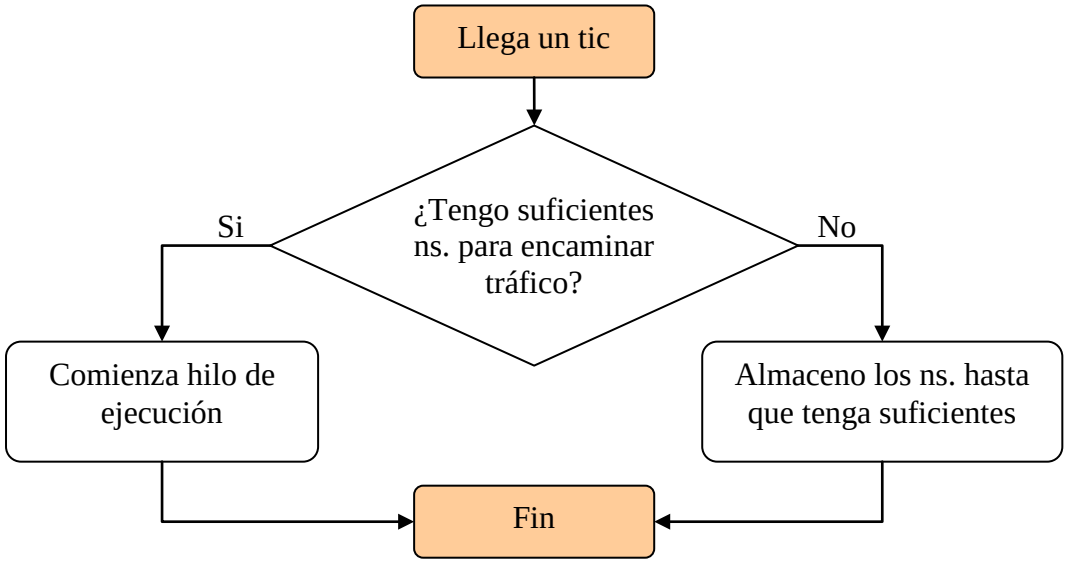
- **Averigua el tamaño del siguiente paquete** a generar, para lo cual se consultan los atributos internos de configuración del emisor. Esto lo hace el método **generarTamanioSiguientePaquiete()**.

- **Comprueba si tiene nanosegundos** para generar ese paquete. Esto lo hace el método **obtenerOctetosTransmitibles()**.
- Mientras que lo anterior se cumpla, **genera el paquete y lo envía al enlace** del puerto. Esto lo hacen los métodos **crearPaquete()** y **ponerTamanio()** y se envían mediante los métodos de **TPuertosNodo** que ya hemos comentado anteriormente.

En cada caso emite los eventos necesarios al recolector de eventos de simulación, que los recibirá y los mostrará en el panel de simulación.

3.3.3.13. Funcionamiento del nodo LER

Cuando el reloj del simulador se ponen en marcha y llegan tics de reloj al nodo LER, se pone en funcionamiento un proceso complejo por su densidad, pero sencillo de entender; el siguiente diagrama lo explica bien.



El proceso de llegar un tic ocurre cuando el reloj del simulador invoca al método **capturarEventoReloj()** del nodo, como ya se explicó. Este método actualiza un atributo interno que almacena el montante de nanosegundos de los que dispone el nodo y llama al método **iniciar()** que ponen en funcionamiento el hilo del nodo. Esto hace que se ejecute automáticamente el método **run()** que todo elemento tiene y es por fin en este método donde se codifica toda la funcionalidad del LER.

El método **run()** realiza las siguientes operaciones en un bucle que finaliza cuando no quedan nanosegundos suficientes para operar:

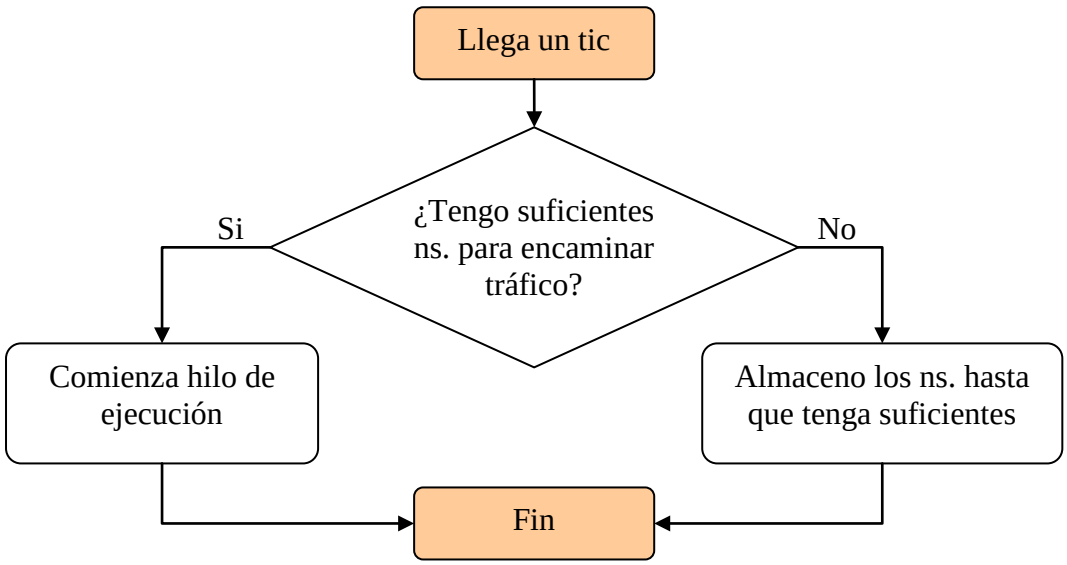
- **Calcula el número de octetos que puede encaminar**, para lo cual se consulta el atributo de nanosegundos acumulados. Esto lo hace el método **obtenerOctetosTransmitibles()**.
- **Comprueba si puede conmutar el paquete que espera en el búfer.** Esto se realiza consultando al conjunto de puertos, como se unos apartados atrás.
- Mientras que lo anterior se cumpla, **leo el paquete del puerto**, con el método apropiado del puerto, se detecta qué tipo de paquete es, consultando al método del paquete que se comentó anteriormente y se llama a alguno de los métodos para conmutar cada tipo de paquete: **conmutarTLDP()**, **conmutarMPLS()**, **conmutarIPv4()** o **conmutarGPSRP()**, del nodo.
- Los métodos de conmutación saben ya que tiene ns. suficientes para conmutar; ahora se comprueba si se puede conmutar y si no, emprender las acciones para que se pueda, como iniciar señalización TLDP, por ejemplo. Suponiendo que si pueden conmutar, **calculan a dónde deben dirigir el tráfico**, consultando la matriz de conmutación (que se habrá calculado con un algoritmo Floyd disponible en la topologia) **y ponen el paquete en el respectivo enlace** usando los métodos del puerto de salida correspondiente, como hemos comentado también.

Para calcular si se puede conmutar o no, además de averiguar la procedencia del paquete el LER consulta su matriz de conmutación/encaminamiento y comprueba si para ese paquete tiene una entrada. Más adelante se comenta el funcionamiento de las matrices de conmutación, pero valga decir que todos los nodos a excepción de los nodos receptores y los nodos emisores, tienen una como atributo. La matriz de conmutación es de tipo **TMatrizConmutacion**.

En cada caso emite los eventos necesarios al recolector de eventos de simulación, que los recibirá y los mostrará en el panel de simulación.

3.3.3.14. Funcionamiento del nodo LERA

Cuando el reloj del simulador se ponen en marcha y llegan tics de reloj al nodo LERA, se pone en funcionamiento un proceso complejo por su densidad, pero sencillo de entender; el siguiente diagrama lo explica bien.



El proceso de llegar un tic ocurre cuando el reloj del simulador invoca al método **capturarEventoReloj()** del nodo, como ya se explicó. Este método actualiza un atributo interno que almacena el montante de nanosegundos de los que dispone el nodo y llama al método **iniciar()** que ponen en funcionamiento el hilo del nodo. Esto hace que se ejecute automáticamente el método **run()** que todo elemento tiene y es por fin en este método donde se codifica toda la funcionalidad del LERA.

El método **run()** realiza las siguientes operaciones en un bucle que finaliza cuando no quedan nanosegundos suficientes para operar:

- **Calcula el número de octetos que puede encaminar**, para lo cual se consulta el atributo de nanosegundos acumulados. Esto lo hace el método **obtenerOctetosTransmitibles()**.
- **Comprueba si puede conmutar el paquete que espera en el búfer.** Esto se realiza consultando al conjunto de puertos, como se dijo unos apartados atrás.

- Mientras que lo anterior se cumpla, **leo el paquete del puerto**, con el método apropiado del puerto, se detecta qué tipo de paquete es, consultando al método del paquete que se comentó anteriormente y se llama a alguno de los métodos para conmutar cada tipo de paquete: **conmutarTLDP()**, **conmutarMPLS()**, **conmutarIPv4()** o **conmutarGPSRP()**, del nodo.
- Los métodos de conmutación saben ya que tiene ns. suficientes para conmutar; ahora se comprueba si se puede conmutar y si no, emprender las acciones para que se pueda, como iniciar señalización TLDP, por ejemplo. Suponiendo que si pueden conmutar, **calculan a dónde deben dirigir el tráfico**, consultando la matriz de conmutación (que se habrá calculado con mediante un algoritmo RABAN, disponible en la topología) **y ponen el paquete en el respectivo enlace** usando los métodos del puerto de salida correspondiente, como hemos comentado también.
- Además de lo anterior, como un LERA es un nodo activo, debe **comprobar si debe almacenar el paquete en su memoria DMGP**, si ya existe una entrada para ese flujo, si cabe, etcétera. Esto se realiza haciendo uso de los métodos de la DMGP.

En cualquier momento, independientemente de que el hilo de ejecución del nodo esté activo o no, el algoritmo EPCD de uno de los puertos activos del nodo podría comenzar un proceso de recuperación de paquetes que implica la emisión de tráfico GPSRP por parte de este nodo. Debe quedar claro que no es un proceso que se inicie en el método **run()** del nodo sino que se hace de forma el método **protocoloEPCD()** el puerto activo cuando detecta una pérdida de paquete GoS, puesto que el puerto de un nodo tiene siempre acceso al nodo padre a través de una referencia cruzada (una referencia al nodo) que se encuentra en la clase **TPuertosNodo**.

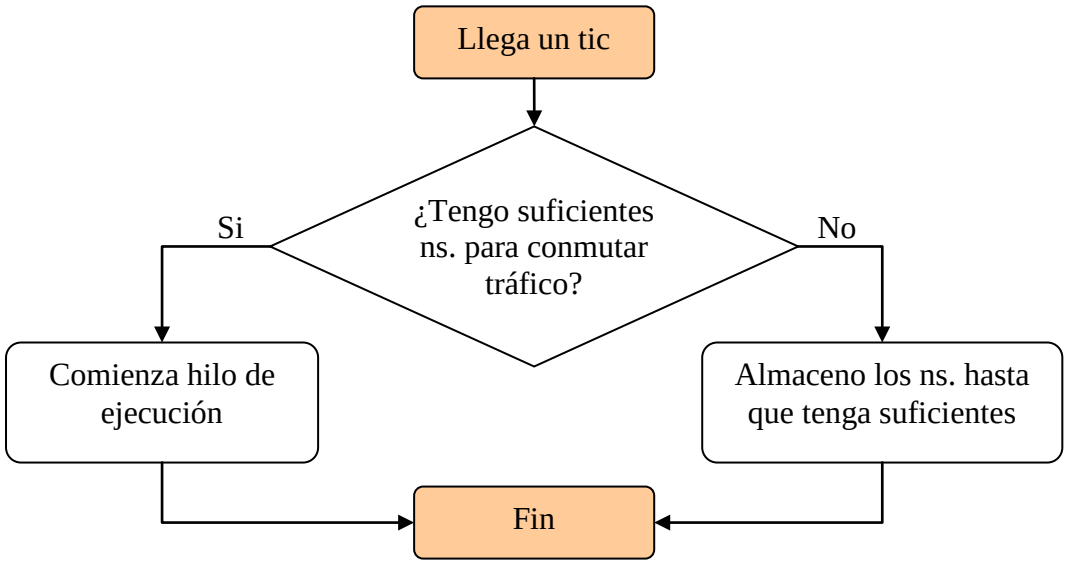
Para calcular si se puede conmutar o no, además de averiguar la procedencia del paquete el LERA consulta su matriz de conmutación/encaminamiento y comprueba si para ese paquete tiene una entrada. Más adelante se comenta el funcionamiento de las matrices de conmutación, pero valga decir que todos los nodos a excepción de los nodos receptores y los nodos emisores, tienen una como atributo. La matriz de conmutación es de tipo **TMatrizConmutacion**.

La memoria DMGP se explicará más adelante. Simplemente decir que sólo los nodos activos **TNodoLERA** y **TNodoLSRA** tienen una memoria DMGP como un atributo. La memoria DMGP se encuentra definida en la clase TDMGP.

En cada caso emite los eventos necesarios al recolector de eventos de simulación, que los recibirá y los mostrará en el panel de simulación.

3.3.3.15. Funcionamiento del nodo LSR

Cuando el reloj del simulador se ponen en marcha y llegan tics de reloj al nodo LSR, se pone en funcionamiento un proceso complejo por su densidad, pero sencillo de entender; el siguiente diagrama lo explica bien.



El proceso de llegar un tic ocurre cuando el reloj del simulador invoca al método **capturarEventoReloj()** del nodo, como ya se explicó. Este método actualiza un atributo interno que almacena el montante de nanosegundos de los que dispone el nodo y llama al método **iniciar()** que ponen en funcionamiento el hilo del nodo. Esto hace que se ejecute automáticamente el método **run()** que todo elemento tiene y es por fin en este método donde se codifica toda la funcionalidad del LSR.

El método **run()** realiza las siguientes operaciones en un bucle que finaliza cuando no quedan nanosegundos suficientes para operar:

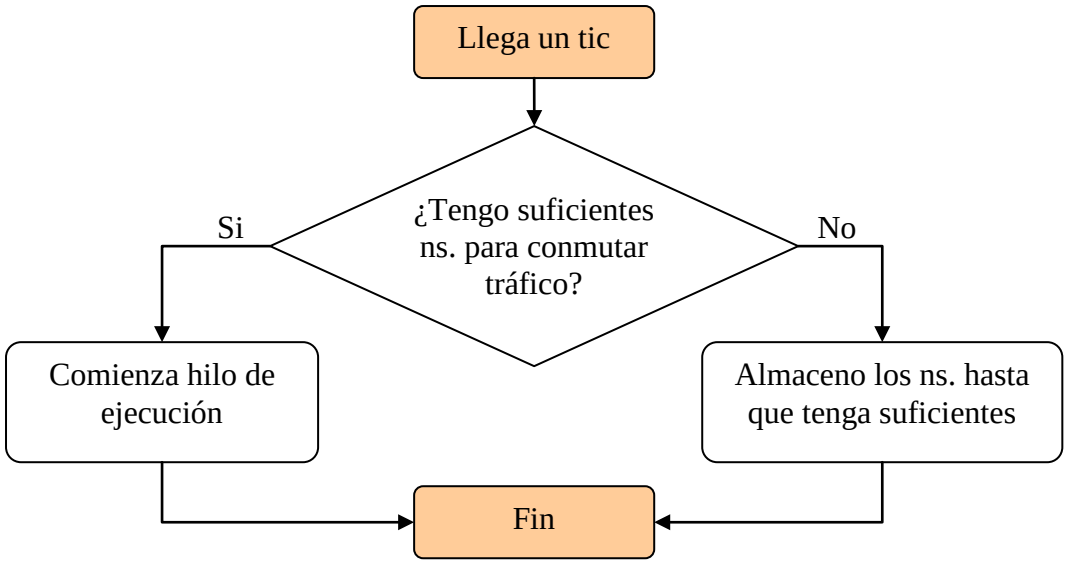
- **Calcula el número de octetos que puede conmutar**, para lo cual se consulta el atributo de nanosegundos acumulados. Esto lo hace el método **obtenerOctetosTransmitibles()**.
- **Comprueba si puede conmutar el paquete que espera en el búfer.** Esto se realiza consultando al conjunto de puertos, como se dijo unos apartados atrás.
- Mientras que lo anterior se cumpla, **leo el paquete del puerto**, con el método apropiado del puerto, se detecta qué tipo de paquete es, consultando al método del paquete que se comentó anteriormente y se llama a alguno de los métodos para conmutar cada tipo de paquete: **conmutarTLDP()**, **conmutarMPLS()** o **conmutarGPSRP()**, del nodo.
- Los métodos de conmutación saben ya que tiene ns. suficientes para conmutar; ahora se comprueba si se puede conmutar y si no, emprender las acciones para que se pueda, como iniciar señalización TLDP, por ejemplo. Suponiendo que si pueden conmutar, **calculan a dónde deben dirigir el tráfico**, consultando la matriz de conmutación (que se habrá calculado con mediante un algoritmo Floyd, disponible en la topología) **y ponen el paquete en el respectivo enlace** usando los métodos del puerto de salida correspondiente, como hemos comentado también.

Para calcular si se puede conmutar o no, además de averiguar la procedencia del paquete el LSR consulta su matriz de conmutación/encaminamiento y comprueba si para ese paquete tiene una entrada. Más adelante se comenta el funcionamiento de las matrices de conmutación, pero valga decir que todos los nodos a excepción de los nodos receptores y los nodos emisores, tienen una como atributo. La matriz de conmutación es de tipo **TMatrizConmutacion**.

En cada caso emite los eventos necesarios al recolector de eventos de simulación, que los recibirá y los mostrará en el panel de simulación.

3.3.3.16. Funcionamiento del nodo LSRA

Cuando el reloj del simulador se ponen en marcha y llegan tics de reloj al nodo LSRA, se pone en funcionamiento un proceso complejo por su densidad, pero sencillo de entender; el siguiente diagrama lo explica bien.



El proceso de llegar un tic ocurre cuando el reloj del simulador invoca al método **capturarEventoReloj()** del nodo, como ya se explicó. Este método actualiza un atributo interno que almacena el montante de nanosegundos de los que dispone el nodo y llama al método **iniciar()** que ponen en funcionamiento el hilo del nodo. Esto hace que se ejecute automáticamente el método **run()** que todo elemento tiene y es por fin en este método donde se codifica toda la funcionalidad del LSRA.

El método **run()** realiza las siguientes operaciones en un bucle que finaliza cuando no quedan nanosegundos suficientes para operar:

- **Calcula el número de octetos que puede encaminar**, para lo cual se consulta el atributo de nanosegundos acumulados. Esto lo hace el método **obtenerOctetosTransmitibles()**.
- **Comprueba si puede conmutar el paquete que espera en el búfer.** Esto se realiza consultando al conjunto de puertos, como se dijo unos apartados atrás.
- Mientras que lo anterior se cumpla, **leo el paquete del puerto**, con el método apropiado del puerto, se detecta qué tipo de paquete es, consultando al método del

paquete que se comentó anteriormente y se llama a alguno de los métodos para conmutar cada tipo de paquete: **conmutarTLDP()**, **conmutarMPLS()**, **conmutarIPv4()** o **conmutarGPSRP()**, del nodo.

- Los métodos de conmutación saben ya que tiene ns. suficientes para conmutar; ahora se comprueba si se puede conmutar y si no, emprender las acciones para que se pueda, como iniciar señalización TLDP, por ejemplo. Suponiendo que si pueden conmutar, **calculan a dónde deben dirigir el tráfico**, consultando la matriz de conmutación (que se habrá calculado con mediante un algoritmo RABAN, disponible en la topología) **y ponen el paquete en el respectivo enlace** usando los métodos del puerto de salida correspondiente, como hemos comentado también.
- Además de lo anterior, como un LERA es un nodo activo, debe **comprobar si debe almacenar el paquete en su memoria DMGP**, si ya existe una entrada para ese flujo, si cabe, etcétera. Esto se realiza haciendo uso de los métodos de la DMGP.

En cualquier momento, independientemente de que el hilo de ejecución del nodo esté activo o no, el algoritmo EPCD de uno de los puertos activos del nodo podría comenzar un proceso de recuperación de paquetes que implica la emisión de tráfico GPSRP por parte de este nodo. Debe quedar claro que no es un proceso que se inicie en el método **run()** del nodo sino que se hace de forma el método **protocoloEPCD()** el puerto activo cuando detecta una pérdida de paquete GoS, puesto que el puerto de un nodo tiene siempre acceso al nodo padre a través de una referencia cruzada (una referencia al nodo) que se encuentra en la clase **TPuertosNodo**.

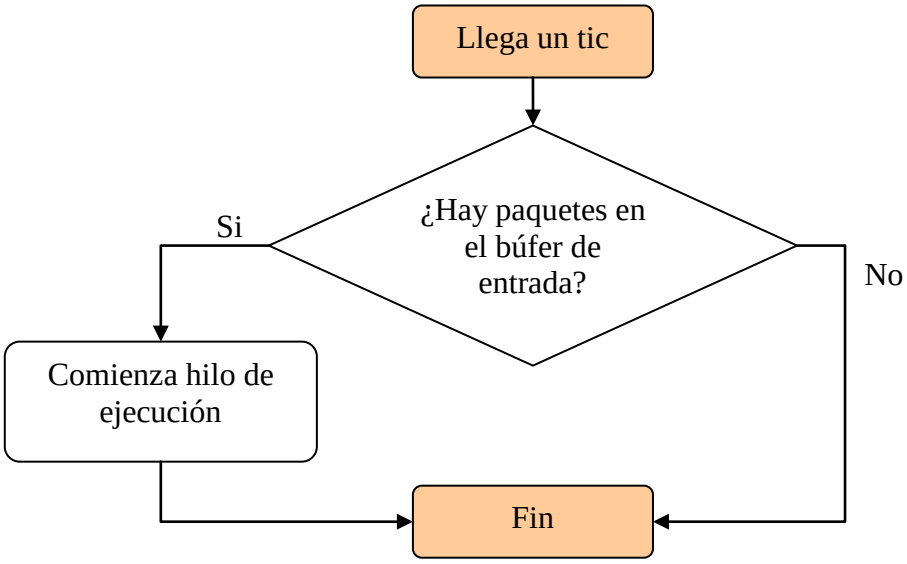
Para calcular si se puede conmutar o no, además de averiguar la procedencia del paquete el LSRA consulta su matriz de conmutación/encaminamiento y comprueba si para ese paquete tiene una entrada. Más adelante se comenta el funcionamiento de las matrices de conmutación, pero valga decir que todos los nodos a excepción de los nodos receptores y los nodos emisores, tienen una como atributo. La matriz de conmutación es de tipo **TMatrizConmutacion**.

La memoria DMGP se explicará más adelante. Simplemente decir que sólo los nodos activos **TNodoLERA** y **TNodoLSRA** tienen una memoria DMGP como un atributo. La memoria DMGP se encuentra definida en la clase TDMGP.

En cada caso emite los eventos necesarios al recolector de eventos de simulación, que los recibirá y los mostrará en el panel de simulación.

3.3.3.17. Funcionamiento del nodo receptor

Cuando el reloj del simulador se ponen en marcha y llegan tics de reloj al nodo receptor, se pone en funcionamiento un proceso sencillo; el siguiente diagrama lo explica bien.



El proceso de llegar un tic ocurre cuando el reloj del simulador invoca al método **capturarEventoReloj()** del nodo, como ya se explicó. Este método actualiza un atributo interno que almacena el montante de nanosegundos de los que dispone el nodo y llama al método **iniciar()** que ponen en funcionamiento el hilo del nodo. Esto hace que se ejecute automáticamente el método **run()** que todo elemento tiene y es por fin en este método donde se codifica toda la funcionalidad del receptor.

El método **run()** realiza las siguientes operaciones en un bucle que finaliza cuando no quedan nanosegundos suficientes para operar:

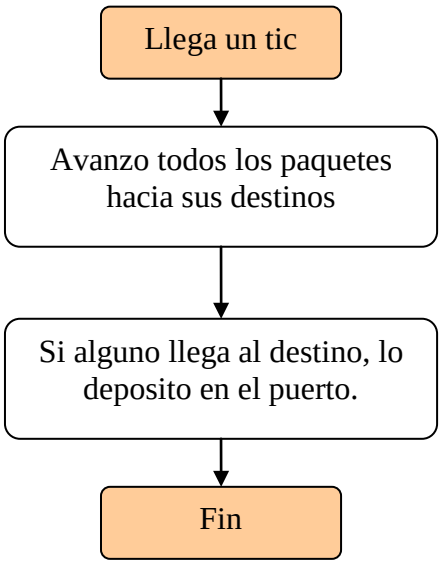
- **Comprueba si hay paquetes en el puerto de entrada**, para lo cual se hace uso del método **hayPaquetesEsperando()** de la clase **TPuertoNormal**.

- **Si hay paquetes, y mientras haya, recibe los datos**; esto es, los extrae del puerto. Para ello utiliza el método apropiado y que ya comentamos, para leer datos de un puerto.

En cada caso emite los eventos necesarios al recolector de eventos de simulación, que los recibirá y los mostrará en el panel de simulación.

3.3.3.18. Funcionamiento de los enlaces

A efectos de funcionamiento, los enlaces internos y externos son similares. Cuando el reloj del simulador se ponen en marcha y llegan tics de reloj al enlace, se pone en funcionamiento un proceso sencillo; el siguiente diagrama lo explica bien.



El proceso de llegar un tic ocurre cuando el reloj del simulador invoca al método **capturarEventoReloj()** del enlace, como ya se explicó. Este método actualiza un atributo interno que almacena el montante de nanosegundos de los que dispone el enlace y llama al método **iniciar()** que ponen en funcionamiento el hilo del enlace. Esto hace que se ejecute automáticamente el método **run()** que todo elemento tiene y es por fin en este método donde se codifica toda la funcionalidad del enlace.

El método **run()** realiza las siguientes operaciones en un bucle que finaliza cuando no quedan nanosegundos suficientes para operar:

- **Decrementa el tiempo que le queda a los paquetes para llegar al destino**, que básicamente consiste a descontar del tiempo total que deben estar cada paquete en el enlace (coincide con el retardo), el número de nanosegundos ofrecidos por el tic, para lo cual se hace uso del método **adelantarPaquetesEnTransito()**.
- Si tras la operación anterior alguno de los paquetes ya ha pasado el tiempo que le correspondía en el enlace, **se pasan los paquetes que lo requieran al puerto del nodo destino**; esto es, se eliminan del enlace y se insertan el puerto. Para ello se hace uso del método **pasarPaquetesADestino()**.

En cada caso emite los eventos necesarios al recolector de eventos de simulación, que los recibirá y los mostrará en el panel de simulación.

3.3.3.19. Funcionamiento de la matriz de conmutación

La matriz de conmutación está definida en las clases **TMatrizConmutacion** y **TEntradaMatrizConmutacion** es la unión de todas las tablas que define el IETF para MPLS, esto es, ILM (*Incoming Label Map*), FTN (*FEC to NHLFE*) y NHLFE (*Next Hop Label Forwarding Entry*). Además incorpora campos propios para la propuesta RLPRP, que serán utilizados exclusivamente por los nodos activos.

TEntradaMatrizConmutacion tiene diversos atributos, pero los principales son estos:

- Una entrada que almacena el **identificador de sesión TLDP del nodo antecesor** y que permitirá el mantenimiento de la sesión TLDP “hacia atrás”. Es un número entero y su nombre es **idLDPAntecesor.**
- Una entrada que almacena el **identificador de sesión TLDP propio** y que permitirá el mantenimiento de la sesión TLDP “hacia adelante”. Es un número entero y su nombre es **idLDPPropio.**
- El **puerto de entrada** por el que deben llegar los paquetes para los cuales se creó esta entrada. Es un número entero y su nombre es **pEntrada**.

- Una entrada que almacena la **etiqueta o el FEC** (dependiendo de si la entrada se refiere a tráfico etiquetado o sin etiquetar) del tráfico que llegará por el puerto de entrada. Su nombre es **labelFEC** y es un entero.
- Un atributo que indicará el **puerto que debe usarse para reenviar el tráfico** que entre con las características indicadas. Es un entero y se llama **pSalida**.
- Un atributo que indica **la etiqueta de salida** que se debe poner al paquete antes de reenviarlo por el puerto de salida. Es un entero y se llama **label**.
- Un atributo que almacena la **operación que se debe realizar sobre la cima de la pila** de etiquetas del paquete entrante. Puede ser cambiar, dejar tal cual, eliminar, añadir, etcétera. Es un entero y se llama **operacion**.
- Un atributo que indicará el **puerto que debe usarse para reenviar el tráfico** que entre con las características indicadas cuando se esté haciendo uso de un camino de respaldo. Es un entero y se llama **pSalidaBackup**.
- Un atributo que indica la **etiqueta de salida** que se debe poner al paquete antes de reenviarlo por el puerto de salida de respaldo. Es un entero y se llama **labelBackup**.

Con todos estos datos, una entrada de la matriz de conmutación permite controlar en todo momento qué se debe hacer con el tráfico que entra. Esta entrada es la que establece que cuando un paquete llega sin etiquetar, hay que etiquetarlo, que cuando un paquete sale del dominio MPLS hay que quitarle la etiqueta, etcétera.

Uno de los métodos más importantes de la clase **TEntradaMatrizConmutacion** es **conmutarLSPPrincpalALSPBackup()** que traspasa los valores de **labelBackup** y **puestoSalidaBackup** a **label** y **puertoSalida**, respectivamente. De esta forma el tráfico es reencaminado por el LSP de respaldo. Al ocurrir esta conmutación **labelBackup** y **puertoSalidaBackup** se dejan sin especificar para que se vuelva a calcular un nuevo LSP de respaldo. El nodo llamará a este método cuando se reciba una eliminación de etiqueta por la caída física de un enlace.

Con respecto a la clase **TMatrizConmutacion**, los métodos que implementa sirven básicamente para realizar búsquedas de tipo **TEntradaMatrizConmutacion** por algún criterio concreto: por identificador TLDP, por puertos de salida, etcétera. En el interior,

contiene un árbol en el que se almacenan todas las entradas que se van creando y que todas juntas formarán la tabla o matriz de encaminamiento/conmutación.

3.3.3.20. Funcionamiento de la DMGP

La DMGP (*Dynamic Memory for GoS PDU's*) se implementa mediante varias clases distintas: **TDMGP**, **TEntradaDMGP** y **TEntradaFlujoDMGP**. La arquitectura interna de la memoria DMGP es un poco compleja, así que la desglosaré poco a poco.

La clase DMGP tiene una lista interna de flujos que realmente son **TEntradaFlujoDMGP**. Cada uno de estos flujos representa un flujo de paquetes que entra por el nodo y para el cual hay que reservar, si es necesario, tamaño de DMGP. Así pues, antes de intentar almacenar paquetes en la DMGP hay que "dar de alta" el flujo al que pertenece dicho paquete y una vez que se tenga la seguridad de que se le ha asignado espacio, podrá comenzar a almacenarse paquetes. Cada paquete almacenado en un flujo no se almacena tal cual, sino que se recubre dentro de una instancia **TEntradaDMGP**; Así que la estructura que tenemos es de cubo. Primero la clase TDMGP tiene una lista de **TEntradaFlujoDMGP** y luego cada flujo tiene una lista de **TEntradaDMGP**, una por cada paquete de ese flujo almacenado.

Entonces harán falta distintas operaciones:

- Buscar un flujo.
- Crear un flujo nuevo.
- Insertar un paquete en un flujo.
- Buscar un paquete de un flujo.
- Etcétera.

Pero en este caso, dado que trabajar de esta manera es muy tedioso, se ha implementado toda la funcionalidad en la clase **TDMGP** y es ésta la que maneja directamente los métodos de las otras clases.

TDMGP tiene tres métodos que serán los que principalmente se usarán:

- **ponerTamanioDMGPEnKB()**, que permite establecer el tamaño de la memoria DMGP en KB.
- **insertarPaquete()**, que permite insertar un paquete en DMGP si puede ser insertado, y que no hace nada si el paquete no puede ser insertado por la razón que sea. Si no hay un flujo dado de alta para el paquete MPLS especificado como parámetro se intenta crear uno lo cual puede terminar con un flujo dado de alta o no. Si no se puede dar de alta un flujo, el paquete no se inserta en DMGP. Si se puede crear un flujo para ese paquete, si se inserta.
- **buscarPaquete()**, permite la búsqueda de un paquete. Busca si hay un flujo asociado al paquete que se desea buscar y si es así busca dentro de él el paquete deseado y lo devuelve. Sino existe dicho paquete en la DMGP, devuelve null.

Cada flujo **TEntradaFlujoDMGP** tiene un atributo iniciado en el momento de su creación que almacena el número de octetos máximos que se pueden ocupar con paquetes de ese flujo. Si se supera, habrá que sacar paquetes antiguos para poder insertar paquetes nuevos.

Cada entrada **TEntradaDMGP** además del paquete almacenado, guarda, entre otras cosas el identificador único y global que tienen todos los paquetes MPLS/IP marcados con GoS. De esta forma se puede localizar un paquete cuando se busquen.

3.3.4. Guía de instalación y puesta en marcha

A continuación se detalla el procedimiento que hay que seguir para instalar Open SimMPLS 1.0 en su PC y dejarlo preparado para usarlo cuando lo necesite.

3.3.4.1. Instalación de Open SimMPLS 1.0

Open SimMPLS 1.0, en su versión *standalone* se distribuye como una aplicación JAR autocontenida. Su instalación por tanto no requiere de ningún paso significativo, y simplemente hay que invocar su ejecución mediante la Máquina Virtual Java de SUN que

debe estar instalado correctamente en el PC. Lo más cómo, por tanto, será copiar el fichero *openSimMPLS.jar* en la carpeta que se desee y la instalación habrá concluido.

Ejemplo en Windows, copiando Open SimMPLS desde el CD

```
md c:\openSimMPLS-1.0
copy d:\openSimMPLS c:\openSimMPLS-1.0\openSimMPLS
```

3.3.4.2. Ejecución de Open SimMPLS

Una vez que hemos "instalado" correctamente la aplicación en el directorio o carpeta que hemos elegido, la ejecución es muy sencilla. Partimos de la base de que JRM se encuentra bien instalado tal y como hemos visto en páginas atrás. En caso afirmativo, debemos tener sin problemas acceso a la Máquina Virtual Java que se invoca mediante el comando `java`. En este caso, deberemos asegurarnos que nos encontramos en un terminal que tiene capacidades gráficas; actualmente todos los terminales Linux/UNIX iniciados desde una sesión X tienen esta capacidad. La consola de Windows también.

Una vez que cumplimos estos requisitos, tecleamos el comando:

Ejecución de openSimMPLS desde un terminal con capacidad gráfica

```
java –jar openSimMPLS.jar
```

Para que esta orden funcione correctamente, debemos asegurarnos de que estamos situados en el directorio o carpeta donde hemos instalado Open SimMPLS. Si todo funciona correctamente, aparecerá la pantalla que se muestra a continuación.

Que, indica que la aplicación está cargándose. En este momento se está creando la interfaz de usuario y se están cargando las imágenes que luego permitirán que el simulador se ejecute con mayor fluidez.

> *NOTA: En equipos muy rápidos esta pantalla puede aparecer y desaparecer tan rápidamente que no de tiempo a verla. Esto es normal y no hay que preocuparse; más bien todo lo contrario.*

Transcurrido el tiempo necesario para la carga previa de imágenes, la pantalla anterior desaparecerá automáticamente y en su lugar se mostrará ahora, ocupando totalmente el monitor del PC, la interfaz principal de Open SimMPLS 1.0, que puede verse a continuación.

3.3.5. Manual de usuario

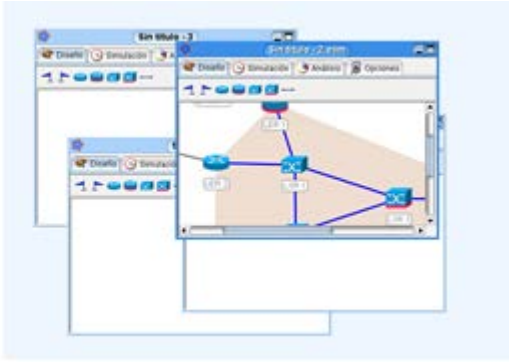
Una vez tenemos Open SimMPLS 1.0 en funcionamiento, y aunque es muy intuitivo, es necesario explicar todas las partes que lo componen, su funcionamiento, su modo de trabajo, etcétera.

3.3.5.1. Familiarización con el entorno de trabajo

El entorno de trabajo del simulador se ha diseñado muy minimalista, buscando que sea sencillo de utilizar. Se divide en el área de trabajo, el menú principal y las ventanas de los escenarios abiertos.

3.3.5.1.1. Área de trabajo

Es el rectángulo mayor de la ventana principal que aparece al arrancar el simulador. En un principio no contiene nada, pero posteriormente aparecerán en ella las ventanas de escenarios que haya abiertas.



3.3.5.1.2. Menú principal

El menú principal está situado en la parte superior de la ventana que se abre nada más iniciar el simulador. En él se encuentran todas las opciones que tienen que ver con el funcionamiento general de la aplicación.

Las opciones del menú principal están agrupadas en tres categorías:

- **Escenario:** que contiene las acciones que tienen que ver con los escenarios, tales como abrir, cerrar, guardar, crear un escenario nuevo, etcétera.
- **Vista:** que contiene las acciones que tienen que ver con la forma en que se mostraran los distintos escenarios abiertos en el área de trabajo, como por ejemplo, minimizar, cascada, mosaico, etcétera.
- **Ayuda:** donde están las acciones que permiten al usuario obtener información adicional; por ejemplo, contenidos de ayuda, contactar con los autores, etcétera.

En las siguientes páginas se irán desglosando todas las opciones del menú principal de una forma clara y concisa.

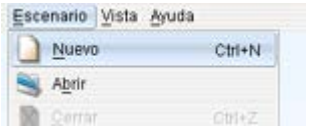
3.3.5.1.3. Ventana de escenarios

Las ventanas de escenarios son aquellas que se incrustarán sobre el área de trabajo. Puede haber diversas y cada una de ellas contiene todo lo necesario para realizar la simulación completa de un escenario propuesto.

En las siguientes líneas se trabajará profundamente con estas ventanas por lo que su uso será aprendido sin problemas.

3.3.5.2. Crear un nuevo escenario

Cuando queramos crear un nuevo escenario de simulación, debemos acudir al menú principal, concretamente a la opción escenario.



Como vemos en la figura, podemos seleccionar la opción **"Nuevo",** que nos permitirá crear un nuevo escenario, que es lo que deseamos. Junto a la opción **"Nuevo",** aparece una combinación de teclas, un atajado de teclado **"Ctrl+N"**. Esto significa que sin necesidad de desplegar nada en el menú principal, y desde cualquier lugar de la ventana principal, podemos crear un escenario pulsando simultáneamente las teclas **"Control"** y **"N"** del teclado.

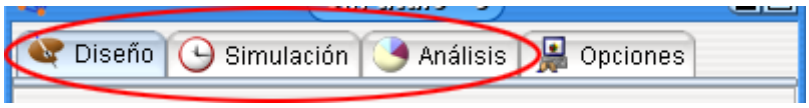
Una vez seleccionada la opción, de cualquiera de los modos comentados, se abrirá una nueva ventana de escenario en el área de trabajo que se añadirá a las que ya pudiesen existir.

3.3.5.3. Modo de trabajo de Open SimMPLS

Para simplificar las tareas de simulación y el uso del simulador, Open SimMPLS 1.0 trabaja en tres modos distintos con cada escenario:

- **Modo diseño:** donde se podrán hacer todas las labores de diseño de topologías y configuración de los elementos de la red que queremos simular.
- **Modo simulación:** donde se podrá realizar la simulación en tiempo real del funcionamiento de la red diseñada.
- **Modo análisis:** donde se podrán ver gráficas analíticas sobre lo que ocurre en la simulación.

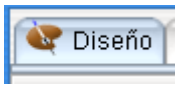
Y para que resulte sencillo llevar a cabo las tareas comentadas, las ventanas de escenario se dividen precisamente en estas tres áreas, mediante pestañas de separación.



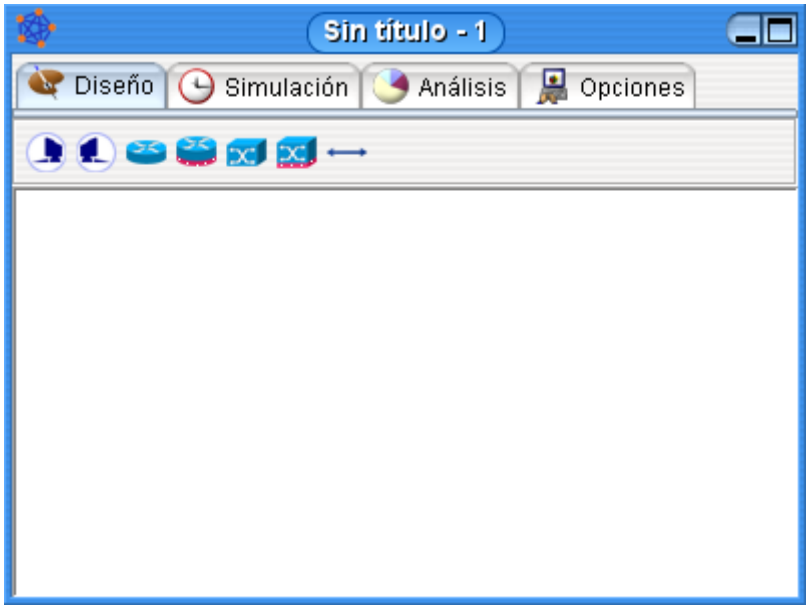
Además, la ventana de escenario incorpora una cuarta pestaña donde se deben configurar aspectos generales de la simulación.

3.3.5.4. Área de diseño

Debemos seleccionar trabajar el área de diseño cuando deseemos modificar el aspecto de la topología de la red, añadir elementos, eliminar elementos, configurarlos, etcétera. Para seleccionar este modo de trabajo, se debe hacer clic sobre la pestaña **"Diseño"** de la ventana de escenario.



Y en ese momento la ventana de escenario mostrará el siguiente aspecto:



Y desde ese momento, se estará trabajando en modo diseño. A partir de ahora todo lo que se haga mientras que no se cambie de modo de trabajo, tendrá que ver con el aspecto y configuración de la red que queremos simular.

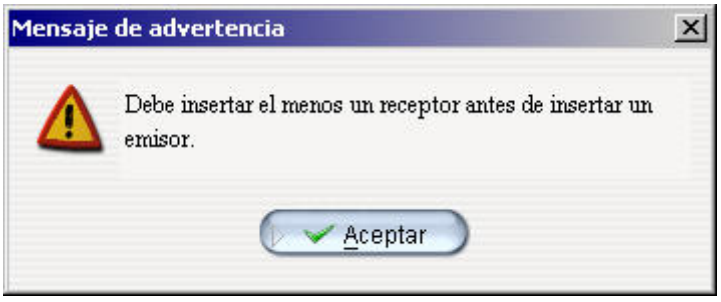
3.3.5.4.1. Insertar elementos nuevos

Lo principal a la hora de crear un nuevo escenario es insertar y configurar los elementos de la red que queremos simular. Para ello, la ventana de escenario, en la pestaña de diseño, incorpora un icono para cada uno de los elementos que permite insertar Open SimMPLS 1.0, como se muestra en la figura siguiente. A continuación se explica cómo insertar y, lo más importante, cómo configurar cada uno de ellos.

3.3.5.4.1.1. Emisor de tráfico

Un emisor de tráfico es el nodo que genera tráfico de red en el simulador. Siempre que se simula una red, además de la topología hay que especificar quién genera tráfico y quién lo recibe puesto que si no, la simulación no haría nada. Para insertar un nodo emisor de tráfico hay que hacer clic sobre el correspondiente icono, señalado en la siguiente figura con un círculo rojo.

Tras lo cual pueden ocurrir dos cosas. La primera de ella es que aparezca un mensaje de error como se muestra en la siguiente figura.



Al insertar un emisor de tráfico siempre debe indicarse en su ventana de configuración, que a continuación detallaremos, el nodo receptor que va a ser destino de ese tráfico. Si se intenta insertar un emisor sin haber antes insertado un receptor, este dato no se puede especificar, por lo que Open SimMPLS 1.0 detecta automáticamente si hay receptores insertados ya en el escenario y si no es así, no permite insertar ningún emisor. Esto evita incoherencias a la hora de formar el escenario de simulación. Simplemente debemos hacer clic en el botón "Aceptar" e insertar un receptor antes de intentar crear un emisor que dirija tráfico a él.

Si por el contrario ya hemos insertado algún receptor de tráfico durante el diseño, no habrá problemas a la hora e intentar añadir un emisor y lo que nos aparecerá en la ventana de escenario será la pantalla de configuración del emisor, que se muestra a continuación.

La pantalla de configuración del emisor se compone de tres partes, separadas por pestañas:

- **General:** donde se especifican parámetros generales del nodo. Casi no varía entre los distintos nodos permitidos por Open SimMPLS 1.0
- **Rápida:** donde podemos hacer una configuración básica muy rápida y sencilla del emisor.
- **Avanzada:** donde podemos refinar mucho más la configuración del emisor a costa de dedicar más tiempo y tener más conocimientos.

No es necesario utilizar las dos últimas opciones siempre; basta con utilizar una de ellas para configurar correctamente el emisor.

La pestaña **"General"**, se activa haciendo clic sobre ella; aunque siempre que abre la pantalla de configuración, es esta pestaña la que aparece seleccionada por defecto.

En esta pestaña, que se muestra dos figuras atrás, aparece un icono que representa al nodo emisor en el escenario. Una vez que el nodo esté configurado e insertado, podremos identificarlo por este icono. Junto al icono aparece una pequeña descripción de qué hace este nodo; en este caso, generar tráfico de red. También aparece un campo **"Nombre del emisor"** donde se nos permite escribir. En él deberemos escribir un nombre para el emisor, nombre que servirá para referirse al él en todo el escenario y nombre que se mostrará, si se desea, en las ventanas de diseño y simulación. Justo esto último es lo que se puede configurar en el *checkbox* titulado **"Ver nombre"**. Si lo marcamos, el nombre del nodo que hemos escrito se mostrará en las distintas pantallas de la ventana de escenario. Si lo desmarcamos, no se verá. Por último, en esta pestaña aparece también una lista desplegable llamada **"Destino del tráfico"**.

Pulsando sobre ella, aparecerá una lista con todos los nodos receptores que hay ya insertados en la topología. En el caso del ejemplo sólo hay un nodo llamado **"Receptor 1"**, pero podría haber muchos. Deberemos seleccionar el nodo receptor de la lista al que el nodo emisor que estamos configurando, enviará el tráfico. Por simplicidad en la configuración, un emisor sólo puede enviar tráfico a un receptor, pero un receptor puede recibir tráfico de más de un emisor.

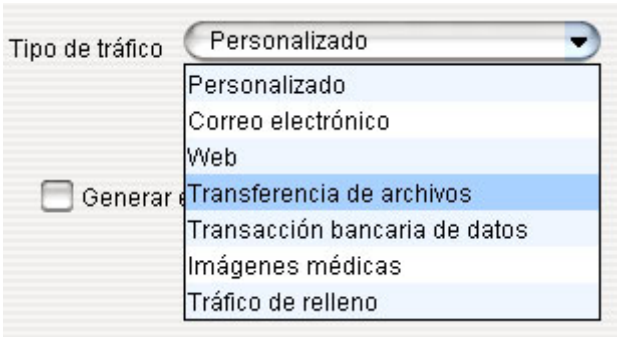
Además de lo anterior, aparece una zona rectangular, con una etiqueta que dice **"Selecciona la posición"**. Representa una imagen en miniatura del escenario que estamos

creando y se utiliza para seleccionar el lugar de la pantalla de diseño en el que queremos colocar el emisor. Se puede seleccionar el lugar haciendo clic sobre la zona de la miniatura y tendrá repercusión inmediata sobre la pantalla real en el momento de pulsar el botón **"Aceptar"**. Un pequeño círculo rojo indica en cada momento la zona aproximada de la pantalla de diseño en la que se insertará el nodo emisor, aunque posteriormente se podrá modificar dicha posición fácilmente.

La segunda pestaña de la pantalla de configuración, la pestaña **"Rápida"**, se activa haciendo clic sobre ella.

Y tras hacerlo aparecerá la ventana de configuración rápida del nodo emisor. Es fácil en todo momento saber en qué zona de la configuración nos encontramos. En la pestaña de configuración rápida, aparece la imagen de un mago, indicando que estamos en la zona donde se puede configurar todo de forma transparente, muy rápida y muy sencilla.

En el caso del emisor, la configuración rápida se limita a tener que seleccionar de la lista desplegable llamada **"Tipo de tráfico"**, el tipo de tráfico que queremos que genere el emisor. Son tipos de tráfico comunes, que pueden ser requeridos con cierta asiduidad y que por tanto están predefinidos para seleccionarlos de forma rápida.

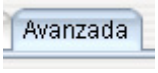


Se puede configurar desde aquí el emisor para generar tráfico de correo electrónico, tráfico web, tráfico de transferencias de archivos, tráfico de transacciones bancarias, tráfico de imágenes médicas o tráfico de relleno. El tipo de tráfico **"Personalizado"** no indica un tipo de tráfico en si, sino que aparece aquí cuando el tráfico a generar no se ha configurado en la pestaña **"Rápida"** sino en la pestaña **"Avanzadas"** y por tanto no coincide con alguno de estos tráficos tipo.

La última opción, llamada **"Generar estadísticas para este emisor"** hará, si se marca, que el simulador guarde toda la información necesaria para construir unas estadísticas completas del funcionamiento del emisor desde el inicio de la simulación hasta el final. Por defecto está deshabilitada. Si se deja sin marcar esta opción, no se podrán saber datos estadísticos del nodo emisor de ninguna de las maneras.

¡Nota! La generación de estadísticas para cualquiera de los elementos de la topología consume una ingente cantidad de recursos debido al gran volumen de información que maneja el simulador. Por defecto están desactivadas y se recomienda que deje esta opción así y la active exclusivamente en aquellos nodos que quiera estudiar en un momento determinado; si no, la fluidez de la simulación decaerá con el transcurso de la misma. Se recomienda que cuando esté interesado en la simulación visual desactive todas las estadísticas de los elementos; y cuando necesite estadísticas, no espere una simulación visual fluida en simulaciones medias-largas.

La tercera pestaña de la pantalla de configuración, la pestaña **"Avanzada"**, se activa haciendo clic sobre ella.



Y tras hacerlo aparecerá la ventana de configuración avanzada del nodo emisor. Es fácil saber en todo momento que nos encontramos en esta área porque aparece la imagen de una caja de herramientas, indicando que estamos en la zona donde se puede configurar todo mucho más detalladamente.

La configuración avanzada del emisor es la que más opciones tiene de todos los elementos que se pueden insertar en la topología del escenario. Esto es así porque el tráfico generado tiene muchas peculiaridades y todas ellas pueden ser configuradas desde esta ventana.

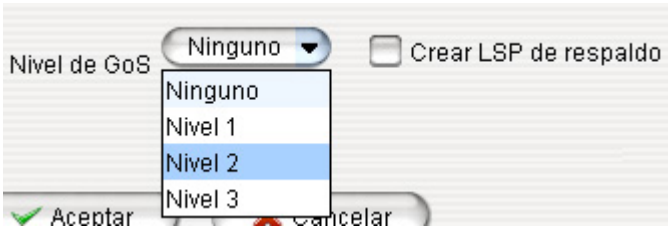
Lo primero que nos encontramos es un deslizador marcado como **"Tasa de tráfico"**, donde inicialmente se puede observar que está seleccionada la cantidad de 64 Mbps. Esta propiedad se refiere al número de bits que el emisor de tráfico es capaz de generar en un segundo y determinará la cantidad de tráfico, y por tanto paquetes, que el emisor puede enviar al nodo destino durante la simulación, la congestión que es capaz de causar al receptor, etcétera. Su unidad de medida es "Megabits por segundo (Mbps)" y los valores permitidos van desde 1 a 10240 Mbps, o lo que es lo mismo, desde 1 Mbps a 10 Gbps (Gigabits por segundo), pues el escenario a simular se supone que será una troncal de datos y los volúmenes de información serán grandes. La forma de variar el valor y seleccionar el que deseamos es moviendo el deslizador de izquierda a derecha o bien colocando el foco sobre el deslizador y usar las teclas de izquierda y derecha del teclado.

Lo siguiente que nos encontramos son unos botones de selección llamados **"Tipo de tráfico"** en los que podemos elegir si deseamos generar tráfico constante o variable. La forma de seleccionar una u otra opción es haciendo clic sobre ella. Nunca se pueden seleccionar ambas opciones simultáneamente y siempre hay una de ellas seleccionada. Si elegimos tráfico variable, el emisor generará aleatoria y automáticamente paquetes de tamaño variable de 0 octetos hasta 65535 siguiendo la distribución de tamaños de paquete real de Internet. Para ello se sigue el modelo estadístico ofrecido por la red Abilene (Internet2), donde se define la siguiente proporción:

Tamaño del paquete	Porcentaje del total
Menor de 100 octetos	47%
Entre 100 octetos y 1400 octetos (inclusive)	24%
Entre 1401 octetos y 1500 octetos (inclusive)	28%
Entre 1501 octetos y 65535 octetos (inclusive)	1%

Si elegimos la opción de generar tráfico constante, esto significa que los paquetes que el emisor generará serán siempre del mismo tamaño y que los generará siguiendo un periodo fijo de tiempo; este tráfico no es el real en Internet, pero es una muy buena opción para hacer escenarios de pruebas concretos, porque es tráfico predecible. Al elegir esta opción, automáticamente se activa la de más abajo, llamada **"Tamaño de la carga útil"** que aparecía desactivado (no se podía elegir) con la opción de tráfico variable. Ahora nos permitirá seleccionar el tamaño de la carga útil de los paquetes que el emisor debe generar; todos los paquetes serán iguales en tamaño. Se permiten valores entre 0 octetos y 65495 octetos y se selecciona el valor que deseamos moviendo el deslizador de izquierda a derecha o bien colocando el foco sobre el deslizador y usar las teclas de izquierda y derecha del teclado

La siguiente opción, por orden descendente, tiene el nombre de **"Encapsular tráfico sobre MPLS"**. Por defecto está desactivada y esto significa que el tráfico que generará el emisor será IPv4. Podemos simular que el tráfico proviene de otro dominio MPLS y por tanto es ya tráfico etiquetado y para esto es para lo que sirve esta opción. Marcándola haremos que el emisor genere tráfico MPLS, etiquetado por tanto, en lugar de tráfico IPv4.



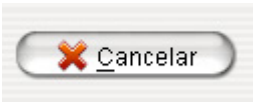
La siguiente opción, que se puede ver sobre estas líneas, nos permite especificar los parámetros de Garantía de Servicio (GoS) deseados para el tráfico. Está compuesta por una lista desplegable, donde podemos seleccionar el nivel de GoS deseado para el tráfico (en la práctica, la prioridad), y por otro lado un *checkbox* donde podremos decidir si deseamos que se nos proporcione un camino de respaldo para el transporte del tráfico. Existen tres niveles de GoS para el tráfico; el nivel 1 es el que menos prioridad da al tráfico (por encima, por supuesto, del tráfico sin GoS) y el tercero es el que genera tráfico más prioritario.

La última opción, llamada **"Generar estadísticas para este emisor"** hará, si se marca, que el simulador guarde toda la información necesaria para construir unas estadísticas completas del funcionamiento del emisor desde el inicio de la simulación hasta el final. Por defecto está deshabilitada. Si se deja sin marcar esta opción, no se podrán saber datos estadísticos del nodo emisor de ninguna de las maneras.

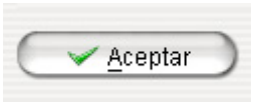
> *¡Nota! La generación de estadísticas para cualquiera de los elementos de la topología consume una ingente cantidad de recursos debido al gran volumen de información que maneja el simulador. Por defecto están desactivadas y se recomienda que deje esta opción así y la active exclusivamente en aquellos nodos que quiera estudiar en un momento determinado; si no, la fluidez de la simulación decaerá con el transcurso de la misma. Se recomienda que cuando esté interesado en la simulación visual desactive todas las estadísticas de los elementos; y cuando necesite estadísticas, no espere una simulación visual fluida en simulaciones medias-largas.*

Cualquier modificación que se haga en la pestaña de configuración **"Avanzada"** provocará que en la ventana **"Rápida"** aparezca el tipo de tráfico como **"Personalizado"**, pero no es una cosa de la que deba preocuparse.

Si en cualquier momento decide dar marcha atrás y no insertar finalmente el nodo emisor en el escenario, pulse el botón **"Cancelar"**. La ventana se cerrará y volverá al escenario que seguirá como antes de intentar insertar el emisor.

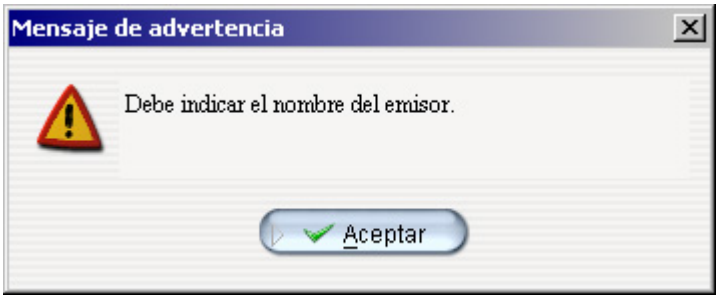


Una vez que haya terminado de configurar el nodo emisor como desee, pulse el botón **"Aceptar"** para añadir el nuevo nodo emisor al escenario.

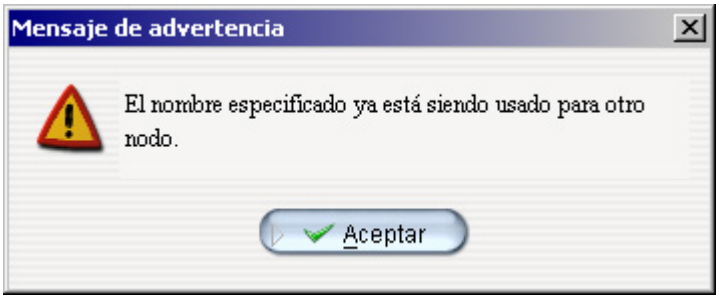


Si en este momento todo está correcto, la ventana de configuración del nodo emisor desaparecerá y volverá a la pantalla de diseño, donde ya se habrá insertado el nuevo nodo emisor correctamente.

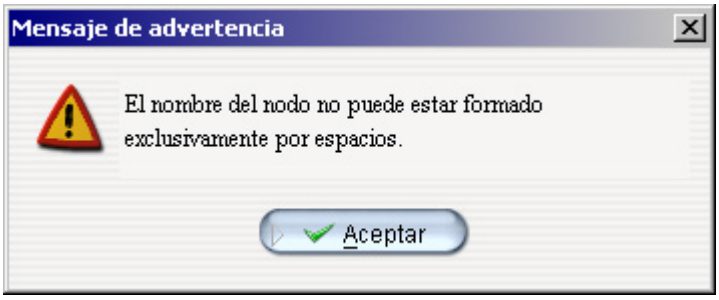
Sin embargo, puede que al pulsar el botón **"Aceptar"**, aparezca alguna ventana indicándole de un error, como vemos en las siguientes figuras.



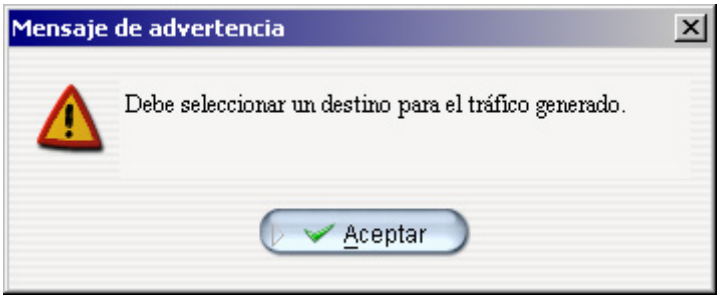
Esta ventana indica que se le ha olvidado escribir el nombre del emisor. Un emisor siempre tiene que tener un nombre, independientemente de que se desee mostrar o no en las pantallas de diseño y simulación.



Esta ventana indica que el nombre que está seleccionando para el nodo emisor lo está usando otro nodo de los que están insertados ya en el escenario. En este simulador cada nodo debe tener un nombre distinto, no puede haber nodos con nombres repetidos.



Esta ventana indica que en el campo donde debería introducir el nombre del nodo emisor, ha introducido sólo espacios en blanco o tabuladores. No se permiten nombres formados exclusivamente por espacios en blanco, aunque se permiten nombres que tengan espacios en blanco además de otros caracteres.



Esta ventana indica que no ha seleccionado qué nodo será el destino para el tráfico que generará el emisor.

En cualquiera de los casos anteriores, usted debe pulsar sobre el botón **"Aceptar"** de la ventana que le indica el error, con lo que volverá de nuevo a la ventana de configuración del nodo emisor y podrá corregir el error. Posteriormente continúe como si no hubiese aparecido un mensaje de error nunca.

3.3.5.4.1.2. Receptor de tráfico

Un receptor de tráfico es el nodo que consume el tráfico generado por un nodo emisor. Siempre que se simula una red, además de la topología hay que especificar quién recibe el tráfico que circulará por la misma puesto que si no, la simulación no haría nada. Para insertar un nodo receptor de tráfico hay que hacer clic sobre el correspondiente icono, señalado en la siguiente figura con un círculo rojo.

Tras lo cual aparecerá en la ventana de escenario será la pantalla de configuración del receptor, que se muestra a continuación.

La pantalla de configuración del receptor se compone de tres partes, separadas por pestañas:

- **General:** donde se especifican parámetros generales del nodo. Casi no varía entre los distintos nodos permitidos por Open SimMPLS 1.0
- **Rápida:** donde podemos hacer una configuración básica muy rápida y sencilla del receptor.
- **Avanzada:** donde supuestamente podemos refinar mucho más la configuración del receptor a costa de dedicar más tiempo y tener más conocimientos.

No es necesario utilizar las dos últimas opciones siempre; basta con utilizar una de ellas para configurar correctamente el receptor.

La pestaña **"General"**, se activa haciendo clic sobre ella; aunque siempre que abre la pantalla de configuración, es esta pestaña la que aparece seleccionada por defecto.

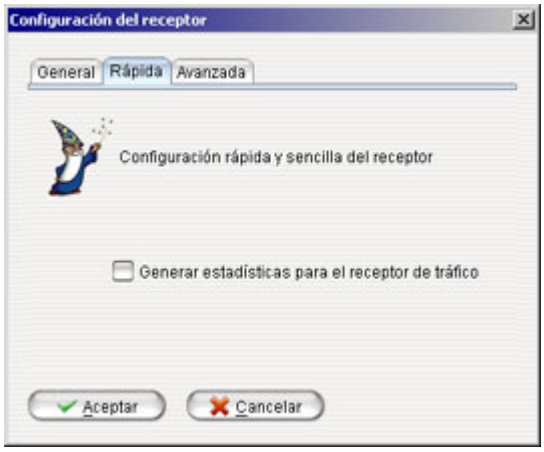
En esta pestaña, que se muestra dos figuras atrás, aparece un icono que representa al nodo receptor en el escenario. Una vez que el nodo esté configurado e insertado, podremos identificarlo por este icono. Junto al icono aparece una pequeña descripción de qué hace este nodo; en este caso, recibir tráfico de red. También aparece un campo **"Nombre del receptor"** donde se nos permite escribir. En él deberemos escribir un nombre para el

receptor, nombre que servirá para referirse a él en todo el escenario y nombre que se mostrará, si se desea, en las ventanas de diseño y simulación. Justo esto último es lo que se puede configurar en el *checkbox* titulado **"Ver nombre"**. Si lo marcamos, el nombre del nodo que hemos escrito se mostrará en las distintas pantallas de la ventana de escenario. Si lo desmarcamos, no se verá.

Además de lo anterior, aparece una zona rectangular, con una etiqueta que dice **"Selecciona la posición"**. Representa una imagen en miniatura del escenario que estamos creando y se utiliza para seleccionar el lugar de la pantalla de diseño en el que queremos colocar el receptor. Se puede seleccionar el lugar haciendo clic sobre la zona de la miniatura y tendrá repercusión inmediata sobre la pantalla real en el momento de pulsar el botón **"Aceptar"**. Un pequeño círculo rojo indica en cada momento la zona aproximada de la pantalla de diseño en la que se insertará el nodo receptor, aunque posteriormente se podrá modificar dicha posición fácilmente.

La segunda pestaña de la pantalla de configuración, la pestaña **"Rápida"**, se activa haciendo clic sobre ella.

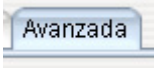
Y tras hacerlo aparecerá la ventana de configuración rápida del nodo receptor. Es fácil en todo momento saber en qué zona de la configuración nos encontramos. En la pestaña de configuración rápida, aparece la imagen de un mago, indicando que estamos en la zona donde se puede configurar todo de forma transparente, muy rápida y muy sencilla.



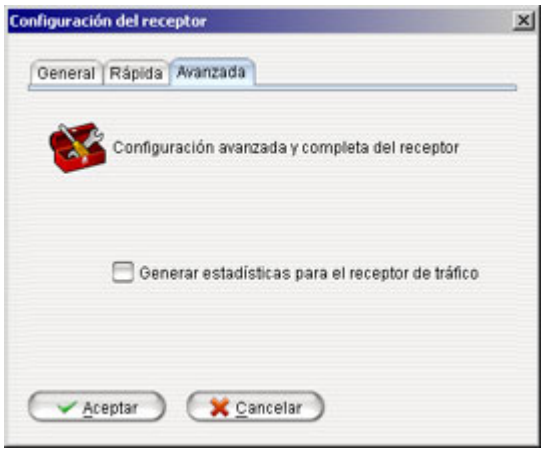
En el caso del receptor, la configuración rápida se limita a la opción llamada **"Generar estadísticas para el receptor de tráfico"** que hará, si se marca, que el simulador guarde toda la información necesaria para construir unas estadísticas completas del funcionamiento del receptor desde el inicio de la simulación hasta el final. Por defecto está deshabilitada. Si se deja sin marcar esta opción, no se podrán saber datos estadísticos del nodo receptor de ninguna de las maneras.

¡Nota! La generación de estadísticas para cualquiera de los elementos de la topología consume una ingente cantidad de recursos debido al gran volumen de información que maneja el simulador. Por defecto están desactivadas y se recomienda que deje esta opción así y la active exclusivamente en aquellos nodos que quiera estudiar en un momento determinado; si no, la fluidez de la simulación decaerá con el transcurso de la misma. Se recomienda que cuando esté interesado en la simulación visual desactive todas las estadísticas de los elementos; y cuando necesite estadísticas, no espere una simulación visual fluida en simulaciones medias-largas.

La tercera pestaña de la pantalla de configuración, la pestaña **"Avanzada"**, se activa haciendo clic sobre ella.



Y tras hacerlo aparecerá la ventana de configuración avanzada del nodo receptor. Es fácil saber en todo momento que nos encontramos en esta área porque aparece la imagen de una caja de herramientas, indicando que estamos en la zona donde se puede configurar todo mucho más detalladamente.

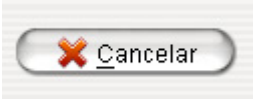


La configuración avanzada del receptor se limita a la opción llamada **"Generar estadísticas para el receptor de tráfico"** que hará, si se marca, que el simulador guarde toda la información necesaria para construir unas estadísticas completas del funcionamiento del receptor desde el inicio de la simulación hasta el final. Por defecto está deshabilitada. Si se deja sin marcar esta opción, no se podrán saber datos estadísticos del nodo receptor de ninguna de las maneras.

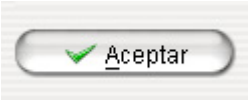
¡Nota! La generación de estadísticas para cualquiera de los elementos de la topología consume una ingente cantidad de recursos debido al gran volumen de información que maneja el simulador. Por defecto están desactivadas y se recomienda que deje esta opción así y la active exclusivamente en aquellos nodos que quiera estudiar en un momento determinado; si no, la fluidez de la simulación decaerá con el transcurso de la misma. Se recomienda que cuando esté interesado en la simulación visual desactive todas las estadísticas de los elementos; y cuando necesite estadísticas, no espere una simulación visual fluida en simulaciones medias-largas.

Como se puede observar, en el caso del receptor, las pestañas de configuración rápida y de configuración avanzada son idénticas. Esto es así porque el receptor no tiene ninguna característica que configurar. El receptor existe en el simulador únicamente porque es necesario para el funcionamiento de la red y porque importan las estadísticas que pueda generar. Sin embargo, a todos los efectos el nodo receptor es un nodo ideal: no se congestiona, no deja de funcionar, puede recibir todo el tráfico que se le envíe y de todo tipo, etcétera.

Si en cualquier momento decide dar marcha atrás y no insertar finalmente el nodo receptor en el escenario, pulse el botón **"Cancelar"**. La ventana se cerrará y volverá al escenario que seguirá como antes de intentar insertar el receptor.

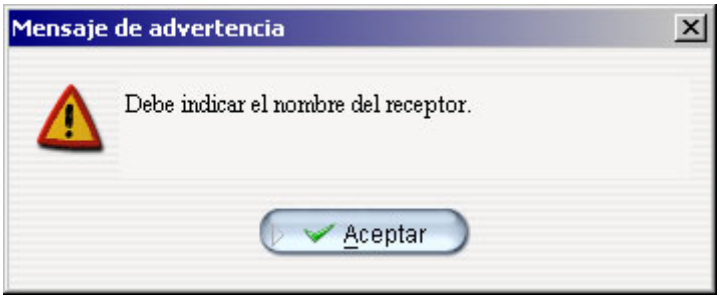


Una vez que haya terminado de configurar el nodo receptor como desee, pulse el botón **"Aceptar"** para añadir el nuevo nodo receptor al escenario.

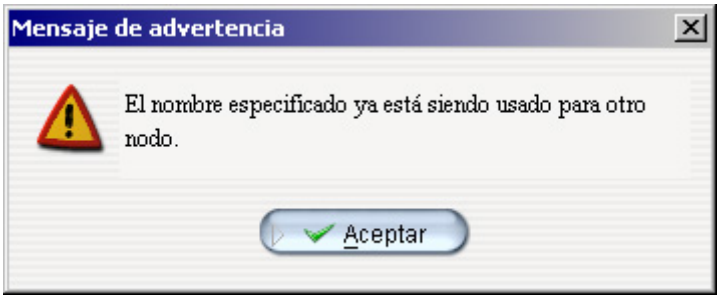


Si en este momento todo está correcto, la ventana de configuración del nodo receptor desaparecerá y volverá a la pantalla de diseño, donde ya se habrá insertado el nuevo nodo receptor correctamente.

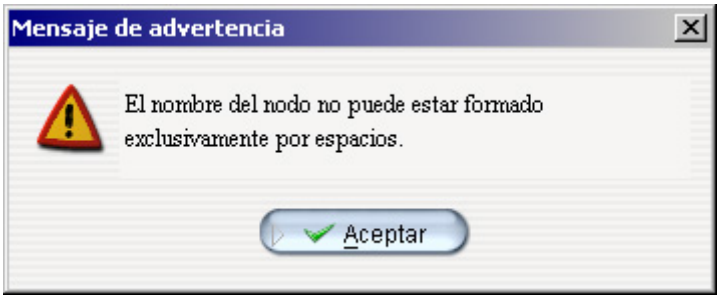
Sin embargo, puede que al pulsar el botón **"Aceptar"**, aparezca alguna ventana indicándole de un error, como vemos en las siguientes figuras.



Esta ventana indica que se le ha olvidado escribir el nombre del receptor. Un receptor siempre tiene que tener un nombre, independientemente de que se desee mostrar o no en las pantallas de diseño y simulación.



Esta ventana indica que el nombre que está seleccionando para el nodo receptor lo está usando otro nodo de los que están insertados ya en el escenario. En este simulador cada nodo debe tener un nombre distinto, no puede haber nodos con nombres repetidos.



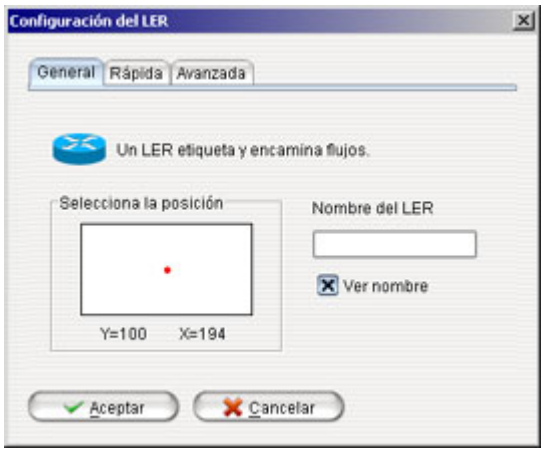
Esta ventana indica que en el campo donde debería introducir el nombre del nodo receptor, ha introducido sólo espacios en blanco o tabuladores. No se permiten nombres formados exclusivamente por espacios en blanco, aunque se permiten nombres que tengan espacios en blanco además de otros caracteres.

En cualquiera de los casos anteriores, usted debe pulsar sobre el botón **"Aceptar"** de la ventana que le indica el error, con lo que volverá de nuevo a la ventana de configuración del nodo receptor y podrá corregir el error. Posteriormente continúe como si no hubiese aparecido un mensaje de error nunca.

3.3.5.4.1.3. Label Edge Router

Un Label Edge Router (LER) el nodo encargado de etiquetar paquetes IPv4 o MPLS, clasificarlo, establecer un camino hacia el destino a través del dominio MPLS y permitir, al final, la entrada del paquete etiquetado al dominio MPLS. Para insertar un nodo LER hay que hacer clic sobre el correspondiente icono, señalado en la siguiente figura con un círculo rojo.

Tras lo cual nos aparecerá en la ventana de escenario la pantalla de configuración del LER, que se muestra a continuación.



La pantalla de configuración del LER se compone de tres partes, separadas por pestañas:

- **General:** donde se especifican parámetros generales del nodo. Casi no varía entre los distintos nodos permitidos por Open SimMPLS 1.0

- **Rápida:** donde podemos hacer una configuración básica muy rápida y sencilla del LER.
- **Avanzada:** donde podemos refinar mucho más la configuración del LER a costa de dedicar más tiempo y tener más conocimientos.

No es necesario utilizar las dos últimas opciones siempre; basta con utilizar una de ellas para configurar correctamente el LER.

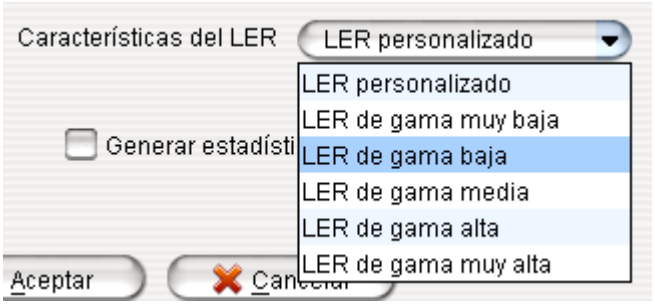
La pestaña **"General"**, se activa haciendo clic sobre ella; aunque siempre que abre la pantalla de configuración, es esta pestaña la que aparece seleccionada por defecto.

En esta pestaña, que se muestra dos figuras atrás, aparece un icono que representa al nodo LER en el escenario. Una vez que el nodo esté configurado e insertado, podremos identificarlo por este icono. Junto al icono aparece una pequeña descripción de qué hace este nodo; en este caso, etiquetar paquetes y encaminar flujos. También aparece un campo **"Nombre del LER"** donde se nos permite escribir. En él deberemos escribir un nombre para el LER, nombre que servirá para referirse al él en todo el escenario y nombre que se mostrará, si se desea, en las ventanas de diseño y simulación. Justo esto último es lo que se puede configurar en el *checkbox* titulado **"Ver nombre"**. Si lo marcamos, el nombre del nodo que hemos escrito se mostrará en las distintas pantallas de la ventana de escenario. Si lo desmarcamos, no se verá. Por último, aparece una zona rectangular, con una etiqueta que dice **"Selecciona la posición"**. Representa una imagen en miniatura del escenario que estamos creando y se utiliza para seleccionar el lugar de la pantalla de diseño en el que queremos colocar el LER. Se puede seleccionar el lugar haciendo clic sobre la zona de la miniatura y tendrá repercusión inmediata sobre la pantalla real en el momento de pulsar el botón **"Aceptar"**. Un pequeño círculo rojo indica en cada momento la zona aproximada de la pantalla de diseño en la que se insertará el nodo LER, aunque posteriormente se podrá modificar dicha posición fácilmente.

La segunda pestaña de la pantalla de configuración, la pestaña **"Rápida"**, se activa haciendo clic sobre ella.

Y tras hacerlo aparecerá la ventana de configuración rápida del nodo LER. Es fácil en todo momento saber en qué zona de la configuración nos encontramos. En la pestaña de configuración rápida, aparece la imagen de un mago, indicando que estamos en la zona donde se puede configurar todo de forma transparente, muy rápida y muy sencilla.

En el caso del LER, la configuración rápida se limita a tener que seleccionar de la lista desplegable llamada **"Características del LER"**, el tipo de LER que queremos insertar, haciendo referencia a sus características, desde un LER simple hasta un LER caro y avanzado.



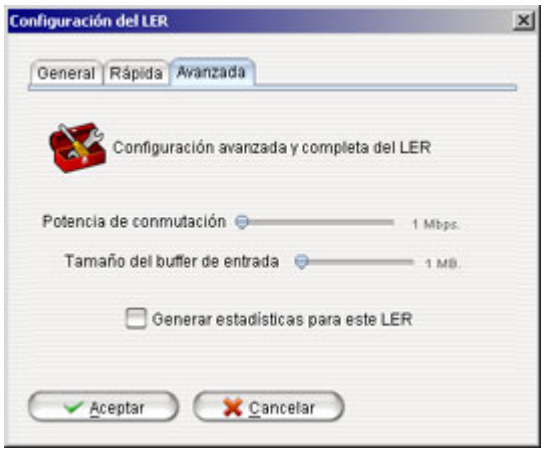
El tipo de LER **"Personalizado"** no indica un tipo de LER en si, sino que aparece aquí cuando el LER no se ha configurado en la pestaña **"Rápida"** sino en la pestaña **"Avanzadas"** y por tanto no coincide con alguno de estos tipos predefinidos.

La última opción, llamada **"Generar estadísticas para este LER"** hará, si se marca, que el simulador guarde toda la información necesaria para construir unas estadísticas completas del funcionamiento del LER desde el inicio de la simulación hasta el final. Por defecto está deshabilitada. Si se deja sin marcar esta opción, no se podrán saber datos estadísticos del nodo LER de ninguna de las maneras.

> *¡Nota! La generación de estadísticas para cualquiera de los elementos de la topología consume una ingente cantidad de recursos debido al gran volumen de información que maneja el simulador. Por defecto están desactivadas y se recomienda que deje esta opción así y la active exclusivamente en aquellos nodos que quiera estudiar en un momento determinado; si no, la fluidez de la simulación decaerá con el transcurso de la misma. Se recomienda que cuando esté interesado en la simulación visual desactive todas las estadísticas de los elementos; y cuando necesite estadísticas, no espere una simulación visual fluida en simulaciones medias-largas.*

La tercera pestaña de la pantalla de configuración, la pestaña **"Avanzada"**, se activa haciendo clic sobre ella.

Y tras hacerlo aparecerá la ventana de configuración avanzada del nodo LER. Es fácil saber en todo momento que nos encontramos en esta área porque aparece la imagen de una caja de herramientas, indicando que estamos en la zona donde se puede configurar todo mucho más detalladamente.



Lo primero que nos encontramos es un deslizador marcado como **"Potencia de conmutación"**, donde inicialmente se puede observar que está seleccionada la cantidad de 1 Mbps. Esta propiedad se refiere al número de bits que el LER es capaz de conmutar en un segundo y determinará la cantidad de tráfico, y por tanto paquetes, que el LER puede reenviar al nodo siguiente en el camino durante la simulación, la congestión que es capaz de causar a dicho nodo, etcétera. Su unidad de medida es "Megabits por segundo (Mbps)" y los valores permitidos van desde 1 a 10240 Mbps, o lo que es lo mismo, desde 1 Mbps a 10 Gbps (Gigabits por segundo), pues el escenario a simular se supone que será una troncal de datos y los volúmenes de información serán grandes. La forma de variar el valor y seleccionar el que deseamos es moviendo el deslizador de izquierda a derecha o bien colocando el foco sobre el deslizador y usar las teclas de izquierda y derecha del teclado.

Lo siguiente que nos encontramos es un deslizador marcado como **"Tamaño del buffer de entrada"**, donde inicialmente se puede observar que está seleccionada la cantidad de 1 MB. Esta propiedad se refiere al número octetos que el nodo puede almacenar temporalmente mientras procesa otros paquetes, antes de comenzar a descartar paquetes. Su unidad de medida es "Megabyte (MB)" y los valores permitidos van desde 1 a 1024 MB, o lo que es lo mismo, desde 1 MB a 1 GB (Gigabyte). La forma de variar el valor y seleccionar el que deseamos es moviendo el deslizador de izquierda a derecha o bien colocando el foco sobre el deslizador y usar las teclas de izquierda y derecha del teclado.

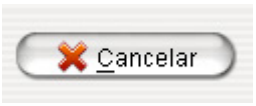
La última opción, llamada **"Generar estadísticas para este LER"** hará, si se marca, que el simulador guarde toda la información necesaria para construir unas estadísticas completas del funcionamiento del LER desde el inicio de la simulación hasta el final. Por defecto está deshabilitada. Si se deja sin marcar esta opción, no se podrán saber datos estadísticos del nodo LER de ninguna de las maneras.

> *¡Nota! La generación de estadísticas para cualquiera de los elementos de la topología consume una ingente cantidad de recursos debido al gran volumen de información que maneja el simulador. Por defecto están desactivadas y se recomienda que deje esta opción así y la active exclusivamente en aquellos nodos que quiera estudiar en un momento determinado; si no, la fluidez de la simulación decaerá con el transcurso de la misma. Se recomienda que cuando esté interesado en la simulación visual desactive todas las estadísticas de los*

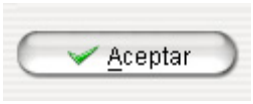
elementos; y cuando necesite estadísticas, no espere una simulación visual fluida en simulaciones medias-largas.

Cualquier modificación que se haga en la pestaña de configuración **"Avanzada"** provocará que en la ventana **"Rápida"** aparezca el tipo de LER como **"Personalizado"**, pero no es una cosa de la que deba preocuparse.

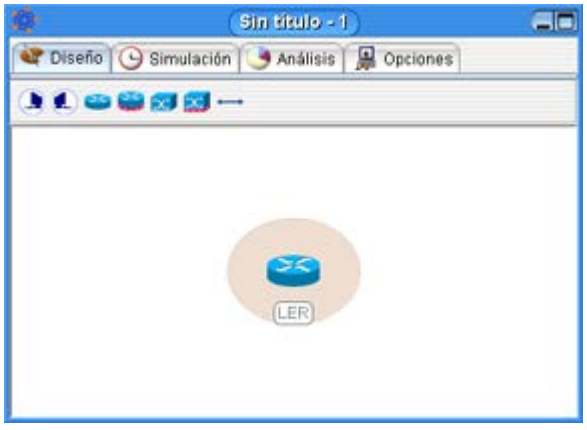
Si en cualquier momento decide dar marcha atrás y no insertar finalmente el nodo LER en el escenario, pulse el botón **"Cancelar"**. La ventana se cerrará y volverá al escenario que seguirá como antes de intentar insertar el LER.



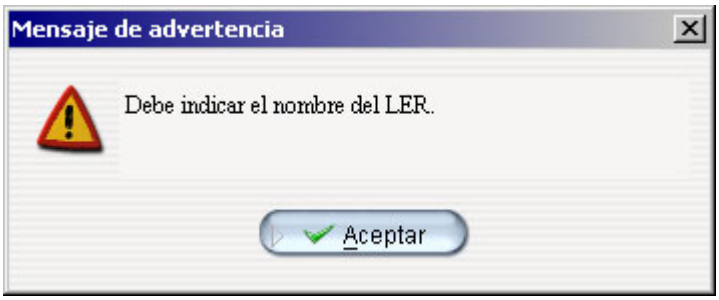
Una vez que haya terminado de configurar el nodo LER como desee, pulse el botón **"Aceptar"** para añadir el nuevo nodo LER al escenario.



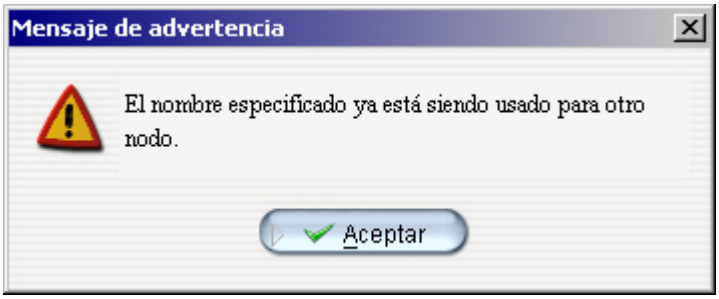
Si en este momento todo está correcto, la ventana de configuración del nodo LER desaparecerá y volverá a la pantalla de diseño, donde ya se habrá insertado el nuevo nodo LER correctamente.



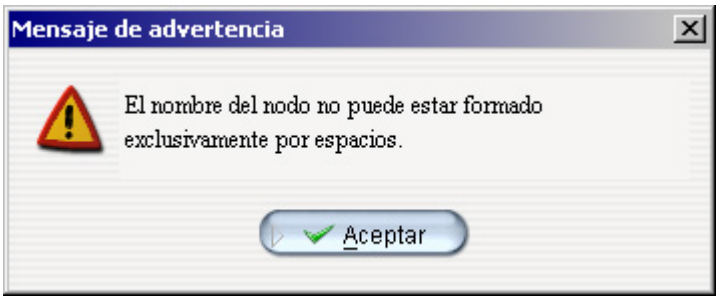
Sin embargo, puede que al pulsar el botón **"Aceptar"**, aparezca alguna ventana indicándole de un error, como vemos en las siguientes figuras.



Esta ventana indica que se le ha olvidado escribir el nombre del LER. Un LER siempre tiene que tener un nombre, independientemente de que se desee mostrar o no en las pantallas de diseño y simulación.



Esta ventana indica que el nombre que está seleccionando para el nodo LER lo está usando otro nodo de los que están insertados ya en el escenario. En este simulador cada nodo debe tener un nombre distinto, no puede haber nodos con nombres repetidos.



Esta ventana indica que en el campo donde debería introducir el nombre del nodo LER, ha introducido sólo espacios en blanco o tabuladores. No se permiten nombres formados

exclusivamente por espacios en blanco, aunque se permiten nombres que tengan espacios en blanco además de otros caracteres.

En cualquiera de los casos anteriores, usted debe pulsar sobre el botón **"Aceptar"** de la ventana que le indica el error, con lo que volverá de nuevo a la ventana de configuración del nodo LER y podrá corregir el error. Posteriormente continúe como si no hubiese aparecido un mensaje de error nunca.

Una vez que el nodo LER esté insertado, verá que siempre hay una zona sombreada de anaranjado que rodea al LER (si sólo hay uno en la topología) o que une a todos los LER existentes entre si (si hay más de uno). Este es el dominio MPLS. Se representa así automáticamente para que se sepa que siempre para entrar o salir del dominio MPLS se debe hacer a través de un LER/LERA.

3.3.5.4.1.4. Label Edge Router activo

Un Label Edge Router Activo (LERA) es el nodo encargado de etiquetar paquetes IPv4 o MPLS, clasificarlo, establecer un camino hacia el destino a través del dominio MPLS y permitir, al final, la entrada del paquete etiquetado al dominio MPLS. Además es el encargado de analizar la cabecera IPv4 para saber si los paquetes tienen requerimientos de garantía de servicio (GoS) y si es así, codificar esos requisitos en la cabecera MPLS. Un tráfico IPv4 marcado con GoS sólo puede conservar esos atributos de GoS dentro del dominio MPLS si accede a él a través de un nodo LERA. Para insertar un nodo LERA hay que hacer clic sobre el correspondiente icono, señalado en la siguiente figura con un círculo rojo.

Tras lo cual nos aparecerá en la ventana de escenario la pantalla de configuración del LERA, que se muestra a continuación.

La pantalla de configuración del LERA se compone de tres partes, separadas por pestañas:

- **General:** donde se especifican parámetros generales del nodo. Casi no varía entre los distintos nodos permitidos por Open SimMPLS 1.0
- **Rápida:** donde podemos hacer una configuración básica muy rápida y sencilla del LERA.
- **Avanzada:** donde podemos refinar mucho más la configuración del LERA a costa de dedicar más tiempo y tener más conocimientos.

No es necesario utilizar las dos últimas opciones siempre; basta con utilizar una de ellas para configurar correctamente el LERA.

La pestaña **"General"**, se activa haciendo clic sobre ella; aunque siempre que abre la pantalla de configuración, es esta pestaña la que aparece seleccionada por defecto.

En esta pestaña, que se muestra dos figuras atrás, aparece un icono que representa al nodo LERA en el escenario. Una vez que el nodo esté configurado e insertado, podremos identificarlo por este icono. Junto al icono aparece una pequeña descripción de qué hace este nodo; en este caso, etiquetar paquetes y encaminar flujos. También aparece un campo **"Nombre del LERA"** donde se nos permite escribir. En él deberemos escribir un nombre para el LERA, nombre que servirá para referirse al él en todo el escenario y nombre que se

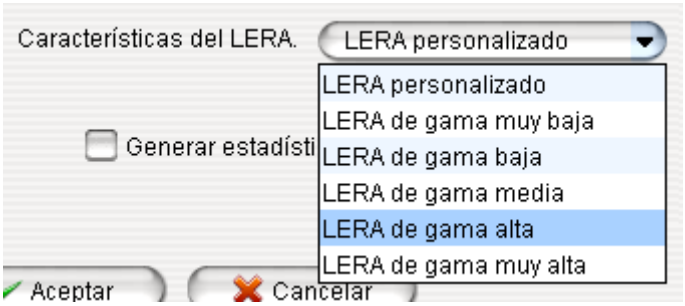
mostrará, si se desea, en las ventanas de diseño y simulación. Justo esto último es lo que se puede configurar en el *checkbox* titulado **"Ver nombre"**. Si lo marcamos, el nombre del nodo que hemos escrito se mostrará en las distintas pantallas de la ventana de escenario. Si lo desmarcamos, no se verá. Por último, aparece una zona rectangular, con una etiqueta que dice **"Selecciona la posición"**. Representa una imagen en miniatura del escenario que estamos creando y se utiliza para seleccionar el lugar de la pantalla de diseño en el que queremos colocar el LERA. Se puede seleccionar el lugar haciendo clic sobre la zona de la miniatura y tendrá repercusión inmediata sobre la pantalla real en el momento de pulsar el botón **"Aceptar"**. Un pequeño círculo rojo indica en cada momento la zona aproximada de la pantalla de diseño en la que se insertará el nodo LERA, aunque posteriormente se podrá modificar dicha posición fácilmente.

La segunda pestaña de la pantalla de configuración, la pestaña **"Rápida"**, se activa haciendo clic sobre ella.

Y tras hacerlo aparecerá la ventana de configuración rápida del nodo LERA. Es fácil en todo momento saber en qué zona de la configuración nos encontramos. En la pestaña de configuración rápida, aparece la imagen de un mago, indicando que estamos en la zona donde se puede configurar todo de forma transparente, muy rápida y muy sencilla.

En el caso del LERA, la configuración rápida se limita a tener que seleccionar de la lista desplegable llamada **"Características del LERA"**, el tipo de LERA que queremos

insertar, haciendo referencia a sus características, desde un LERA simple hasta un LERA caro y avanzado.



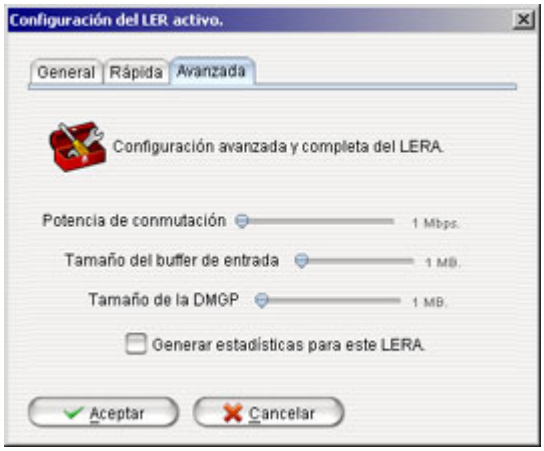
El tipo de LERA **"Personalizado"** no indica un tipo de LERA en si, sino que aparece aquí cuando el LERA no se ha configurado en la pestaña **"Rápida"** sino en la pestaña **"Avanzadas"** y por tanto no coincide con alguno de estos tipos predefinidos.

La última opción, llamada **"Generar estadísticas para este LERA"** hará, si se marca, que el simulador guarde toda la información necesaria para construir unas estadísticas completas del funcionamiento del LERA desde el inicio de la simulación hasta el final. Por defecto está deshabilitada. Si se deja sin marcar esta opción, no se podrán saber datos estadísticos del nodo LERA de ninguna de las maneras.

¡Nota! La generación de estadísticas para cualquiera de los elementos de la topología consume una ingente cantidad de recursos debido al gran volumen de información que maneja el simulador. Por defecto están desactivadas y se recomienda que deje esta opción así y la active exclusivamente en aquellos nodos que quiera estudiar en un momento determinado; si no, la fluidez de la simulación decaerá con el transcurso de la misma. Se recomienda que cuando esté interesado en la simulación visual desactive todas las estadísticas de los elementos; y cuando necesite estadísticas, no espere una simulación visual fluida en simulaciones medias-largas.

La tercera pestaña de la pantalla de configuración, la pestaña **"Avanzada"**, se activa haciendo clic sobre ella.

Y tras hacerlo aparecerá la ventana de configuración avanzada del nodo LERA. Es fácil saber en todo momento que nos encontramos en esta área porque aparece la imagen de una caja de herramientas, indicando que estamos en la zona donde se puede configurar todo mucho más detalladamente.



Lo primero que nos encontramos es un deslizador marcado como **"Potencia de conmutación"**, donde inicialmente se puede observar que está seleccionada la cantidad de 1 Mbps. Esta propiedad se refiere al número de bits que el LERA es capaz de conmutar en un segundo y determinará la cantidad de tráfico, y por tanto paquetes, que el LERA puede reenviar al nodo siguiente en el camino durante la simulación, la congestión que es capaz de causar a dicho nodo, etcétera. Su unidad de medida es "Megabits por segundo (Mbps)" y los valores permitidos van desde 1 a 10240 Mbps, o lo que es lo mismo, desde 1 Mbps a 10 Gbps (Gigabits por segundo), pues el escenario a simular se supone que será una troncal de datos y los volúmenes de información serán grandes. La forma de variar el valor y seleccionar el que deseamos es moviendo el deslizador de izquierda a derecha o bien colocando el foco sobre el deslizador y usar las teclas de izquierda y derecha del teclado.

Lo siguiente que nos encontramos es un deslizador marcado como **"Tamaño del buffer de entrada"**, donde inicialmente se puede observar que está seleccionada la cantidad de 1 MB. Esta propiedad se refiere al número octetos que el nodo puede almacenar temporalmente mientras procesa otros paquetes, antes de comenzar a descartar paquetes. Su unidad de medida es "Megabyte (MB)" y los valores permitidos van desde 1 a 1024 MB, o lo que es lo mismo, desde 1 MB a 1 GB (Gigabyte). La forma de variar el valor y

seleccionar el que deseamos es moviendo el deslizador de izquierda a derecha o bien colocando el foco sobre el deslizador y usar las teclas de izquierda y derecha del teclado.

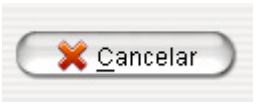
Posteriormente se encuentra un deslizador marcado como **"Tamaño de la DMGP"**, donde inicialmente se puede observar que está seleccionada la cantidad de 1KB. Esta propiedad se refiere al número octetos que el LERA puede almacenar temporalmente para su posible retransmisión local; es decir, a mayor DMGP, mayor probabilidad de que este paquete pueda servir una petición de retransmisión de un paquete descargado en otro nodo de la red. Su unidad de medida es "Kilobyte (KB)" y los valores permitidos van desde 1 a 102400 KB (100 MB). La forma de variar el valor y seleccionar el que deseamos es moviendo el deslizador de izquierda a derecha o bien colocando el foco sobre el deslizador y usar las teclas de izquierda y derecha del teclado.

La última opción, llamada **"Generar estadísticas para este LERA"** hará, si se marca, que el simulador guarde toda la información necesaria para construir unas estadísticas completas del funcionamiento del LERA desde el inicio de la simulación hasta el final. Por defecto está deshabilitada. Si se deja sin marcar esta opción, no se podrán saber datos estadísticos del nodo LERA de ninguna de las maneras.

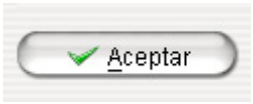
> *¡Nota! La generación de estadísticas para cualquiera de los elementos de la topología consume una ingente cantidad de recursos debido al gran volumen de información que maneja el simulador. Por defecto están desactivadas y se recomienda que deje esta opción así y la active exclusivamente en aquellos nodos que quiera estudiar en un momento determinado; si no, la fluidez de la simulación decaerá con el transcurso de la misma. Se recomienda que cuando esté interesado en la simulación visual desactive todas las estadísticas de los elementos; y cuando necesite estadísticas, no espere una simulación visual fluida en simulaciones medias-largas.*

Cualquier modificación que se haga en la pestaña de configuración **"Avanzada"** provocará que en la ventana **"Rápida"** aparezca el tipo de LERA como **"Personalizado"**, pero no es una cosa de la que deba preocuparse.

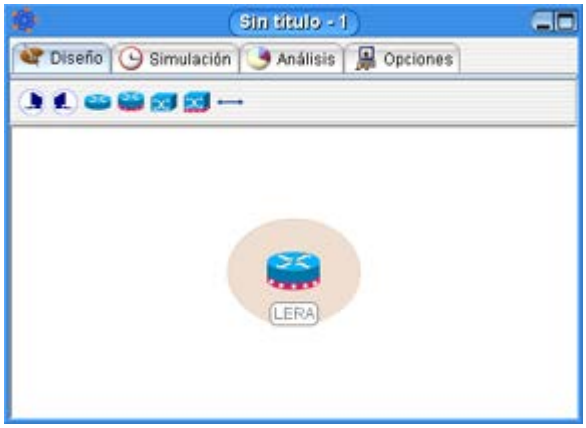
Si en cualquier momento decide dar marcha atrás y no insertar finalmente el nodo LERA en el escenario, pulse el botón **"Cancelar"**. La ventana se cerrará y volverá al escenario que seguirá como antes de intentar insertar el LERA.



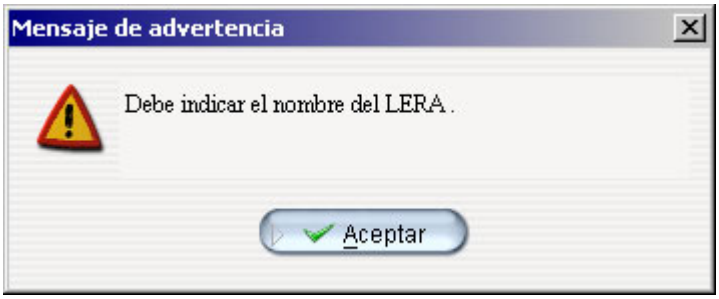
Una vez que haya terminado de configurar el nodo LERA como desee, pulse el botón **"Aceptar"** para añadir el nuevo nodo LERA al escenario.



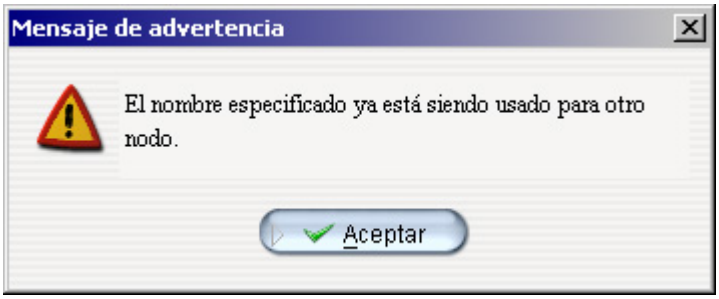
Si en este momento todo está correcto, la ventana de configuración del nodo LERA desaparecerá y volverá a la pantalla de diseño, donde ya se habrá insertado el nuevo nodo LERA correctamente.



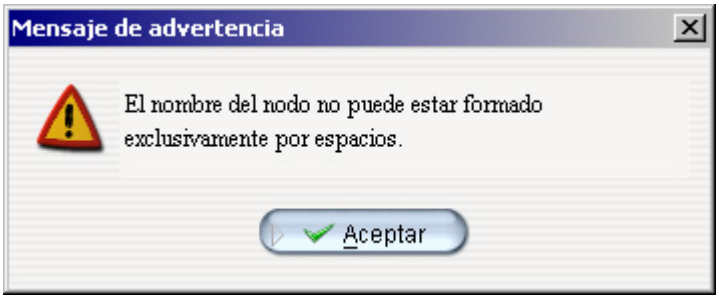
Sin embargo, puede que al pulsar el botón **"Aceptar"**, aparezca alguna ventana indicándole de un error, como vemos en las siguientes figuras.



Esta ventana indica que se le ha olvidado escribir el nombre del LERA. Un LERA siempre tiene que tener un nombre, independientemente de que se desee mostrar o no en las pantallas de diseño y simulación.



Esta ventana indica que el nombre que está seleccionando para el nodo LERA lo está usando otro nodo de los que están insertados ya en el escenario. En este simulador cada nodo debe tener un nombre distinto, no puede haber nodos con nombres repetidos.



Esta ventana indica que en el campo donde debería introducir el nombre del nodo LERA, ha introducido sólo espacios en blanco o tabuladores. No se permiten nombres formados exclusivamente por espacios en blanco, aunque se permiten nombres que tengan espacios en blanco además de otros caracteres.

En cualquiera de los casos anteriores, usted debe pulsar sobre el botón **"Aceptar"** de la ventana que le indica el error, con lo que volverá de nuevo a la ventana de configuración del nodo LERA y podrá corregir el error. Posteriormente continúe como si no hubiese aparecido un mensaje de error nunca.

Una vez que el nodo LERA esté insertado, verá que siempre hay una zona sombreada de anaranjado que rodea al LERA (si sólo hay uno en la topología) o que une a todos los LERA existentes entre si (si hay más de uno). Este es el dominio MPLS. Se representa así automáticamente para que se sepa que siempre para entrar o salir del dominio MPLS se debe hacer a través de un LER/LERA.

3.3.5.4.1.5. Label Switch Router

Un Label Switch Router (LSR) es el nodo encargado conmutar tráfico MPLS en el interior del dominio. Es rápido pues sólo observa la etiqueta puesta sobre el paquete por el LER/LERA de entrada al dominio MPLS. Un nodo LSR jamás puede hacer de nodo de entrada al dominio MPLS pues no tiene capacidad para ello. Para insertar un nodo LSR hay que hacer clic sobre el correspondiente icono, señalado en la siguiente figura con un círculo rojo.

Tras lo cual nos aparecerá en la ventana de escenario la pantalla de configuración del LSR, que se muestra a continuación.

La pantalla de configuración del LSR se compone de tres partes, separadas por pestañas:

- **General:** donde se especifican parámetros generales del nodo. Casi no varía entre los distintos nodos permitidos por Open SimMPLS 1.0
- **Rápida:** donde podemos hacer una configuración básica muy rápida y sencilla del LSR.
- **Avanzada:** donde podemos refinar mucho más la configuración del LSR a costa de dedicar más tiempo y tener más conocimientos.

No es necesario utilizar las dos últimas opciones siempre; basta con utilizar una de ellas para configurar correctamente el LSR.

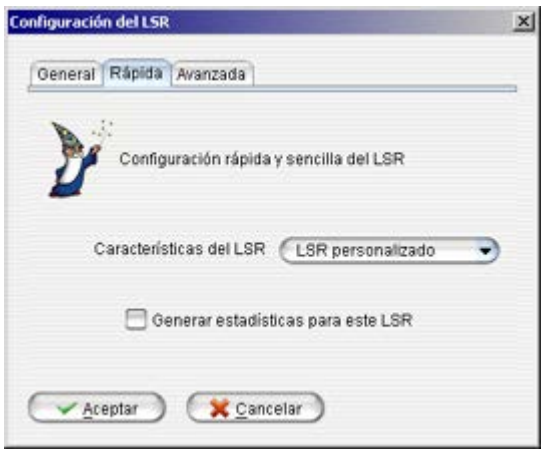
La pestaña **"General"**, se activa haciendo clic sobre ella; aunque siempre que abre la pantalla de configuración, es esta pestaña la que aparece seleccionada por defecto.

En esta pestaña, que se muestra dos figuras atrás, aparece un icono que representa al nodo LSR en el escenario. Una vez que el nodo esté configurado e insertado, podremos identificarlo por este icono. Junto al icono aparece una pequeña descripción de qué hace este nodo; en este caso, etiquetar paquetes y encaminar flujos. También aparece un campo **"Nombre del LSR"** donde se nos permite escribir. En él deberemos escribir un nombre para el LSR, nombre que servirá para referirse al él en todo el escenario y nombre que se

mostrará, si se desea, en las ventanas de diseño y simulación. Justo esto último es lo que se puede configurar en el *checkbox* titulado **"Ver nombre"**. Si lo marcamos, el nombre del nodo que hemos escrito se mostrará en las distintas pantallas de la ventana de escenario. Si lo desmarcamos, no se verá. Por último, aparece una zona rectangular, con una etiqueta que dice **"Selecciona la posición"**. Representa una imagen en miniatura del escenario que estamos creando y se utiliza para seleccionar el lugar de la pantalla de diseño en el que queremos colocar el LSR. Se puede seleccionar el lugar haciendo clic sobre la zona de la miniatura y tendrá repercusión inmediata sobre la pantalla real en el momento de pulsar el botón **"Aceptar"**. Un pequeño círculo rojo indica en cada momento la zona aproximada de la pantalla de diseño en la que se insertará el nodo LSR, aunque posteriormente se podrá modificar dicha posición fácilmente.

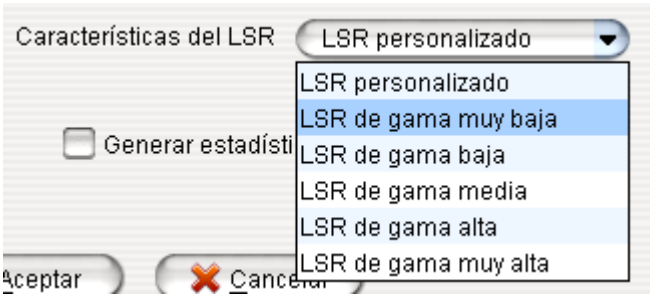
La segunda pestaña de la pantalla de configuración, la pestaña **"Rápida"**, se activa haciendo clic sobre ella.

Y tras hacerlo aparecerá la ventana de configuración rápida del nodo LSR. Es fácil en todo momento saber en qué zona de la configuración nos encontramos. En la pestaña de configuración rápida, aparece la imagen de un mago, indicando que estamos en la zona donde se puede configurar todo de forma transparente, muy rápida y muy sencilla.



En el caso del LSR, la configuración rápida se limita a tener que seleccionar de la lista desplegable llamada **"Características del LSR"**, el tipo de LSR que queremos insertar,

haciendo referencia a sus características, desde un LSR simple hasta un LSR caro y avanzado.

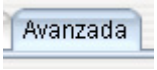


El tipo de LSR **"Personalizado"** no indica un tipo de LSR en si, sino que aparece aquí cuando el LSR no se ha configurado en la pestaña **"Rápida"** sino en la pestaña **"Avanzadas"** y por tanto no coincide con alguno de estos tipos predefinidos.

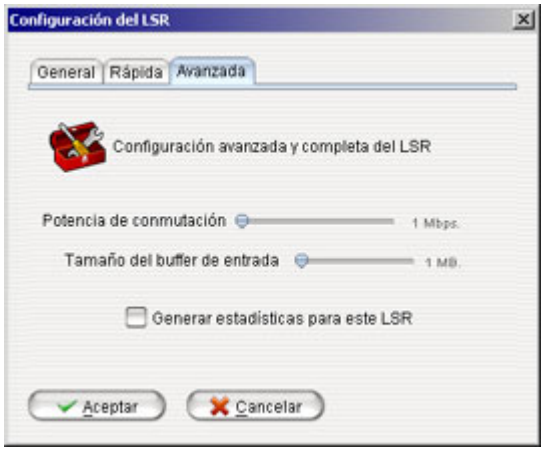
La última opción, llamada **"Generar estadísticas para este LSR"** hará, si se marca, que el simulador guarde toda la información necesaria para construir unas estadísticas completas del funcionamiento del LSR desde el inicio de la simulación hasta el final. Por defecto está deshabilitada. Si se deja sin marcar esta opción, no se podrán saber datos estadísticos del nodo LSR de ninguna de las maneras.

¡Nota! La generación de estadísticas para cualquiera de los elementos de la topología consume una ingente cantidad de recursos debido al gran volumen de información que maneja el simulador. Por defecto están desactivadas y se recomienda que deje esta opción así y la active exclusivamente en aquellos nodos que quiera estudiar en un momento determinado; si no, la fluidez de la simulación decaerá con el transcurso de la misma. Se recomienda que cuando esté interesado en la simulación visual desactive todas las estadísticas de los elementos; y cuando necesite estadísticas, no espere una simulación visual fluida en simulaciones medias-largas.

La tercera pestaña de la pantalla de configuración, la pestaña **"Avanzada"**, se activa haciendo clic sobre ella.



Y tras hacerlo aparecerá la ventana de configuración avanzada del nodo LSR. Es fácil saber en todo momento que nos encontramos en esta área porque aparece la imagen de una caja de herramientas, indicando que estamos en la zona donde se puede configurar todo mucho más detalladamente.



Lo primero que nos encontramos es un deslizador marcado como **"Potencia de conmutación"**, donde inicialmente se puede observar que está seleccionada la cantidad de 1 Mbps. Esta propiedad se refiere al número de bits que el LSR es capaz de conmutar en un segundo y determinará la cantidad de tráfico, y por tanto paquetes, que el LSR puede reenviar al nodo siguiente en el camino durante la simulación, la congestión que es capaz de causar a dicho nodo, etcétera. Su unidad de medida es "Megabits por segundo (Mbps)" y los valores permitidos van desde 1 a 10240 Mbps, o lo que es lo mismo, desde 1 Mbps a 10 Gbps (Gigabits por segundo), pues el escenario a simular se supone que será una troncal de datos y los volúmenes de información serán grandes. La forma de variar el valor y seleccionar el que deseamos es moviendo el deslizador de izquierda a derecha o bien colocando el foco sobre el deslizador y usar las teclas de izquierda y derecha del teclado.

Lo siguiente que nos encontramos es un deslizador marcado como **"Tamaño del buffer de entrada"**, donde inicialmente se puede observar que está seleccionada la cantidad de 1 MB. Esta propiedad se refiere al número octetos que el nodo puede almacenar temporalmente mientras procesa otros paquetes, antes de comenzar a descartar paquetes. Su unidad de medida es "Megabyte (MB)" y los valores permitidos van desde 1 a 1024 MB, o lo que es lo mismo, desde 1 MB a 1 GB (Gigabyte). La forma de variar el valor y

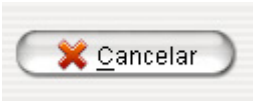
seleccionar el que deseamos es moviendo el deslizador de izquierda a derecha o bien colocando el foco sobre el deslizador y usar las teclas de izquierda y derecha del teclado.

La última opción, llamada **"Generar estadísticas para este LSR"** hará, si se marca, que el simulador guarde toda la información necesaria para construir unas estadísticas completas del funcionamiento del LSR desde el inicio de la simulación hasta el final. Por defecto está deshabilitada. Si se deja sin marcar esta opción, no se podrán saber datos estadísticos del nodo LSR de ninguna de las maneras.

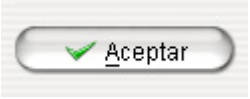
¡Nota! La generación de estadísticas para cualquiera de los elementos de la topología consume una ingente cantidad de recursos debido al gran volumen de información que maneja el simulador. Por defecto están desactivadas y se recomienda que deje esta opción así y la active exclusivamente en aquellos nodos que quiera estudiar en un momento determinado; si no, la fluidez de la simulación decaerá con el transcurso de la misma. Se recomienda que cuando esté interesado en la simulación visual desactive todas las estadísticas de los elementos; y cuando necesite estadísticas, no espere una simulación visual fluida en simulaciones medias-largas.

Cualquier modificación que se haga en la pestaña de configuración **"Avanzada"** provocará que en la ventana **"Rápida"** aparezca el tipo de LSR como **"Personalizado"**, pero no es una cosa de la que deba preocuparse.

Si en cualquier momento decide dar marcha atrás y no insertar finalmente el nodo LSR en el escenario, pulse el botón **"Cancelar"**. La ventana se cerrará y volverá al escenario que seguirá como antes de intentar insertar el LSR.

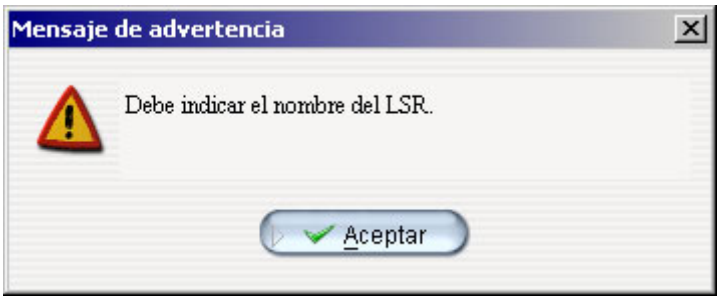


Una vez que haya terminado de configurar el nodo LSR como desee, pulse el botón **"Aceptar"** para añadir el nuevo nodo LSR al escenario.

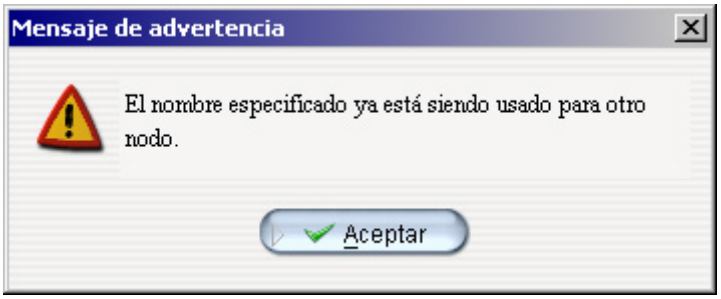


Si en este momento todo está correcto, la ventana de configuración del nodo LSR desaparecerá y volverá a la pantalla de diseño, donde ya se habrá insertado el nuevo nodo LSR correctamente.

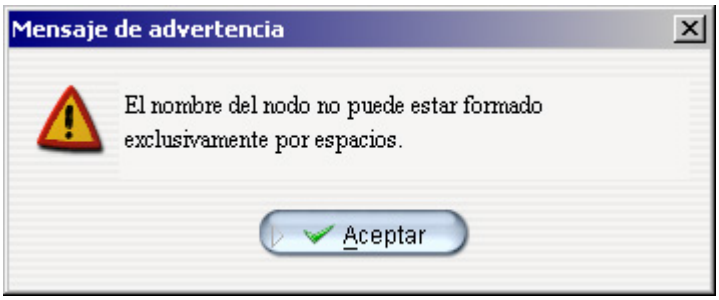
Sin embargo, puede que al pulsar el botón **"Aceptar"**, aparezca alguna ventana indicándole de un error, como vemos en las siguientes figuras.



Esta ventana indica que se le ha olvidado escribir el nombre del LSR. Un LSR siempre tiene que tener un nombre, independientemente de que se desee mostrar o no en las pantallas de diseño y simulación.



Esta ventana indica que el nombre que está seleccionando para el nodo LSR lo está usando otro nodo de los que están insertados ya en el escenario. En este simulador cada nodo debe tener un nombre distinto, no puede haber nodos con nombres repetidos.



Esta ventana indica que en el campo donde debería introducir el nombre del nodo LSR, ha introducido sólo espacios en blanco o tabuladores. No se permiten nombres formados exclusivamente por espacios en blanco, aunque se permiten nombres que tengan espacios en blanco además de otros caracteres.

En cualquiera de los casos anteriores, usted debe pulsar sobre el botón **"Aceptar"** de la ventana que le indica el error, con lo que volverá de nuevo a la ventana de configuración del nodo LSR y podrá corregir el error. Posteriormente continúe como si no hubiese aparecido un mensaje de error nunca.

3.3.5.4.1.6. Label Switch Router activo

Un Label Switch Router Activo (LSRA) es el nodo encargado conmutar tráfico MPLS en el interior del dominio. Es rápido pues sólo observa la etiqueta puesta sobre el paquete por el LER/LERA de entrada al dominio MPLS. Un nodo LSRA jamás puede hacer de nodo de entrada al dominio MPLS pues no tiene capacidad para ello. Además, un nodo LSRA tiene capacidad de recuperación de paquetes activos y de reestructuración de caminos en un entorno local y capacidad de almacenar paquetes activos temporalmente para su posible retransmisión. Para insertar un nodo LSRA hay que hacer clic sobre el correspondiente icono, señalado en la siguiente figura con un círculo rojo.

Tras lo cual nos aparecerá en la ventana de escenario la pantalla de configuración del LSRA, que se muestra a continuación.

La pantalla de configuración del LSRA se compone de tres partes, separadas por pestañas:

- **General:** donde se especifican parámetros generales del nodo. Casi no varía entre los distintos nodos permitidos por Open SimMPLS 1.0
- **Rápida:** donde podemos hacer una configuración básica muy rápida y sencilla del LSRA.
- **Avanzada:** donde podemos refinar mucho más la configuración del LSRA a costa de dedicar más tiempo y tener más conocimientos.

No es necesario utilizar las dos últimas opciones siempre; basta con utilizar una de ellas para configurar correctamente el LSRA.

La pestaña **"General"**, se activa haciendo clic sobre ella; aunque siempre que abre la pantalla de configuración, es esta pestaña la que aparece seleccionada por defecto.

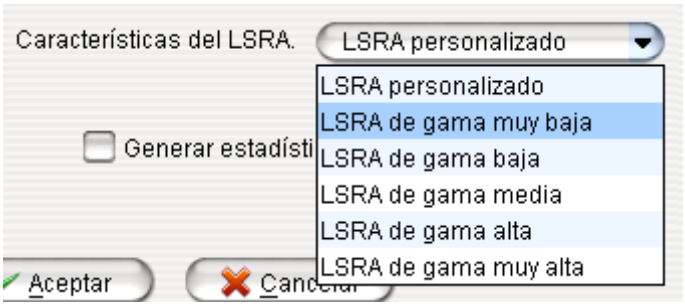
En esta pestaña, que se muestra dos figuras atrás, aparece un icono que representa al nodo LSRA en el escenario. Una vez que el nodo esté configurado e insertado, podremos

identificarlo por este icono. Junto al icono aparece una pequeña descripción de qué hace este nodo; en este caso, etiquetar paquetes y encaminar flujos. También aparece un campo **"Nombre del LSRA"** donde se nos permite escribir. En él deberemos escribir un nombre para el LSRA, nombre que servirá para referirse al él en todo el escenario y nombre que se mostrará, si se desea, en las ventanas de diseño y simulación. Justo esto último es lo que se puede configurar en el *checkbox* titulado **"Ver nombre"**. Si lo marcamos, el nombre del nodo que hemos escrito se mostrará en las distintas pantallas de la ventana de escenario. Si lo desmarcamos, no se verá. Por último, aparece una zona rectangular, con una etiqueta que dice **"Selecciona la posición"**. Representa una imagen en miniatura del escenario que estamos creando y se utiliza para seleccionar el lugar de la pantalla de diseño en el que queremos colocar el LSRA. Se puede seleccionar el lugar haciendo clic sobre la zona de la miniatura y tendrá repercusión inmediata sobre la pantalla real en el momento de pulsar el botón **"Aceptar"**. Un pequeño círculo rojo indica en cada momento la zona aproximada de la pantalla de diseño en la que se insertará el nodo LSRA, aunque posteriormente se podrá modificar dicha posición fácilmente.

La segunda pestaña de la pantalla de configuración, la pestaña **"Rápida"**, se activa haciendo clic sobre ella.

Y tras hacerlo aparecerá la ventana de configuración rápida del nodo LSRA. Es fácil en todo momento saber en qué zona de la configuración nos encontramos. En la pestaña de configuración rápida, aparece la imagen de un mago, indicando que estamos en la zona donde se puede configurar todo de forma transparente, muy rápida y muy sencilla.

En el caso del LSRA, la configuración rápida se limita a tener que seleccionar de la lista desplegable llamada **"Características del LSRA"**, el tipo de LSRA que queremos insertar, haciendo referencia a sus características, desde un LSRA simple hasta un LSRA caro y avanzado.



El tipo de LSRA **"Personalizado"** no indica un tipo de LSRA en si, sino que aparece aquí cuando el LSRA no se ha configurado en la pestaña **"Rápida"** sino en la pestaña **"Avanzadas"** y por tanto no coincide con alguno de estos tipos predefinidos.

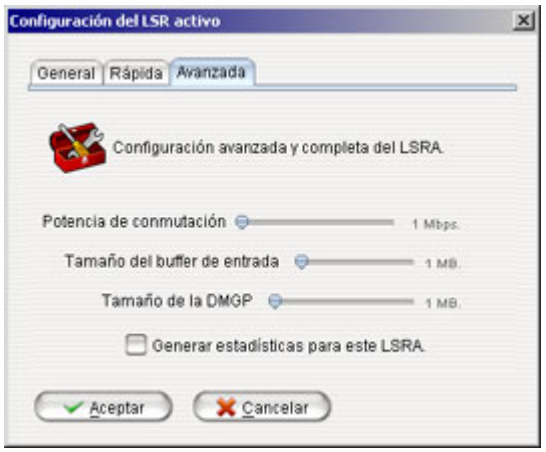
La última opción, llamada **"Generar estadísticas para este LSRA"** hará, si se marca, que el simulador guarde toda la información necesaria para construir unas estadísticas completas del funcionamiento del LSRA desde el inicio de la simulación hasta el final. Por defecto está deshabilitada. Si se deja sin marcar esta opción, no se podrán saber datos estadísticos del nodo LSRA de ninguna de las maneras.

¡Nota! La generación de estadísticas para cualquiera de los elementos de la topología consume una ingente cantidad de recursos debido al gran volumen de información que maneja el simulador. Por defecto están desactivadas y se recomienda que deje esta opción así y la

active exclusivamente en aquellos nodos que quiera estudiar en un momento determinado; si no, la fluidez de la simulación decaerá con el transcurso de la misma. Se recomienda que cuando esté interesado en la simulación visual desactive todas las estadísticas de los elementos; y cuando necesite estadísticas, no espere una simulación visual fluida en simulaciones medias-largas.

La tercera pestaña de la pantalla de configuración, la pestaña **"Avanzada"**, se activa haciendo clic sobre ella.

Y tras hacerlo aparecerá la ventana de configuración avanzada del nodo LSRA. Es fácil saber en todo momento que nos encontramos en esta área porque aparece la imagen de una caja de herramientas, indicando que estamos en la zona donde se puede configurar todo mucho más detalladamente.



Lo primero que nos encontramos es un deslizador marcado como **"Potencia de conmutación"**, donde inicialmente se puede observar que está seleccionada la cantidad de 1 Mbps. Esta propiedad se refiere al número de bits que el LSRA es capaz de conmutar en un segundo y determinará la cantidad de tráfico, y por tanto paquetes, que el LSRA puede reenviar al nodo siguiente en el camino durante la simulación, la congestión que es capaz de causar a dicho nodo, etcétera. Su unidad de medida es "Megabits por segundo (Mbps)" y los valores permitidos van desde 1 a 10240 Mbps, o lo que es lo mismo, desde 1 Mbps a 10 Gbps (Gigabits por segundo), pues el escenario a simular se supone que será una troncal

de datos y los volúmenes de información serán grandes. La forma de variar el valor y seleccionar el que deseamos es moviendo el deslizador de izquierda a derecha o bien colocando el foco sobre el deslizador y usar las teclas de izquierda y derecha del teclado.

Lo siguiente que nos encontramos es un deslizador marcado como **"Tamaño del buffer de entrada"**, donde inicialmente se puede observar que está seleccionada la cantidad de 1 MB. Esta propiedad se refiere al número octetos que el nodo puede almacenar temporalmente mientras procesa otros paquetes, antes de comenzar a descartar paquetes. Su unidad de medida es "Megabyte (MB)" y los valores permitidos van desde 1 a 1024 MB, o lo que es lo mismo, desde 1 MB a 1 GB (Gigabyte). La forma de variar el valor y seleccionar el que deseamos es moviendo el deslizador de izquierda a derecha o bien colocando el foco sobre el deslizador y usar las teclas de izquierda y derecha del teclado.

Posteriormente se encuentra un deslizador marcado como **"Tamaño de la DMGP"**, donde inicialmente se puede observar que está seleccionada la cantidad de 1 KB. Esta propiedad se refiere al número octetos que el LSRA puede almacenar temporalmente para su posible retransmisión local; es decir, a mayor DMGP, mayor probabilidad de que este paquete pueda servir una petición de retransmisión de un paquete descargado en otro nodo de la red. Su unidad de medida es "Kilobyte (KB)" y los valores permitidos van desde 1 a 102400 KB (100 MB). La forma de variar el valor y seleccionar el que deseamos es moviendo el deslizador de izquierda a derecha o bien colocando el foco sobre el deslizador y usar las teclas de izquierda y derecha del teclado.

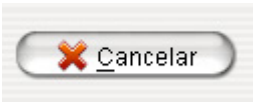
La última opción, llamada **"Generar estadísticas para este LSRA"** hará, si se marca, que el simulador guarde toda la información necesaria para construir unas estadísticas completas del funcionamiento del LSRA desde el inicio de la simulación hasta el final. Por defecto está deshabilitada. Si se deja sin marcar esta opción, no se podrán saber datos estadísticos del nodo LSRA de ninguna de las maneras.

> *¡Nota! La generación de estadísticas para cualquiera de los elementos de la topología consume una ingente cantidad de recursos debido al gran volumen de información que maneja el simulador. Por defecto están desactivadas y se recomienda que deje esta opción así y la active exclusivamente en aquellos nodos que quiera estudiar en un momento determinado; si no, la fluidez de la simulación decaerá con el transcurso de la misma. Se recomienda que*

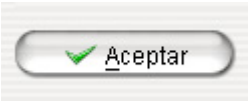
cuando esté interesado en la simulación visual desactive todas las estadísticas de los elementos; y cuando necesite estadísticas, no espere una simulación visual fluida en simulaciones medias-largas.

Cualquier modificación que se haga en la pestaña de configuración **"Avanzada"** provocará que en la ventana **"Rápida"** aparezca el tipo de LSRA como **"Personalizado"**, pero no es una cosa de la que deba preocuparse.

Si en cualquier momento decide dar marcha atrás y no insertar finalmente el nodo LSRA en el escenario, pulse el botón **"Cancelar"**. La ventana se cerrará y volverá al escenario que seguirá como antes de intentar insertar el LSRA.

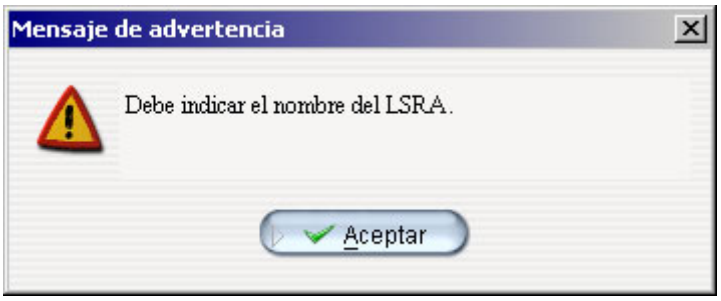


Una vez que haya terminado de configurar el nodo LSRA como desee, pulse el botón **"Aceptar"** para añadir el nuevo nodo LSRA al escenario.

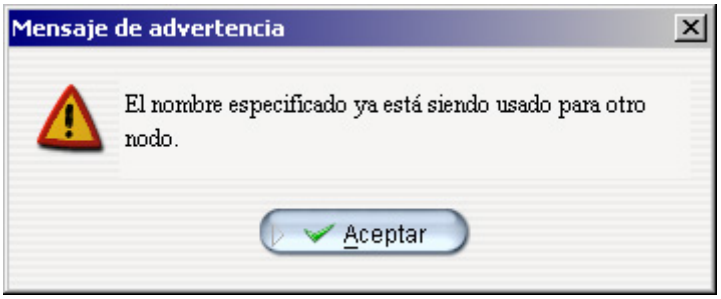


Si en este momento todo está correcto, la ventana de configuración del nodo LSRA desaparecerá y volverá a la pantalla de diseño, donde ya se habrá insertado el nuevo nodo LSRA correctamente.

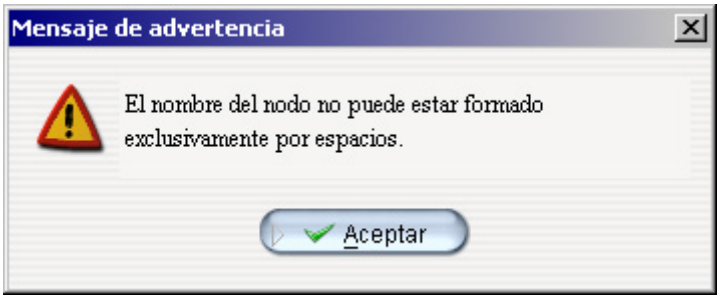
Sin embargo, puede que al pulsar el botón **"Aceptar"**, aparezca alguna ventana indicándole de un error, como vemos en las siguientes figuras.



Esta ventana indica que se le ha olvidado escribir el nombre del LSRA. Un LSRA siempre tiene que tener un nombre, independientemente de que se desee mostrar o no en las pantallas de diseño y simulación.



Esta ventana indica que el nombre que está seleccionando para el nodo LSRA lo está usando otro nodo de los que están insertados ya en el escenario. En este simulador cada nodo debe tener un nombre distinto, no puede haber nodos con nombres repetidos.



Esta ventana indica que en el campo donde debería introducir el nombre del nodo LSRA, ha introducido sólo espacios en blanco o tabuladores. No se permiten nombres formados

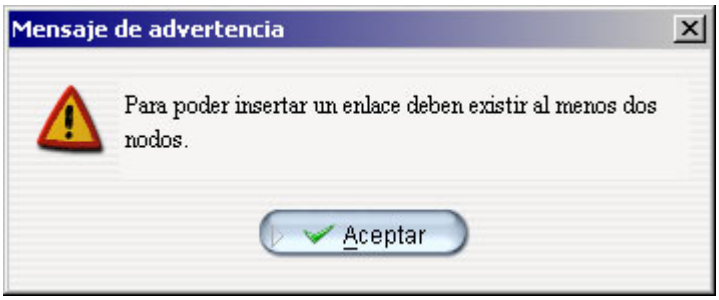
exclusivamente por espacios en blanco, aunque se permiten nombres que tengan espacios en blanco además de otros caracteres.

En cualquiera de los casos anteriores, usted debe pulsar sobre el botón **"Aceptar"** de la ventana que le indica el error, con lo que volverá de nuevo a la ventana de configuración del nodo LSRA y podrá corregir el error. Posteriormente continúe como si no hubiese aparecido un mensaje de error nunca.

3.3.5.4.1.7. Enlace

Un enlace es el elemento que une dos nodos cualesquiera de la red. Es imposible que una red no tenga enlaces pues es por ellos por los que fluye el tráfico. Para insertar un enlace hay que hacer clic sobre el correspondiente icono, señalado en la siguiente figura con un círculo rojo.

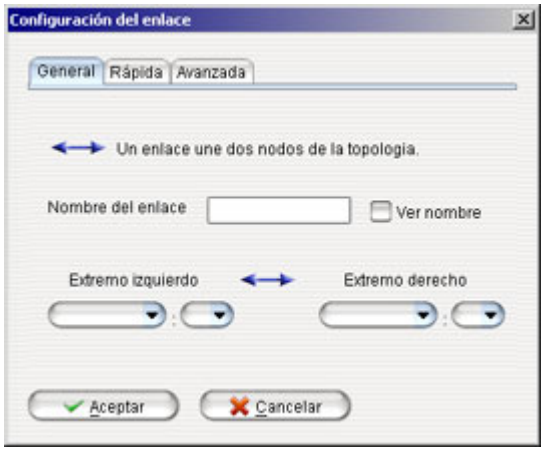
Tras lo cual pueden ocurrir dos cosas. La primera de ella es que aparezca un mensaje de error como se muestra en la siguiente figura.



Como hemos comentado, un enlace une dos nodos de la topología. La ventana de configuración del enlace va a solicitar que se especifique dichos extremos. Si se intenta insertar un enlace sin haber antes insertado un par de nodos, este dato no se puede especificar, por lo que Open SimMPLS 1.0 detecta automáticamente si hay dos o más nodos insertados ya en el escenario y si no es así, no permite insertar ningún enlace. Esto evita incoherencias a la hora de formar el escenario de simulación. Simplemente debemos

hacer clic en el botón “Aceptar” e insertar los dos nodos que deseamos unir con el enlace antes de intentar crear dicho enlace.

Si por el contrario ya hemos insertado un par de nodos que poder unir, no habrá problemas a la hora e intentar añadir un enlace y lo que nos aparecerá en la ventana de escenario será la pantalla de configuración del enlace, que se muestra a continuación.

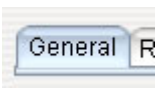


La pantalla de configuración del enlace se compone de tres partes, separadas por pestañas:

- **General:** donde se especifican parámetros generales del enlace. Casi no varía entre los distintos elementos permitidos por Open SimMPLS 1.0
- **Rápida:** donde podemos hacer una configuración básica muy rápida y sencilla del enlace.
- **Avanzada:** donde podemos refinar mucho más la configuración del enlace a costa de dedicar más tiempo y tener más conocimientos.

No es necesario utilizar las dos últimas opciones siempre; basta con utilizar una de ellas para configurar correctamente el enlace.

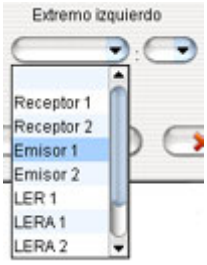
La pestaña **“General”**, se activa haciendo clic sobre ella; aunque siempre que abre la pantalla de configuración, es esta pestaña la que aparece seleccionada por defecto.



En esta pestaña, que se muestra dos figuras atrás, aparece un icono que representa al enlace. Junto al icono aparece una pequeña descripción de qué hace este elemento; en este caso, unir dos nodos de la topología. También aparece un campo **"Nombre del enlace"** donde se nos permite escribir. En él deberemos escribir un nombre para el enlace, nombre que servirá para referirse al él en todo el escenario y nombre que se mostrará, si se desea, en las ventanas de diseño y simulación. Justo esto último es lo que se puede configurar en el *checkbox* titulado **"Ver nombre"**. Si lo marcamos, el nombre del enlace que hemos escrito se mostrará en las distintas pantallas de la ventana de escenario. Si lo desmarcamos, no se verá. En la parte inferior, por último, tenemos un conjunto de listas desplegables, como podemos ver en la siguiente figura.

Las dos primeras sirven para configurar el primer nodo que estará conectado el enlace, al que llamaremos extremo izquierdo. Las dos últimas sirven para lo mismo, pero en referencia al nodo del otro extremo del enlace, que llamaremos extremo derecho. En realidad esta nomenclatura no tiene nada que ver con la posición de los nodos en la pantalla de diseño, sino que es una forma de nombrar los extremos. Se podría haber llamado "Extremo A" y "Extremo B" y hubiese dado igual.

Para seleccionar los nodos que conectará el enlace, se ha de hacer de izquierda a derecha en estas cuatro listas desplegables. En la primera (izquierda) se muestran todos aquellos nodos que tienen puertos libres y por tanto pueden admitir conexiones. Es lo primero que hay que elegir.

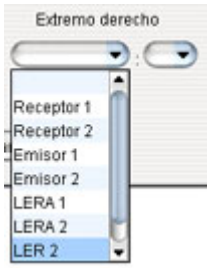


Una vez hayamos seleccionado el primero de los nodos al que se conectará el enlace, debemos elegir uno de sus puertos libres, donde irá "enchufado" el enlace. La segunda lista desplegable muestra ahora los puertos libres del nodo seleccionado en la primera lista.

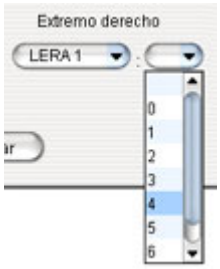
Seleccionando uno de ellos habrá concluido la configuración del primero de los nodos que une el enlace. Hay que tener en cuenta que en Open SimMPLS 1.0 no todos los nodos tienen igual número de puertos; LER, LERA, LSR y LSRA tienen 8 puertos cada uno; emisor y Receptor tiene sólo 1. Una vez seleccionado el puerto del extremo izquierdo, se muestran en la tercera lista desplegable todos aquellos nodos que tienen puertos libres y que se pueden conectar (son compatibles) con el primer nodo seleccionado. ¿Qué significa esto? En MPLS, no todos los nodos pueden estar interconectados entre sí. En Open SimMPLS 1.0 las posibilidades de interconexión están restringidas de la siguiente forma:

- **Emisor:** sólo se puede conectar a LER o LERA.
- **Receptor:** sólo se puede conectar a LER o LERA.
- **LER:** se pueden conectar a cualquier tipo de nodo.
- **LERA:** se pueden conectar a cualquier tipo de nodo.
- **LSR:** sólo se pueden conectar a LSR, LSRA, LER o LERA
- **LSRA:** sólo se pueden conectar a LSR, LSRA, LER o LERA
- Un nodo no se puede conectar a si mismo mediante un enlace.
- Un par de nodos sólo pueden estar conectados por un enlace.

Así que si en un momento dado hay un nodo con puertos libres que no aparece en esta lista desplegable, es porque no puede ser conectado al nodo "izquierdo" que hemos seleccionado. Seleccionamos el nodo del segundo extremo de esta lista.

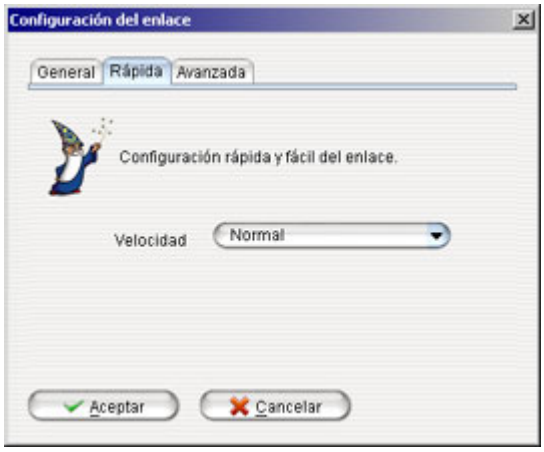


Y ya por último, debemos elegir el puerto de dicho nodo al que queremos conectar el enlace. Igual que ocurrió antes, pero esta vez en la cuarta lista desplegable, se nos mostrarán los puertos libres del nodo que hemos elegido como extremo derecho.

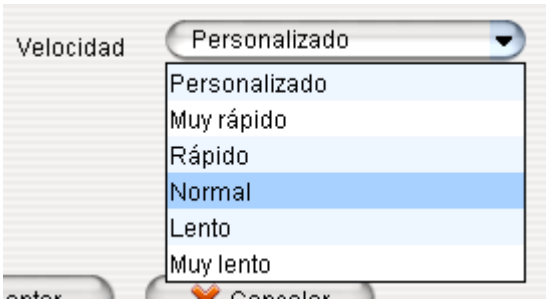


La segunda pestaña de la pantalla de configuración, la pestaña **"Rápida"**, se activa haciendo clic sobre ella.

Y tras hacerlo aparecerá la ventana de configuración rápida del enlace. Es fácil en todo momento saber en qué zona de la configuración nos encontramos. En la pestaña de configuración rápida, aparece la imagen de un mago, indicando que estamos en la zona donde se puede configurar todo de forma transparente, muy rápida y muy sencilla.

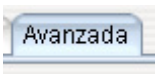


En el caso del enlace, la configuración rápida se limita a tener que seleccionar de la lista desplegable llamada **"Velocidad"**, la velocidad de propagación del enlace; la velocidad con la que transportará el tráfico de red de un extremo a otro. Son valores comunes, que pueden ser requeridos con cierta asiduidad y que por tanto están predefinidos para seleccionarlos de forma rápida.

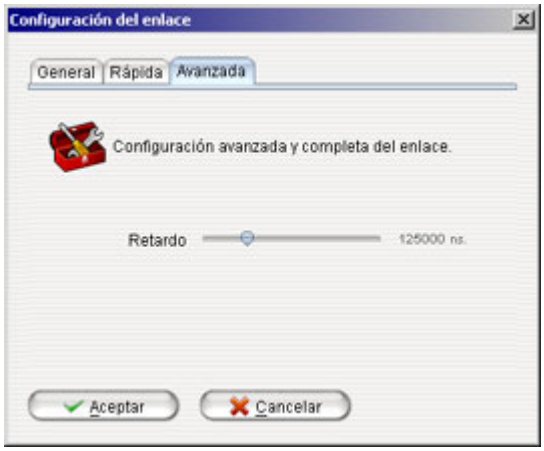


Se puede configurar desde aquí el enlace para que sea normal, rápido, lento, etcétera. El tipo de velocidad **"Personalizado"** no indica una velocidad en si, sino que aparece aquí cuando la velocidad del enlace no se ha configurado en la pestaña **"Rápida"** sino en la pestaña **"Avanzadas"** y por tanto no coincide con alguno de estas predefinidas.

La tercera pestaña de la pantalla de configuración, la pestaña **"Avanzada"**, se activa haciendo clic sobre ella.



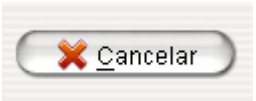
Y tras hacerlo aparecerá la ventana de configuración avanzada del enlace. Es fácil saber en todo momento que nos encontramos en esta área porque aparece la imagen de una caja de herramientas, indicando que estamos en la zona donde se puede configurar todo mucho más detalladamente.



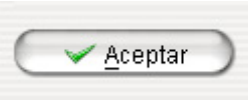
La configuración avanzada se limita a tener que seleccionar el retardo deseado para el enlace mediante un deslizador llamado **"Retardo"**; La forma de variar el valor y seleccionar el que deseamos es moviendo el deslizador de izquierda a derecha o bien colocando el foco sobre el deslizador y usar las teclas de izquierda y derecha del teclado. El retardo se mide en "nanosegundos (ns.)" y los valores permitidos van desde 1 ns. hasta 500.000 ns.

Cualquier modificación que se haga en la pestaña de configuración **"Avanzada"** provocará que en la ventana **"Rápida"** aparezca la velocidad del enlace como **"Personalizado"**, pero no es una cosa de la que deba preocuparse.

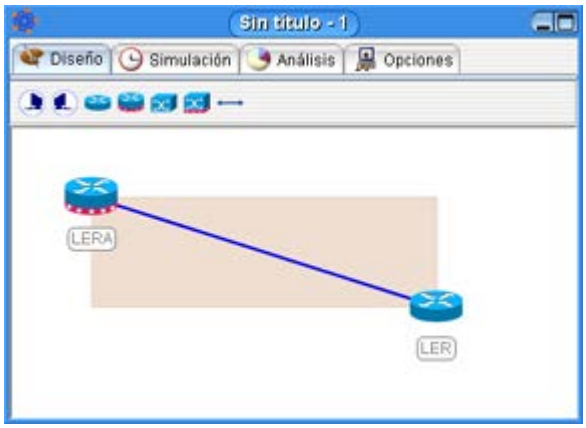
Si en cualquier momento decide dar marcha atrás y no insertar finalmente el enlace en el escenario, pulse el botón **"Cancelar"**. La ventana se cerrará y volverá al escenario que seguirá como antes de intentar insertar el enlace.



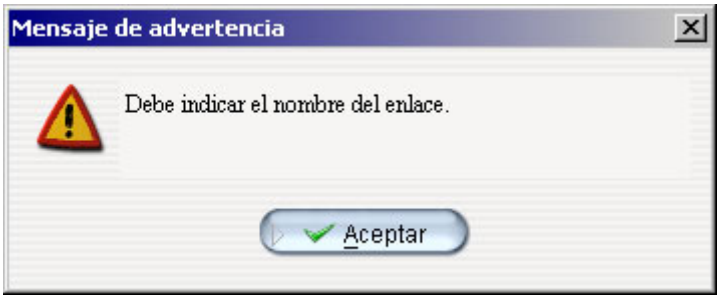
Una vez que haya terminado de configurar el enlace como desee, pulse el botón **"Aceptar"** para añadir el nuevo enlace al escenario.



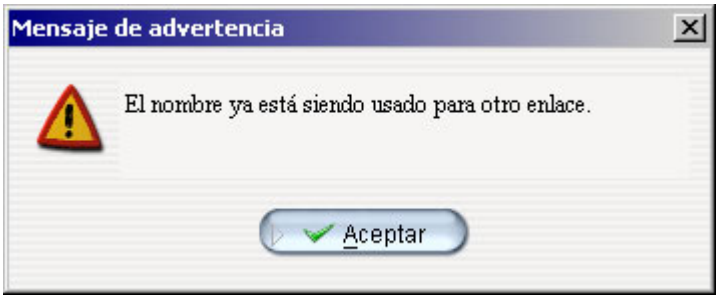
Si en este momento todo está correcto, la ventana de configuración del enlace desaparecerá y volverá a la pantalla de diseño, donde ya se habrá insertado el nuevo enlace correctamente.



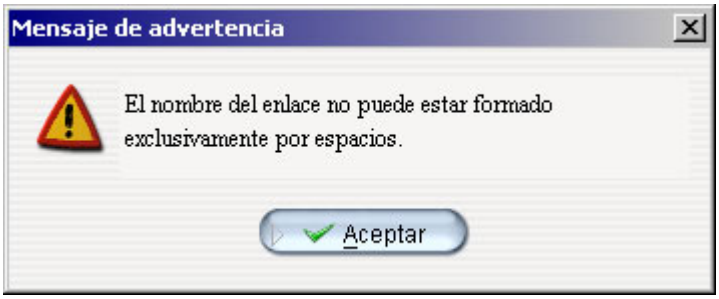
Sin embargo, puede que al pulsar el botón **"Aceptar"**, aparezca alguna ventana indicándole de un error, como vemos en las siguientes figuras.



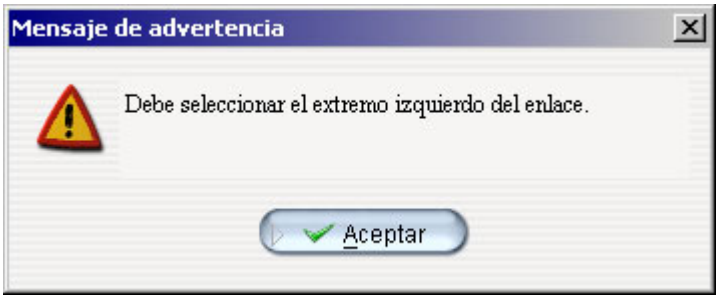
Esta ventana indica que se le ha olvidado escribir el nombre del enlace. Un enlace siempre tiene que tener un nombre, independientemente de que se desee mostrar o no en las pantallas de diseño y simulación.



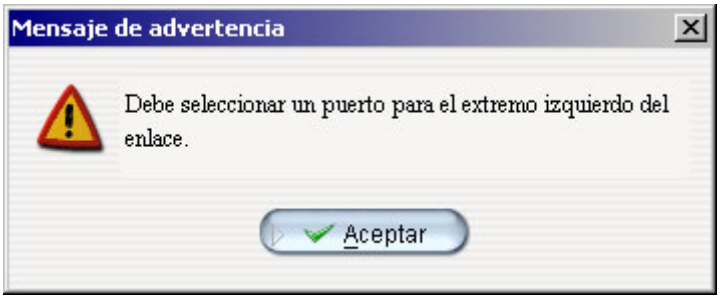
Esta ventana indica que el nombre que está seleccionando para el enlace lo está usando otro enlace de los que están insertados ya en el escenario. En este simulador cada enlace debe tener un nombre distinto, no puede haber enlaces con nombres repetidos.



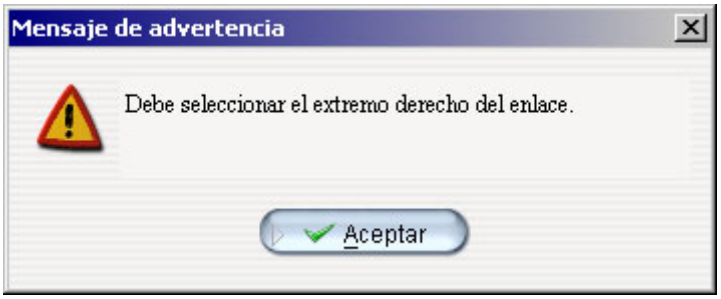
Esta ventana indica que en el campo donde debería introducir el nombre del enlace, ha introducido sólo espacios en blanco o tabuladores. No se permiten nombres formados exclusivamente por espacios en blanco, aunque se permiten nombres que tengan espacios en blanco además de otros caracteres.



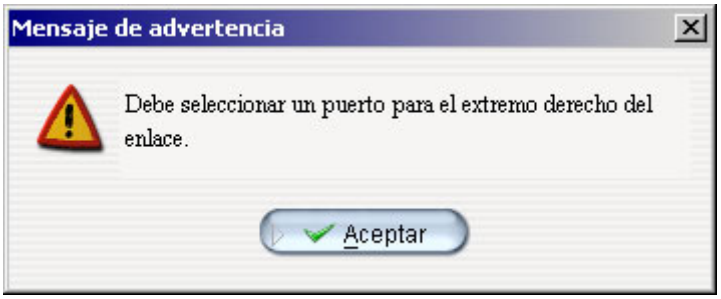
Esta ventana indica que no ha seleccionado el nodo “Extremo izquierdo” al que se conectará el enlace. Un enlace no puede quedar sin conectar por ninguno de sus dos extremos.



Esta ventana indica que, aunque ha seleccionado el nodo que será extremo izquierdo del enlace, no ha seleccionado un puerto del mismo al que debe conectarse el enlace. No basta con seleccionar el nodo extremo, ha de seleccionar también el puerto.



Esta ventana indica que no ha seleccionado el nodo "Extremo derecho" al que se conectará el enlace. Un enlace no puede quedar sin conectar por ninguno de sus dos extremos.



Esta ventana indica que, aunque ha seleccionado el nodo que será extremo derecho del enlace, no ha seleccionado un puerto del mismo al que debe conectarse el enlace. No basta con seleccionar el nodo extremo, ha de seleccionar también el puerto.

En cualquiera de los casos anteriores, usted debe pulsar sobre el botón **"Aceptar"** de la ventana que le indica el error, con lo que volverá de nuevo a la ventana de configuración

del enlace y podrá corregir el error. Posteriormente continúe como si no hubiese aparecido un mensaje de error nunca.

3.3.5.4.2. Modificar elementos insertados

Una vez que se han insertado todos los elementos deseados en la topología, podemos tener la necesidad de modificar algún aspecto de la configuración de algún nodo, de algún enlace… De hecho será lo que normalmente se haga puesto que al ser un simulador para la realización de pruebas, es más que probable que con un mismo escenario se quiera estudiar el comportamiento el variar alguno de lo componentes de la red.

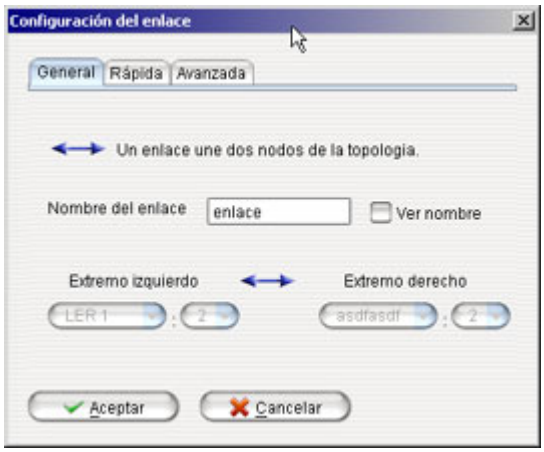
Para ello, Open SimMPLS 1.0 ofrece la posibilidad de reeditar las características de los elementos insertados. Básicamente esto consiste en que volverá a aparecer la ventana de configuración del elemento deseado y se podrán modificar los parámetros como si se estuviese insertando de nuevo el elemento. Veamos como.

3.3.5.4.2.1. Enlaces

Para editar un enlace que ya está insertado, se debe estar en la pantalla de diseño, que es en la única en la que se pueden hacer cambios en la topología. Una vez allí, ha de situarse el puntero del ratón sobre el enlace hasta que el cursor cambie de una flecha a una mano, indicando que está sobre un elemento de la topología. En ese momento hay que hacer clic con el botón derecho del ratón para obtener un menú contextual.

Y de las opciones que aparecen seleccionamos **“Propiedades”**, lo cual mostrará la pantalla de configuración del enlace, con los datos con los que actualmente está configurado el

enlace seleccionado. Del resto de opciones del menú emergente nos olvidaremos por el momento.



A partir de este momento, el enlace se configura como ya se ha visto en este manual de usuario en el apartado “Insertar enlaces”; todo es exactamente igual. Todo a excepción de que cuando se está modificando un enlace ya insertado, no podemos seleccionar los nodos extremos ni sus puertos (aparecen en gris, deshabilitados). Esos son los rasgos que definen a un enlace y no se pueden modificar. Si se desea modificar los nodos que une un enlace, hay que eliminar el enlace y volverlo a crear con las características deseadas.

3.3.5.4.2.2. Nodos

Para editar un nodo que ya está insertado, se debe estar en la pantalla de diseño, que es en la única en la que se pueden hacer cambios en la topología. Una vez allí, ha de situarse el puntero del ratón sobre el nodo hasta que el cursor cambie de una flecha a una mano, indicando que está sobre un elemento de la topología. En ese momento hay que hacer clic con el botón derecho del ratón para obtener un menú contextual.

Y de las opciones que aparecen seleccionamos **“Propiedades”**, lo cual mostrará la pantalla de configuración del nodo, con los datos con los que actualmente está configurado el nodo seleccionado. Del resto de opciones del menú emergente nos olvidaremos por el momento.

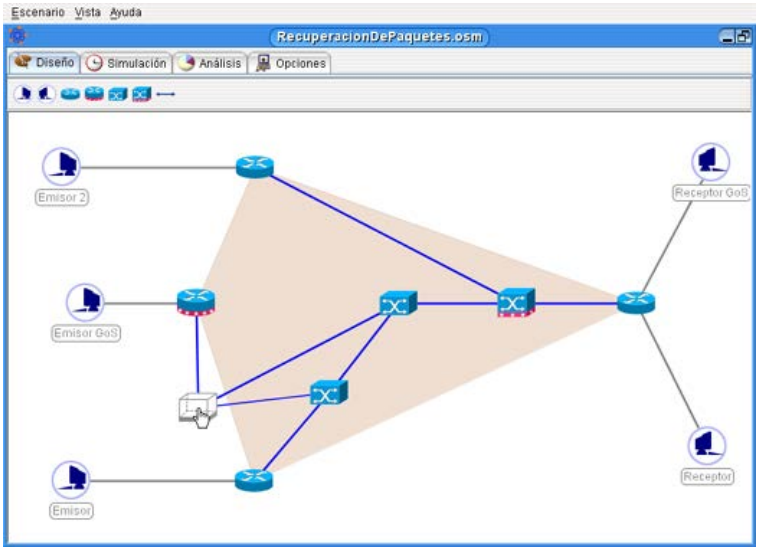
A partir de este momento, el nodo se configura como ya se ha visto en este manual de usuario en los apartados “Insertar emisor”, “Insertar receptor”, “Insertar LER”, “Insertar LERA”, “Insertar LSR” e “Insertar LSRA”; todo es exactamente igual. Todo a excepción de que cuando se está modificando un nodo ya insertado, no podemos modificar desde la ventana de configuración la posición en la topología. La miniatura de selección de la posición no parece y en su lugar aparece un aspa roja. Ya veremos que hay otras formas más cómodas para posicionar en el lugar correcto un nodo.

3.3.5.4.3. Rediseñar la topología

Una vez que tenemos bien configurados los elementos de la topología, podemos necesitar hacer cambios en la distribución de los elementos, podemos querer eliminar un elemento, etcétera.

3.3.5.4.3.1. Cambiar la distribución de los elementos

Si desea variar de posición los elementos de la topología durante el diseño, ha de estar en la pantalla de diseño. Una vez allí, simplemente sitúese sobre un nodo cualquiera hasta que el cursor del ratón aparezca con forma de mano, indicando que está sobre un objeto. A continuación pulse el botón izquierdo del ratón y sin soltarlo desplace el ratón.



El nodo aparecerá en blanco, con forma de armazón, y se desplazará a medida que usted mueva el ratón por el área de diseño. Los enlaces que partan o lleguen de ese nodo, se moverán también con él. Una vez que el nodo esté situado en el lugar que usted desee, suelte el botón izquierdo del ratón y habrá movido a su gusto el nodo desde una a otra posición.

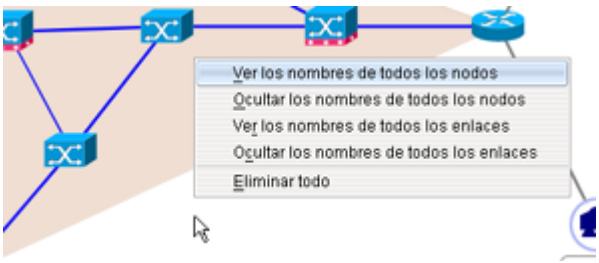
Sólo los nodos pueden ser movidos. Para mover un enlace debe mover los nodos que une dicho enlace. Un enlace por sí solo no puede ser movido.

3.3.5.4.3.2. Mostrar el nombre de los elementos

Si desea mostrar u ocultar el nombre de un elemento de la topología, ha de estar en la pantalla de diseño. Una vez allí, simplemente sitúese sobre un nodo o enlace cualquiera hasta que el cursor del ratón aparezca con forma de mano, indicando que está sobre un objeto. A continuación haga clic con el botón derecho del ratón y aparecerá un menú emergente con opciones sobre ese objeto.

Haciendo clic sobre la opción **"Ver nombre"**, usted podrá controlar si desea o no ver el nombre del elemento. Si el nombre del elemento está siendo mostrado actualmente, esta operación provocará que se oculte. En caso contrario, provocará que se muestre.

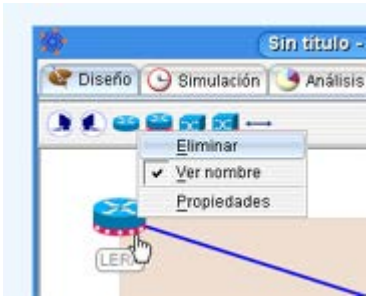
Esta operación actúa sobre un único elemento, por ejemplo un enlace o un nodo. Sin embargo es habitual querer realizar en un momento dado esta misma operación sobre varios nodos o enlaces a la vez. Para ello, Open SimMPLS 1.0 incorpora otros mecanismos. Si hace clic con el botón secundario sobre una zona de la pantalla de diseño donde no haya objetos (donde el puntero del ratón sea una flecha y no una mano), aparecerá un menú emergente.

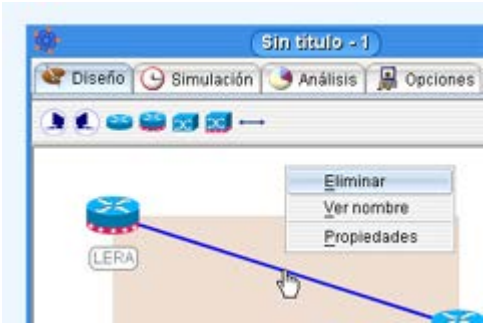


En el se muestran varias opciones. Las que nos interesan en este momentos son **"Ver los nombres de todos los nodos"**, **"Ocultar los nombres de todos los nodos"**, **"Ver los nombres de todos los enlaces"** y **"Ocultar los nombres de todos los enlaces"**, que realizan la operación de ocultar o mostrar el nombre, sobre todos los enlaces o sobre todos los nodos. Para utilizar cualquiera de estas opciones, simplemente haga clic con el botón principal del ratón en una de ellas.

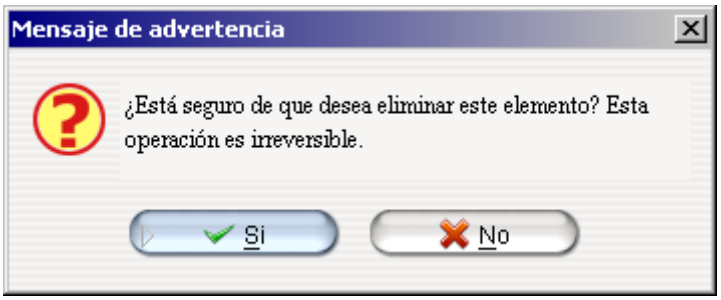
3.3.5.4.3.3. Eliminar elementos

Usted puede eliminar en cualquier momento un elemento de la topología. Para ello deberá estar situado en la pantalla de diseño. Si desea mostrar u ocultar el nombre de un elemento simplemente sitúese sobre un nodo o enlace cualquiera (el que desee eliminar) hasta que el cursor del ratón aparezca con forma de mano, indicando que está sobre un objeto. A continuación haga clic con el botón derecho del ratón y aparecerá un menú emergente con opciones sobre ese objeto.





Una de las opciones es **"Eliminar"**. Si usted hace clic con el botón principal del ratón en dicha opción, aparecerá la siguiente ventana de consulta.



Donde se indica que no podrá deshacer esta operación. Si aún desea eliminar el elemento, pulse sobre el botón **"Si"**. Si ha cambiado de idea o se había equivocado de elemento u opción, pulse sobre **"No"**. En el primero de los casos, volverá a la pantalla de diseño y el elemento habrá desaparecido del escenario. Lo habrá eliminado. En el segundo caso, volverá a la pantalla de diseño y todo estará como estaba; el elemento seguirá intacto.

Este procedimiento tiene una excepción. Si pulsamos sobre el botón **"Si"** en la ventana anterior al intentar eliminar un nodo receptor, es posible que nos aparezca la siguiente ventana de advertencia.

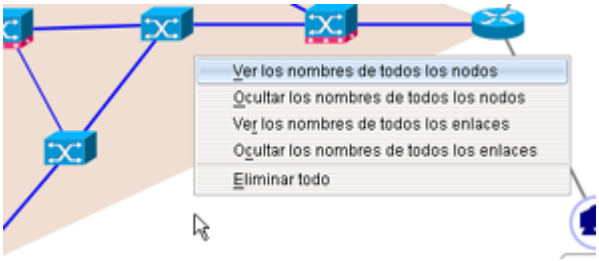
Open SimMPLS 1.0 intenta en todo momento que no se pierda la consistencia del escenario. Si usted pudiese eliminar un receptor al que un nodo emisor está dirigiendo su tráfico, podría olvidarse de reconfigurar dicho nodo emisor y eso sería desastroso. Para evitarlo, el simulador no le permitirá eliminar un nodo receptor mientras que hay un nodo emisor que le envíe tráfico. Si desea realmente eliminar el nodo receptor, antes debe reconfigurar los emisores para que ninguno envíe tráfico a él; o bien eliminar el emisor que esté enviando tráfico a este receptor. Pulse **"Aceptar"**; volverá a la pantalla de diseño y todo seguirá como estaba.

Si no le ha aparecido esta ventana, no se preocupe porque no será necesario. Siga la norma general que se sigue para todos los elementos.

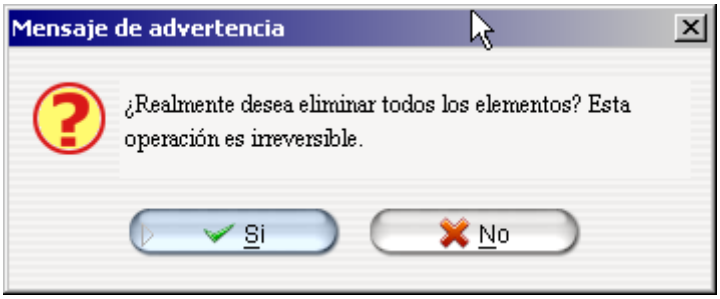
En cualquier caso, es importante saber dos cosas:

- Cuando elimina un enlace, elimina exclusivamente ese enlace.
- Si elimina un nodo, elimina ese nodo y todos los enlaces que parten o llegan a él.

Además de la operación anterior, que se aplica a un elemento concreto de la topología (y los enlaces pertinentes, en el caso de un nodo), Open SimMPLS 1.0 permite la eliminación de todo el escenario, es decir, de todos los elementos insertados hasta el momento. Para realizar esta operación, hay que hacer clic sobre un parte de la pantalla de diseño donde no haya elementos (donde el puntero del ratón no sea un mano, sino una flecha).

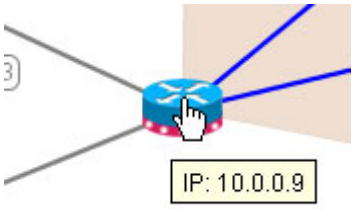


Y posteriormente, de entre todas las opciones que aparecerán en el menú que se desplegará, hay que hacer clic en **"Eliminar todo"**. Tras hacer esto, parecerá la siguiente ventana.

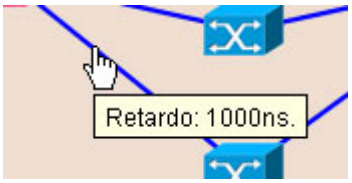


Que indica que la operación que se ha seleccionado no tiene marcha atrás si continuamos con ella. Si desea eliminar todos los elementos insertados, pulse sobre el botón **"Si"**. La ventana desaparecerá y volverá a la pantalla de diseño, que ahora estará limpia, como si el escenario acabase de ser creado. Si no desea eliminar todo, pulse en el botón **"No"** y volverá a la pantalla de diseño, donde todo estará como antes.

Por último, dos aspectos importantes adicionales. Este simulador se basa en IP sobre MPLS; así pues cada nodo debe tener una IP, aunque no es parametrizable; el simulador asigna dichas IP de forma automática. Para conocer la IP de un nodo, simplemente hay que colocar el puntero del ratón sobre él (en el área de diseño) y esperar un segundo a que aparezca un recuadro indicándonos dicho valor.

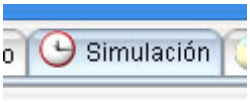


Del mismo modo, un enlace tiene un retardo concreto que, aunque se puede averiguar abriendo la ventana de configuración del mismo, lo cierto es que resulta más sencillo hacerlo del mismo modo que obtenemos la IP, pero colocándonos esta vez sobre un enlace en lugar de sobre un nodo.

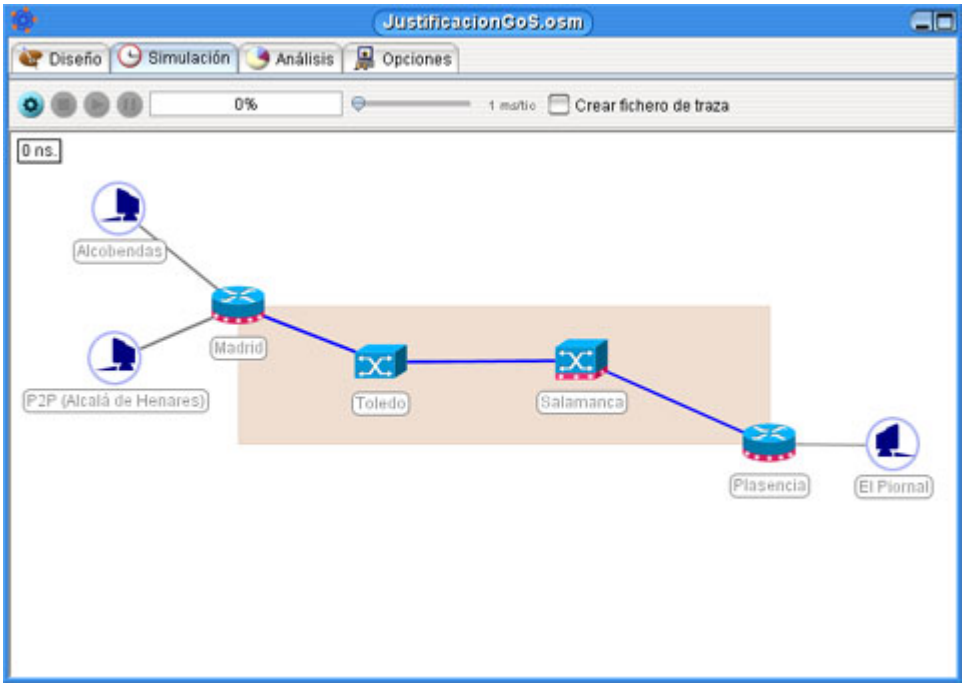


3.3.5.5. Área de simulación

Debemos seleccionar trabajar en el área de simulación cuando hemos finalizado de crear y configurar la topología del escenario en el área de diseño y queramos ver qué tal se comportará dicha topología una vez comience a generarse tráfico, saturaciones, caídas de enlaces, etcétera. Para seleccionar este modo de trabajo, se debe hacer clic sobre la pestaña **"Simulación"** de la ventana de escenario.



Y entonces el simulador conmutará al área de simulación que presentará el siguiente aspecto.



En esta figura concreta hay un escenario que sería el que se habría diseñado en el área de diseño.

Como se puede observar, el área de simulación presenta una estructura similar a la de diseño. La diferencia estriba en que en el lugar donde aparecían los elementos a insertar en el escenario, en el área de la simulación aparecen unos iconos para manejar el funcionamiento de la simulación; además de otros elementos. La mayor parte de los que se puede realizar en el área de simulación se maneja a través de las opciones de la barra de herramientas de simulación.

En los siguientes apartados veremos para qué se usa cada una de estas opciones.

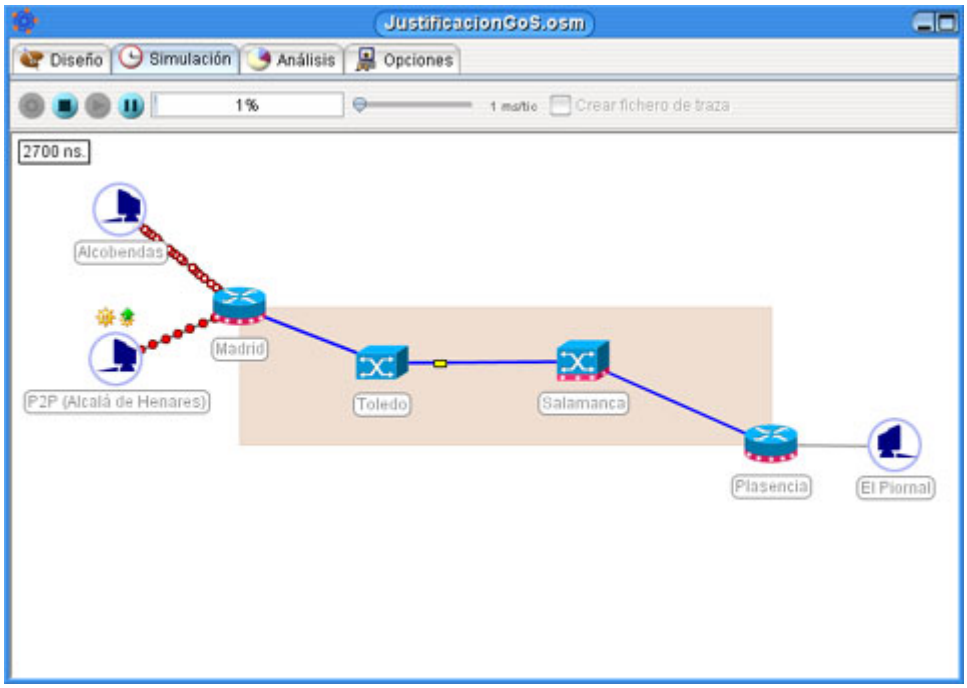
3.3.5.5.1. Comenzar la simulación

Como en este momento la topología debe estar finalizada (se finalizó en el área de diseño), simplemente hay que poner en funcionamiento la simulación. Esto se hace mediante un

clic en el icono que simula un engranaje, que se muestra señalado con un círculo rojo en la siguiente figura.

Además es el único de los cuatro iconos que se puede activar ya que en principio los otros aparecen deshabilitados.

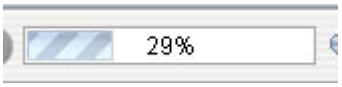
En el momento de realizar esta acción, los nodos emisores comenzarán a generar tráfico y éste se podrá ver fluyendo en el área de simulación desde los emisores a sus respectivos nodos receptores, según estén configurados. La interpretación de la simulación visual se comentará un poco más adelante, pero como muestra de lo que ocurriría, la siguiente figura muestra una captura del inicio de una simulación.



Además de comenzar la simulación, diversas opciones se activan y desactivan para permitir nuevas acciones en la simulación. Tras comenzar la misma, los iconos de manejo de la simulación han cambiado, como muestra la siguiente figura.

Como la simulación está en funcionamiento, no tiene sentido permitir el comienzo de la misma, puesto que ya está en ejecución; así que el botón de inicio de la simulación se desactiva. Sin embargo, precisamente por el mismo motivo, tienen sentido los botones utilizados para finalizar y parar la simulación, lo cual también se muestra en la figura anterior.

En cualquier caso, mientras la simulación está en funcionamiento, una barra de progreso indica en todo momento el porcentaje de la simulación que se lleva alcanzado. Ya veremos que una simulación debe tener una duración finita y por tanto un final. Esta barra de progreso nos orientará en todo momento de cuánto queda, en términos de porcentaje, para llegar al fin.



También un contador existente dentro del área de simulación, muestra el número de nanosegundos que se llevan simulados. No se refiere a tiempo real sino a tiempo simulado.

3.3.5.5.2. Detener la simulación

Para detener la simulación, es necesario que la simulación esté en funcionamiento, pues es sólo en ese momento cuando tiene sentido poder pararla. Además es necesario que esté en funcionamiento porque sólo en ese momento estará activo el icono que hay que pulsar, que se muestra en la siguiente figura, resaltado en rojo (el icono estándar de la pausa en vídeos, equipos de música, etcétera).

Al hacerlo, inmediatamente la simulación se detiene, no progresa. Todo queda congelado: los paquetes en tránsito, los paquetes siendo descartados, el tiempo, etcétera. Es bueno utilizar esta acción de detener la simulación, cuando deseemos observar algo en la simulación que ocurra de forma efímera, como la caída de un paquete, la señalización en un momento dado, etcétera. En este estado la simulación está en pausa.

Además de todo lo anterior, la barra de herramientas de simulación cambia, activando y desactivando los iconos correspondientes al estado de pausa. Como se muestra en la figura siguiente.

3.3.5.5.3. Reanudar la simulación

Para reanudar la simulación, es necesario que la simulación esté en modo pausa, o sea, detenida, pues es sólo en ese momento cuando tiene sentido poder reanudarla. Además es necesario que esté en pausa porque sólo en ese momento estará activo el icono que hay que pulsar, que se muestra en la siguiente figura, resaltado en rojo (el icono estándar del *play* en vídeos, equipos de música, etcétera).

Al seleccionar esta acción, la simulación volverá al modo de ejecución, como si no se hubiese parado nunca y además lo hará siguiendo justo por el punto donde se había detenido, conservando el mismo aspecto. Asimismo, como la simulación vuelve a estar en funcionamiento, la botonera de control vuelve a mostrar el aspecto que cuando se comenzaba la ejecución.

3.3.5.5.4. Finalizar la simulación

En cualquier momento, tanto si la simulación está ejecutándose, como si está detenida (pausa), podemos elegir finalizar por completo, como si nunca hubiese comenzado al ejecución de la simulación. No obstante, la última imagen de la simulación quedará fija por si se desea analizar que paso en el momento de finalizar.

Para finalizar la simulación debe hacerse clic con el botón principal del ratón sobre el icono que se muestra en la siguiente figura circunscrito en rojo, que representa el típico símbolo de *stop* de los vídeos, DVD y reproductores.

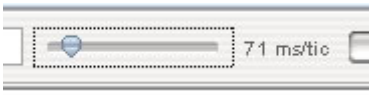
Esta acción producirá que además de finalizar la simulación (no se podrá reanudar en el punto por donde iba, sino que tendrá que empezar de cero), como ahora no está en funcionamiento la simulación la botonera de control muestra el mismo aspecto que al inicio, indicando que la única opción posible que se puede seleccionar es comenzar la simulación.

3.3.5.5.5. Ajustar la velocidad de la simulación

Por defecto, Open SimMPLS 1.0 ofrece la mayor velocidad posible de simulación. Esto dependerá en gran medida del escenario que se haya diseñado (si es más complejo y con más tráfico, se ralentizará más), de la capacidad de su PC, del número de elementos de la topología que estén generando estadísticas, de si está o no activada la generación de un

fichero de traza, etcétera. Así pues no se puede acelerar una simulación más allá de cómo se encuentra ésta en un inicio.

Sin embargo se puede ralentizar, lo cual es muy útil para poder observar con detenimiento cómo van ocurriendo las cosas durante la simulación sin necesidad de tener que detener y reanudar la simulación constantemente. Para ello hay que hacer uso del deslizador que se encuentra en la barra de herramientas de simulación, como muestra la siguiente figura.



Este elemento permite establecer cuántos milisegundos se deben hacer de pausa entre cada vez que el simulador refresca la pantalla para mostrar la evolución continua de la simulación. Por defecto está al valor mínimo permitido, 1 milisegundo, que equivale a no hacer pausa alguna sino simular a la máxima velocidad posible. Se puede elevar este valor hasta 500 milisegundos (medio segundo) entre cada refresco de pantalla con lo cual se logra ralentizar la simulación hasta unos niveles en los cuales sea cómodo poder observar qué pasa en ciertas situaciones.

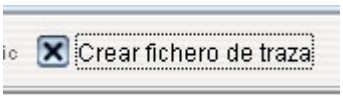
La forma de variar el valor y seleccionar el que deseamos es moviendo el deslizador de izquierda a derecha o bien colocando el foco sobre el deslizador y usando las teclas de izquierda y derecha del teclado. Además esta operación puede (y debe) hacerse cuando la simulación está en funcionamiento ya que vuelve automáticamente a su valor de 1 milisegundo al comenzar la simulación.

3.3.5.5.6. Generar una traza de la simulación

Toda la simulación visual que se puede observar en el área de simulación en tiempo real no es sino la representación gráfica de los valores internos de todos los elementos que componen el escenario, paso a paso. En la mayoría de las ocasiones la simulación visual junto con las estadísticas generadas por los nodos que estén configurados para ello, es más que suficiente para comprender correctamente qué ha ocurrido. Sin embargo, hay veces que es necesario buscar una interpretación numérica, con los datos en la mano, a alguna

gráfica o situación compleja; pero la simulación no tiene marcha atrás. Por ello, es posible generar un fichero de traza donde se almacenen, en texto plano, todos los acontecimientos que han tenido lugar durante la simulación, en qué elemento ha ocurrido, en qué instante, etcétera, para poder ser revisado a posteriori.

Si desea generar este fichero de traza, haga clic con el botón principal del ratón sobre recuadro de selección llamado **"Crear fichero de traza"**, de la barra de herramientas de simulación, como se muestra en la figura siguiente.



Esto sólo puede hacerse antes de que la simulación comience y durará hasta que la simulación finalice; el resto del tiempo, este recuadro permanecerá deshabilitado y no se podrá seleccionar ni deseleccionar.

El fichero de traza llevará el mismo nombre que aparece como título en la ventana de escenario, pero con extensión TXT. Por ejemplo, en la siguiente figura tenemos tendríamos abierto un escenario llamado **"JustificacionGoS.osm"**.

Si activásemos la generación de un fichero de traza como se ha comentado, éste se crearía en el mismo directorio/carpeta donde se encuentra el fichero del escenario abierto y se llamaría **"JustificacionGoS.txt"**. Si el escenario no se hubiese guardado aún, el directorio/carpeta destino del fichero de traza sería el directorio por defecto y el nombre, el que tuviese el escenario, que generalmente sería **"Sin título – 1.osm"**, **"Sin título – 2.osm"**, etcétera.

Una vez generado el fichero de traza, se puede inspeccionar con cualquier editor de texto plano como *Worpad* (Windows) o *KWrite* (Linux).

¡Nota! La generación de traza ralentiza la simulación debido a la latencia del acceso a disco. Si su objetivo es obtener una simulación fluida, obvie la traza; manténgala desactivada. Intente no activar la generación de traza a no ser que realmente quiera obtenerla. Actívela bajo su propio criterio, pero en cualquier caso, de activarla, no espere obtener una simulación visual ágil.

3.3.5.5.7. Interpretación de la simulación visual

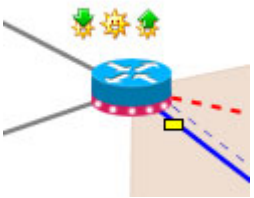
Nos centraremos ahora en la parte central del área de simulación, donde está el escenario que hemos diseñado y donde una vez que se pone en funcionamiento la simulación, aparecerán todo un conjunto de símbolos que hay que interpretar. En los siguientes apartados se supone que la simulación está en funcionamiento o, eventualmente, en pausa.

3.3.5.5.7.1. Elementos

Los elementos del escenario son, como vimos páginas atrás, nodos y enlaces. Cada uno de ellos puede cambiar de forma, color, etcétera en el transcurso de la simulación.

3.3.5.5.7.1.1. General

Todos los nodos pueden realizar diversas acciones durante la simulación. Los nodos emisores, crean paquetes y los envían. Los receptores los reciben y en el camino los LER y LSR, activos o no, los reciben, y los conmutan/encaminan; adicionalmente también crean paquetes propios que envían. Para denotar todo esto visualmente, sobre cada nodo aparece en cada momento una imagen que indica qué operación está realizando. Se mostrará, por ejemplo, como aparece en la siguiente figura.



En este caso sólo se ven sobre el nodo tres imágenes pequeñitas. En realidad podría haber más. Todas las imágenes y sus significados son:

Si un nodo muestra sobre él esta imagen, significa que ha recibido un paquete. De cualquier tipo de tráfico.

Si un nodo muestra sobre él esta imagen, significa que dentro del nodo se ha creado un paquete.

Si un nodo muestra esta imagen sobre él, significa que el nodo ha enviado un paquete que él mismo ha generado, hacia otro nodo.

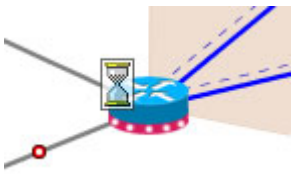
Si un nodo muestra esta imagen sobre él, significa que ha conmutado o encaminado un paquete que le había llegado por un puerto de entrada.

Así, en la figura anterior se puede observar que este LERA ha recibido un paquete, ha generado otro y este último lo ha enviado a un puerto de salida en dirección a otro nodo.

Otro hecho importante para entender la simulación es saber que cuando un nodo tiene una potencia de conmutación o generación de tráfico muy baja, puede darse el caso que durante grandes espacios de tiempo parezca inactivo durante la simulación, cuando realmente está bien conmutando o bien generando un paquete de gran tamaño en relación a dicha potencia o velocidad. Para ello tenemos otra imagen.

Cuando un nodo está ocupado en una tarea que le cuesta mucho tiempo, lo denotará mostrando este reloj de arena sobre él.

En la siguiente figura se ve claro este hecho.



Pues una vez visto los aspectos generales de todos los nodos de la topología, vamos a centrarnos en explicar todos los elementos, uno por uno.

3.3.5.5.7.1.2. Nodos LER

El nodo LER se representa en la topología (y por tanto en la simulación) con el siguiente icono.

Pero durante la simulación, el nodo puede irse congestionando por lo que su aspecto puede cambiar. El significado de las posibles formas que puede tomar un LER es el siguiente:

Si un LER muestra este aspecto durante la simulación significa que ha su nivel de congestión está entre 0% y 50%.

Si un LER muestra este aspecto durante la simulación significa que ha su nivel de congestión está entre el 51% y el 75%.

Si un LER muestra este aspecto durante la simulación significa que ha su nivel de congestión está entre el 76% y el 95%.

Si un LER muestra este aspecto durante la simulación significa que ha su nivel de congestión está entre el 96% y el 100%. En este momento es más que probable que inminentemente el nodo comience a descartar paquetes por saturación de su búfer.

El cambio de un aspecto a otro se realiza de forma automática a medida que los paquetes se van acumulando en búfer el nodo.

3.3.5.5.7.1.3. Nodos LER activos

El nodo LER activo (LERA) se representa en la topología, y por tanto en la simulación, con el siguiente icono.

Pero durante la simulación, el nodo puede irse congestionando por lo que su aspecto puede cambiar. El significado de las posibles formas que puede tomar un LERA es el siguiente:

Si un LERA muestra este aspecto durante la simulación significa que ha su nivel de congestión está entre 0% y 50%.

Si un LERA muestra este aspecto durante la simulación significa que ha su nivel de congestión está entre el 51% y el 75%.

Si un LERA muestra este aspecto durante la simulación significa que ha su nivel de congestión está entre el 76% y el 95%.

Si un LERA muestra este aspecto durante la simulación significa que ha su nivel de congestión está entre el 96% y el 100%. En este momento es más que probable que inminentemente el nodo comience a descartar paquetes por saturación de su búfer.

El cambio de un aspecto a otro se realiza de forma automática a medida que los paquetes se van acumulando en búfer el nodo.

3.3.5.5.7.1.4. Nodos LSR

El nodo LSR se representa en la topología, y por tanto en la simulación, con el siguiente icono.

Pero durante la simulación, el nodo puede irse congestionando por lo que su aspecto puede cambiar. El significado de las posibles formas que puede tomar un LSR es el siguiente:

Si un LSR muestra este aspecto durante la simulación significa que ha su nivel de congestión está entre 0% y 50%.

Si un LSR muestra este aspecto durante la simulación significa que ha su nivel de congestión está entre el 51% y el 75%.

Si un LSR muestra este aspecto durante la simulación significa que ha su nivel de congestión está entre el 76% y el 95%.

Si un LSR muestra este aspecto durante la simulación significa que ha su nivel de congestión está entre el 96% y el 100%. En este momento es más que probable que inminentemente el nodo comience a descartar paquetes por saturación de su búfer.

El cambio de un aspecto a otro se realiza de forma automática a medida que los paquetes se van acumulando en búfer el nodo.

3.3.5.5.7.1.5. Nodos LSR activos

El nodo LSR activa (LSRA) se representa en la topología, y por tanto en la simulación, con el siguiente icono.

Pero durante la simulación, el nodo puede irse congestionando por lo que su aspecto puede cambiar. El significado de las posibles formas que puede tomar un LSRA es el siguiente:

Si un LSRA muestra este aspecto durante la simulación significa que ha su nivel de congestión está entre 0% y 50%.

Si un LSRA muestra este aspecto durante la simulación significa que ha su nivel de congestión está entre el 51% y el 75%.

Si un LSRA muestra este aspecto durante la simulación significa que ha su nivel de congestión está entre el 76% y el 95%.

Si un LSRA muestra este aspecto durante la simulación significa que ha su nivel de congestión está entre el 96% y el 100%. En este momento es más que probable que inminentemente el nodo comience a descartar paquetes por saturación de su búfer.

El cambio de un aspecto a otro se realiza de forma automática a medida que los paquetes se van acumulando en búfer el nodo.

3.3.5.5.7.1.6. Emisores

El nodo emisor se representa en la topología, y por tanto en la simulación, con el siguiente icono.

Durante la simulación, el nodo emisor no puede congestionarse puesto que no recibe datos así que su aspecto no puede cambiar. Un nodo emisor siempre mantendrá el mismo aspecto.

3.3.5.5.7.1.7. Receptores

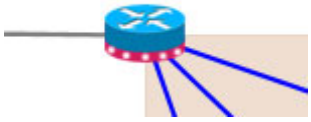
El nodo receptor se representa en la topología, y por tanto en la simulación, con el siguiente icono.

Durante la simulación, el nodo emisor podría congestionarse puesto que recibe datos; sin embargo en Open SimMPLS 1.0 el receptor se considera ideal, con un buffer ilimitado y

con una capacidad, ilimitada también, para aceptar paquetes. Así que su aspecto no puede cambiar. Un nodo emisor siempre mantendrá el mismo aspecto.

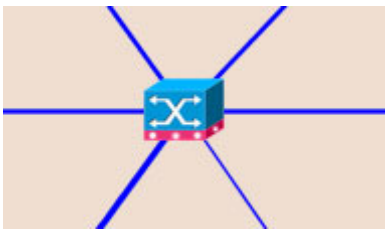
3.3.5.5.7.1.8. Enlaces

Un enlace se muestra en el área de simulación en forma de línea. El color y el grosor de la línea reflejan la configuración del enlace. Observemos la siguiente figura.



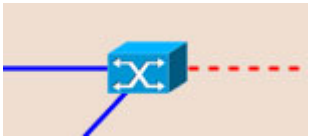
En ella se muestran cuatro enlaces; tres azules y uno gris. El color gris indica que el enlace es externo, que no forma parte del dominio MPLS y que une un LER/LERA, de entrada o salida, con un nodo emisor o con un nodo receptor. Es una ayuda visual para diferenciar claramente los límites del dominio MPLS. Los enlaces azules indican lo contrario; que son enlaces internos al dominio MPLS y por tanto nunca unirán en ninguno de sus extremos, un emisor o un receptor.

En cualquiera de los dos casos, el grosor indicará el retardo del enlace. Observemos ahora la siguiente figura.



Se ve claramente que no todos los enlaces tienen el mismo grosor. Cuanto más grueso es un enlace, esto significa que su retardo es mayor y que por tanto los paquetes fluirán a través de él con mayor rapidez. En el extremo opuesto, un enlace que se muestre en la topología muy fino indicará que tiene un gran retardo y que los paquetes tardarán más en llegar al destino si circulan por él.

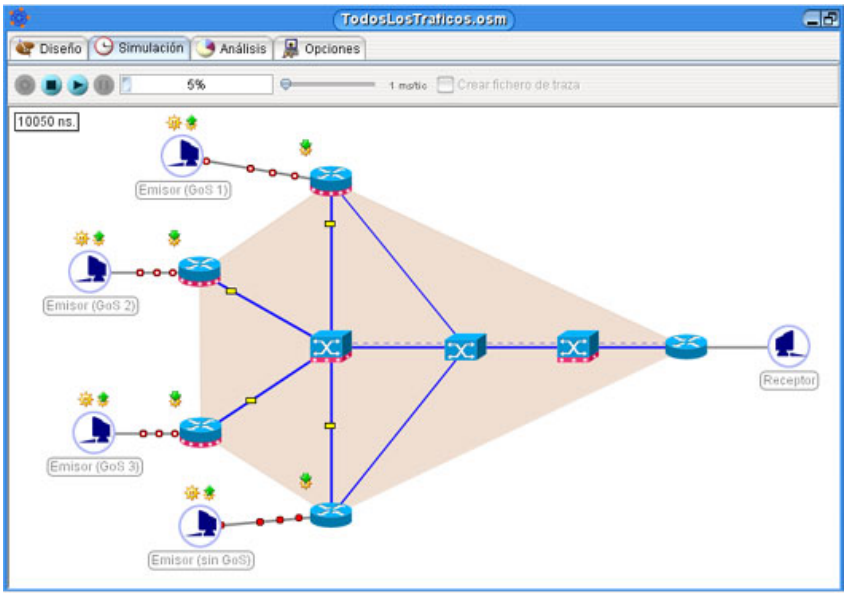
Para finalizar, el enlace se puede mostrar como una línea discontinua de grosor fijo, como se puede ver en la siguiente imagen.



En este caso el color rojo y la línea discontinua indican que el enlace está caído. Para todos los efectos en la simulación, será como si el enlace no existiera. Representa errores del enlace como por ejemplo una rotura o una desconexión. Nunca se mostrará automáticamente un enlace de esta forma; por defecto los enlaces no caen solos; se tendrá que interactuar manualmente con la simulación para caer un enlace; pero esto se verá más adelante.

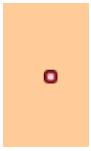
3.3.5.5.7.2. Paquetes

Los paquetes de tráfico son, posiblemente, la parte más importante de la simulación, pues nos permiten saber que tipos de tráfico aparecen en el escenario, que cantidad de cada uno de ellos, cuándo se produce la señalización y cómo. Por donde circulan los paquetes y a qué velocidad, etcétera. La siguiente imagen muestra un escenario típico con distintas clases de tráfico.



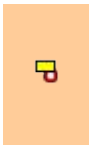
Una vez acostumbrados a la representación visual de cada tipo de tráfico, el análisis de lo que está ocurriendo en la simulación resulta fácil. Los tipos de paquetes que pueden circular por la red simulada en un momento dado son los siguientes:

Paquete IP, sin ninguna marca de Garantía de Servicio ni requerimientos de LSP de respaldo.



Paquete IP, con alguna marca de Garantía de Servicio o requerimientos de LSP de respaldo (o ambas cosas).

Paquete MPLS, sin ninguna marca de Garantía de Servicio ni requerimientos de LSP de respaldo.

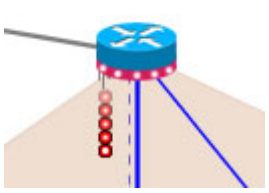


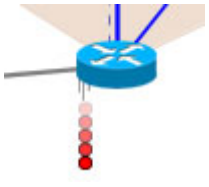
Paquete MPLS, con alguna marca de Garantía de Servicio o requerimientos de LSP de respaldo (o ambas cosas).

Paquete TLDP, usado para la señalización de los LSP por los que debe circular el tráfico MPLS.

Paquete GPSRP, utilizado para la solicitud de retransmisión de paquetes GoS descartados. Sus posibles respuestas se representan también así.

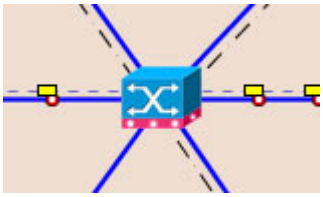
Los paquetes además de circular por la red que se esté simulando, pueden ser descartados en los nodos. En ese caso los paquetes aparecerán, literalmente, como cayendo de dicho nodo. En las siguientes figuras se muestran algunos ejemplos de ello:





3.3.5.5.7.3. Label Switched Paths

Los Label Switched Paths son los caminos lógicos establecidos vía señalización por TLDP para que un flujo entrante en el dominio MPLS siga una ruta fija hasta su salida del mismo, lo más rápidamente posible. Open SimMPLS 1.0 implementa dos tipos de LSP, el LSP tradicional de cualquier arquitectura MPLS, y caminos de respaldo propios de esta propuesta, iniciados mediante la tecnología RLPRP. Ambos tienen su representación característica dentro del área de simulación, como observamos en la siguiente imagen.



Los LSP tradicionales se muestran en la simulación con una línea fina azul y discontinua de tipo raya-raya, paralela al enlace sobre el cual se ha establecido el LSP. Los LSP de respaldo se muestran con una línea más gruesa, negra y discontinua de tipo raya-punto-raya también paralela al enlace sobre el cual se ha establecido el LSP de respaldo, pero en el otro lado del mismo, no en el mismo lado donde se muestran los LSP tradicionales.

3.3.5.5.8. Interactuando con la simulación

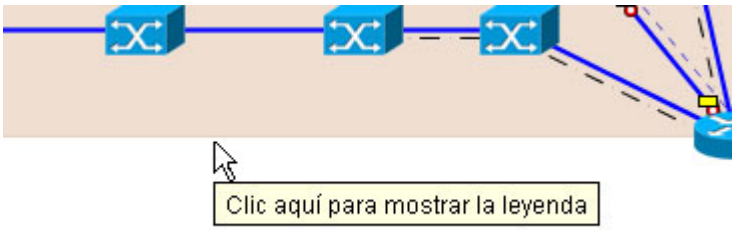
Con lo que se ha visto anteriormente hemos completado el conjunto de señales visuales que se deben poder interpretar en una simulación para saber qué está ocurriendo. Sin embargo, hay muchas acciones que se pueden llevar a cabo mientras la simulación está en funcionamiento, ejecutándose; las veremos a continuación.

3.3.5.5.8.1. Mostrar y ocultar la leyenda

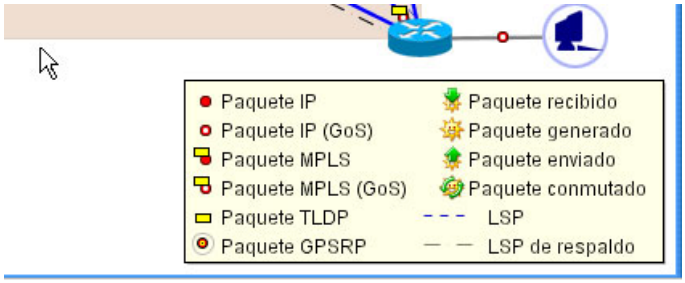
Todo lo explicado párrafos atrás sobre la representación de los paquetes y los LSP, puede ser consultado en tiempo real mientras que la simulación está en funcionamiento mediante el mostrado u ocultación de una leyenda.

Para visualizar la leyenda explicativa se ha de estar en el área de simulación. Si se deja el puntero del ratón quieto sobre un lugar de dicho área en el que no hay ningún elemento (o

sea, cuando el puntero es una flecha y no una mano), aparece un recuadro que nos india lo que podemos hacer.



Así pues, haciendo clic con el botón principal del ratón en un lugar donde no haya elementos, conseguiremos que aparezca la leyenda.



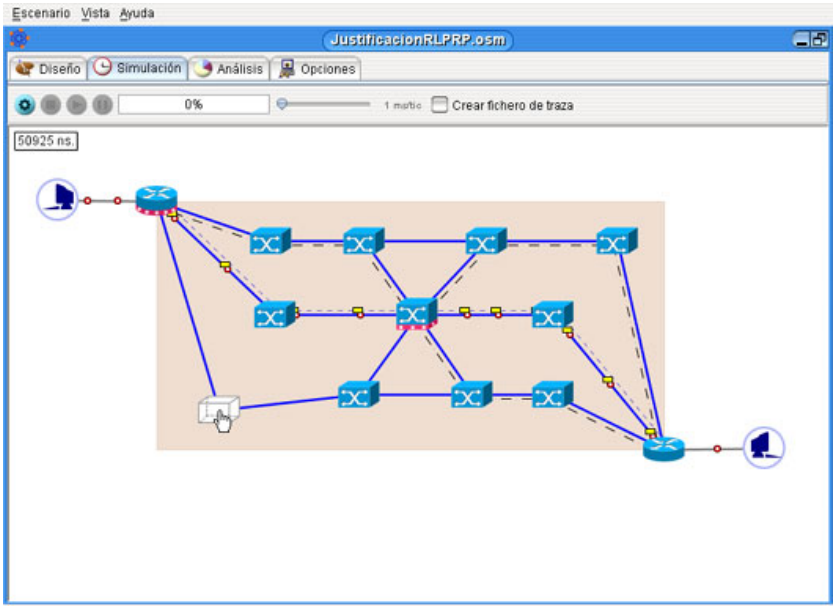
Que siempre aparece en la esquina inferior izquierda del área de simulación y se superpone, si es necesario, sobre cualquier elemento de la simulación, paquetes, etcétera..

Para dejar visualizar la leyenda explicativa se ha de estar en el área de simulación. Si se deja el puntero del ratón quieto sobre un lugar de dicho área en el que no hay ningún elemento (o sea, cuando el puntero es una flecha y no una mano), aparece un recuadro que nos india lo que podemos hacer.

Así pues, haciendo clic con el botón principal del ratón en un lugar donde no haya elementos, conseguiremos que en esta ocasión desaparezca la leyenda.

3.3.5.5.8.2. Cambiar la distribución de los elementos

Si desea variar de posición los elementos de la topología durante la simulación, ha de estar en la pantalla de diseño. Una vez allí, simplemente sitúese sobre un nodo cualquiera hasta que el cursor del ratón aparezca con forma de mano, indicando que está sobre un objeto. A continuación pulse el botón izquierdo del ratón y sin soltarlo desplace el ratón.



El nodo aparecerá en blanco, con forma de armazón, y se desplazará a medida que usted mueva el ratón por el área de diseño. Los enlaces que partan o lleguen de ese nodo, se moverán también con él. Una vez que el nodo esté situado en el lugar que usted desee, suelte el botón izquierdo del ratón y habrá movido a su gusto el nodo desde una a otra posición. Esta operación puede ser realizada incluso con la simulación en fase de ejecución; los paquetes de los enlaces seguirán en ellos aunque los mueva.

Sólo los nodos pueden ser movidos. Para mover un enlace debe mover los nodos que une dicho enlace. Un enlace por sí solo no puede ser movido.

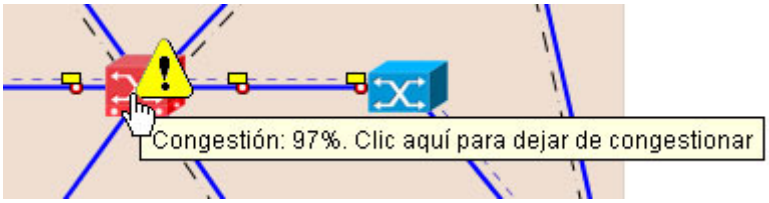
3.3.5.5.8.3. Congestionar, descongestionar y obtener congestión.

Los nodos pueden congestionarse. Se puede obtener su nivel de congestión (porcentaje de llenado del buffer); para ello, estando en el área de simulación, simplemente sitúe el puntero del ratón sobre el nodo cuyo nivel de congestión desee conocer. Aparecerá entonces un recuadro con la información requerida.

Como se aprecia en la figura, además de mostrar el nivel de congestión, se informa de que se puede congestionar artificialmente al nodo haciendo clic en ese momento con el botón principal del ratón. Consultar el nivel de congestión es una operación que se puede realizar independientemente de la situación de la simulación (parada, en curso…). Sin embargo, saturar artificialmente un nodo es una acción que se debe hacer sólo cuando la simulación esta en curso, ejecutándose. Si hacemos clic, ocurre lo siguiente.

El nodo se sitúa automáticamente con un 97% de congestión en los búferes. Por tanto muestra el aspecto de un nodo saturado más de 95%, como se explico páginas atrás. A partir de ese momento es cuestión de poco tiempo que el nodo comience a descartar paquetes si el tráfico que le llega lo permite. Se suele usar esta acción para estudiar las pérdidas y recuperaciones de paquetes sin tener que esperar un tiempo determinado a que se congestione el nodo por medios naturales.

Ahora nos situamos sobre el nodo, para comprobar el nivel de congestión que tiene. Lo que aparece se muestra en la siguiente figura.



Nos muestra que el nodo tiene un nivel de congestión del 97% (que irá subiendo si sigue recibiendo tráfico) y además no da la oportunidad de dejar de congestionarlo artificialmente. No es necesario hacerlo, pero nos puede interesar. En ese caso, simplemente volvemos a hacer clic sobre el nodo con el botón principal del ratón y el nodo dejará el estado de congestión artificial. El nivel de congestión del nodo quedará con el valor que tenía inicialmente antes de congestionarlo artificialmente más el porcentaje de congestión que haya ido aumentando desde que se congestionó artificialmente.

Por ejemplo:

1. Congestión antes de saturar artificialmente = 12% (De forma natural).
2. Congestión tras saturar artificialmente = 97% (Se ha añadido un 85% artificialmente).
3. Al 99% dejamos de saturar artificialmente (Ha aumentado un 2% de forma natural).
4. Si descongestionamos, el nodo queda con un 99-85 = 14%.

3.3.5.5.8.4. Caída y recuperación de enlaces.

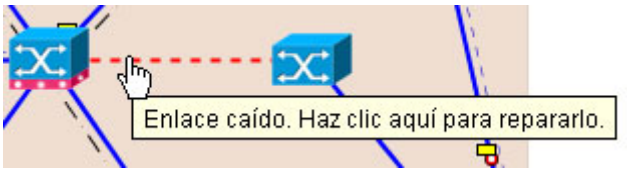
En condiciones reales, un enlace de red está sujeto a la posibilidad de fallos; obras, descargas eléctricas, fallos humanos, etcétera, pueden hacer que un enlace falle y el tráfico se pierda. Sin embargo un enlace no falla por si solo aunque soporte mucho tráfico. Open SimMPLS 1.0 permite simular este hecho, permite que un enlace pueda fallar en un momento dado, pero al igual que en la realidad, tampoco es algo que ocurra como evolución de la simulación sino que se ha de manipular manualmente.

Si desea obligar a que un enlace falle, debe encontrarse en el área de simulación y la simulación debe estar en funcionamiento. Sitúe el puntero del ratón sobre el enlace que desee que falle y espere un segundo. Aparecerá un mensaje indicativo como se muestra en la siguiente figura.

El mensaje nos indica, en este caso, que el enlace está funcionando perfectamente; y nos da la posibilidad de hacer que falle simplemente haciendo un clic con el botón principal del ratón en dicho enlace. Si lo hacemos, el enlace cae.

El momento de la caída se muestra visualmente mediante una "explosión". Tras dicha explosión el enlace permanecerá en estado de fallo constante, se mostrará con una línea roja discontinua y todos los paquetes circulantes serán descartados. Si el enlace soportaba LSP o LSP de respaldo, comenzará la señalización para reestructurar dichos LSP bien de la forma tradicional MPLS como de la forma de nuestra propuesta, mediante RLPRP. Dicha señalización no tiene vuelta atrás aunque volvamos a activar el enlace.

Si queremos activar el enlace de nuevo, hemos de repetir la operación. No situamos con el puntero del ratón sobre el enlace que deseamos activar (debe estar caído, claro) y aparece el siguiente mensaje:

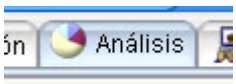


Que nos indica que el enlace está caído y nos ofrece la posibilidad de repararlo haciendo, de nuevo, un clic sobre él. Si lo hacemos, ocurre lo que se muestra en la siguiente figura.

El enlace vuelve a estar activo y el momento de dicha activación se muestra visualmente con unas estrellas que aparecen en el enlace. Sin embargo, aunque el enlace vuelva a estar activo, todos los LSP que pudiese haber establecidos sobre él así como el tráfico que pudiese circular, se ha perdido. En este momento la señalización que comenzó con la caída del enlace, habrá calculado un nuevo LSP para redirigir el tráfico.

3.3.5.6. Área de análisis

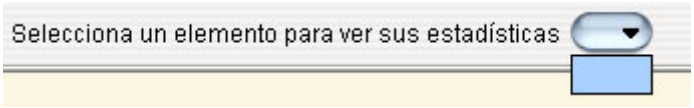
Debemos seleccionar trabajar en el área de análisis cuando hemos configurado en la topología algunos elementos para que generen estadísticas y queremos observar las gráficas que se van generando (o están ya generadas, si la simulación ha terminado). Para seleccionar este modo de trabajo, se debe hacer clic sobre la pestaña **"Análisis"** de la ventana de escenario.



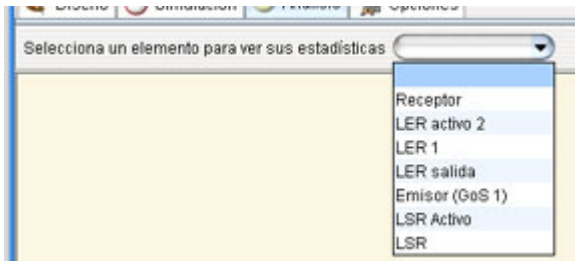
El área de análisis es de fondo amarillo y se divide, de nuevo, en dos subpartes: una barra de herramientas de análisis y un área más grande (la de color amarillo) donde se mostrarán las estadísticas del elemento seleccionado.

3.3.5.6.1. Selección del elemento a mostrar

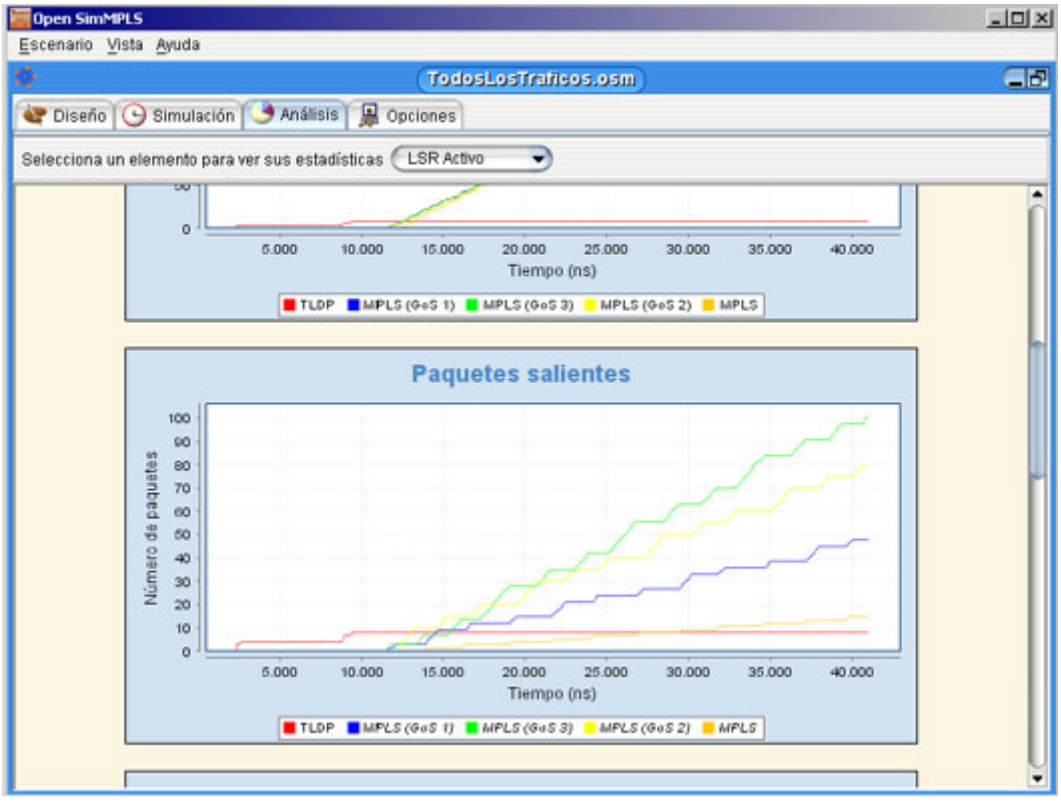
Para seleccionar el elemento del cual queremos mostrar las gráficas debemos hacer clic con el botón principal del ratón en la lista desplegable que se encuentra en la barra de herramientas de análisis y que se llama **"Seleccione un elemento para ver sus estadísticas"**. La lista mostrará todos los elementos que actualmente están generando o han generado estadísticas. Si no aparece ningún elemento, como se puede observar en la siguiente figura, es porque no se ha configurado ningún nodo para generar estadísticas o porque la simulación no ha comenzado aún.



Si por el contrario hay elementos configurados para mostrar estadísticas, aparecerán en esta lista desplegable, como muestra la siguiente imagen.



Para ver las gráficas estadísticas de cualquiera de ellos, hay que hacer clic con el botón principal del ratón en el que interese. Entonces se mostrarán en el área amarilla las gráficas correspondientes a dicho nodo. Si la simulación está en curso, las gráficas tendrán un comportamiento dinámico, variando según va evolucionando la simulación. Si no, permanecerán inamovibles.



Además de las gráficas, en el área amarilla se muestran datos del escenario: el título, el autor y la descripción. Ya veremos que estos datos opcionales se configuran en otro lugar. La siguiente imagen muestra estos datos en un caso real.

No todos los nodos proporcionan el mismo número de gráficas, ni del mismo tipo, ni presentan en ellas los mismos datos. Pero no es una opción parametrizable. En la versión 1.0 de Open SimMPLS el número y tipo de gráficas y los datos representados para cada tipo de nodo es fijo.

Los datos representados para cada tipo de nodo del escenario son:

Nodo Emisor:

Gráfica 1: (gráfica de líneas) representa el número de paquetes salientes.

Nodo LER/LSR:

Grafica 1: (gráfica de líneas) representa el número de paquetes entrantes.

Grafica 2: (gráfica de líneas) representa el número de paquetes salientes.

Grafica 3: (gráfica de líneas) representa el número de paquetes descartados.

Nodo LERA/LSRA:

Grafica 1: (gráfica de líneas) representa el número de paquetes entrantes.

Grafica 2: (gráfica de líneas) representa el número de paquetes salientes.

Grafica 3: (gráfica de líneas) representa el número de paquetes descartados.

Grafica 4: (gráfica de barras) representa el número de recuperaciones servidas.

Grafica 5: (gráfica de barras) representa el número de recuperaciones requeridas.

Nodo Receptor:

Gráfica 1: (gráfica de líneas) representa el número de paquetes entrantes.

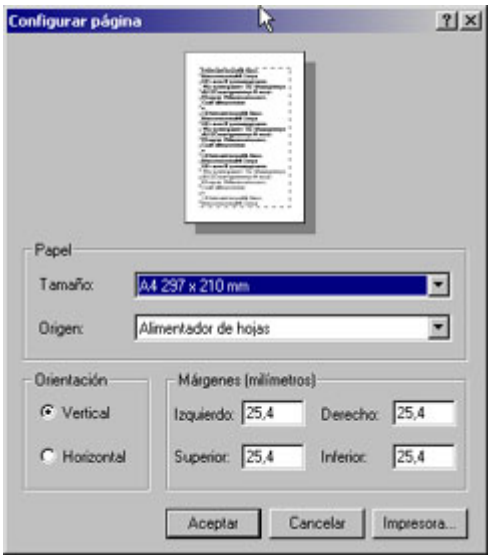
3.3.5.6.2. Manejando las gráficas

Las gráficas generadas por Open SimMPLS 1.0 para cada elemento no son meras imágenes sino que son objetos interactivos. Se puede obtener un menú emergente con opciones sobre cada una de las gráficas simplemente haciendo clic sobre ellas con el botón secundario del ratón, como muestra las siguientes figuras.

De entre todas las posibles opciones, que son bastantes, comentaremos en las siguientes líneas las más importantes.

3.3.5.6.2.1. Imprimir las gráficas

Toda gráfica en el simulador se puede imprimir directamente. Para ello hay que desplegar el menú emergente como se explica en el punto anterior y seleccionar la opción **"Imprimir"**; para ello hay que hacer clic sobre esta opción con el botón principal del ratón. Aparecerá entonces una ventana donde se nos permite configurar la página donde vamos a imprimirla y, eventualmente, la impresora a utilizar.

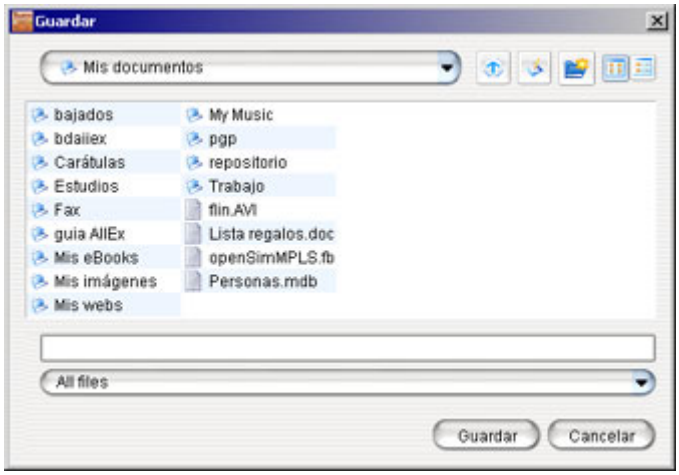


Si en este momento ha cambiado de impresión y no desea enviar la gráfica a la impresora, haga clic con el botón principal del ratón en el botón **"Cancelar"** y volverá al área de análisis como si nunca hubiese intentado imprimir nada. Si por el contrario desea seguir adelante, seleccione los valores que desee para el papel y los márgenes y pulse **"Aceptar"**. La ventana de configuración del papel se cerrará y si todo va bien, la gráfica se imprimirá en la impresora seleccionada.

3.3.5.6.2.2. Almacenar las gráficas

Una de las opciones más importantes es quizás la opción de exportar las gráficas a un fichero gráfico. Normalmente se requieren las gráficas para escribir artículos, informes… donde se detallen las pruebas que se han realizado y sus resultados. En este caso disponer de un fichero gráfico con las gráficas estadísticas es muy cómodo; más que tener realizar capturas de pantalla para poder obtenerlas.

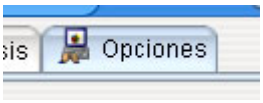
Para guardar como una imagen la gráfica deseada, abra el menú emergente como se explica dos puntos atrás y seleccione la opción **"Guardar como"**; para ello hay que hacer clic sobre esta opción con el botón principal del ratón. Aparecerá entonces una ventana donde se nos permite seleccionar el lugar donde queremos almacenar la gráfica y el nombre que deseamos para el fichero.



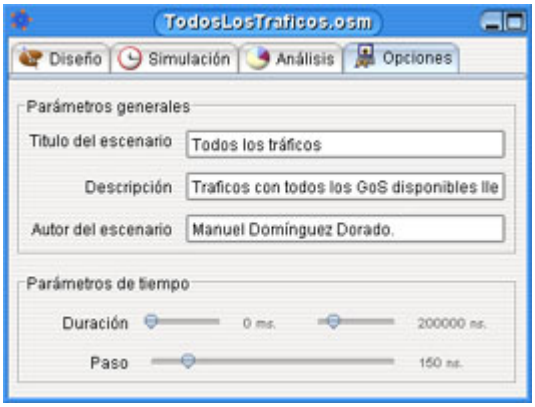
Si en este momento cambia de idea, pulse **"Cancelar"** y volverá al área de análisis como si nunca hubiese comenzado esta operación. Si desea seguir adelante, seleccione un lugar y un nombre para la gráfica y pulse **"Aceptar"** con el botón principal del ratón. La ventana de diálogo se cerrará y la gráfica quedará almacenada en un fichero con formato PNG en el lugar seleccionado. Podrá acceder a ella y abrirla o manejarla con cualquier visor de imágenes.

3.3.5.7. Opciones generales del escenario

Hay dos aspectos generales sobre el escenario a configurar los datos del escenario y los parámetros temporales. Para configurar dichos parámetros ha de estar situado en la pestaña de opciones, lo que conseguirá si hace clic con el botón principal del ratón en la pestaña opciones de la ventana de escenario.



Entonces aparecerá el área de opciones principales que presenta un aspecto como el que se muestra en la siguiente imagen.



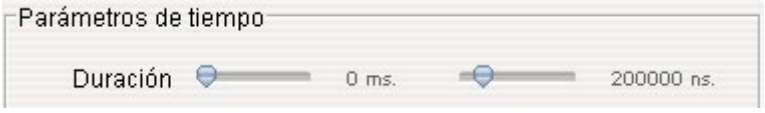
3.3.5.7.1. Datos del escenario

Los datos del escenario tienen carácter meramente informativo. Si va a compartir por correo electrónico el escenario, o si va a dejarlos públicos en Internet, por ejemplo, sería conveniente que los datos de autoría, la descripción y el título del escenario puedan acompañar al fichero; Estos datos se configuran aquí.

En el recuadro **"Título del escenario"** debería escribir un nombre descriptivo pero breve del escenario, por ejemplo "Troncal Madrid-Sevilla". En el recuadro **"Descripción"** debería comentar en poco espacio (unas decenas de palabras) lo importante del escenario como por ejemplo "Hay cuatro flujos, todos confluyen en el nodo X que acaba congestionado". Por último, en el cuadro **"Autor del escenario"**, debería introducir su nombre, si es usted quien ha realizado el escenario.

3.3.5.7.2. Parámetros temporales

Hasta ahora en ningún lugar hemos hablado de la duración de la simulación. La simulación no es infinita, no al menos en este simulador; así que hay que establecer un límite. En Open SimMPLS se hace mediante la suma de dos valores que se pueden modificar mediante dos deslizadores, como se ve en la siguiente figura.



La duración final de la simulación será la suma de ambos valores, pasados a nanosegundos. El primero de los deslizadores permite establecer, mediante grano grueso, el número de milisegundos; el segundo, permite con grano fino ajustar de forma más precisa, en nanosegundos. Entre ambos deslizadores siempre deben sumar al menos un nanosegundo de simulación y como máximo 2.999.999 nanosegundos (casi 3 milisegundos). No se permiten simulaciones más largas en este simulador.

La forma de variar el valor de ambos selectores y seleccionar el que deseamos es moviendo el deslizador deseado de izquierda a derecha o bien colocando el foco sobre el deslizador y usar las teclas de izquierda y derecha del teclado. Así habremos establecido cuántos nanosegundos queremos que dure la simulación.

El siguiente valor a configurar es el paso o tic de simulación. Cada tic de reloj se muestra el estado de la simulación en el área de simulación; esto significa que con tics menores, la simulación será mucho más lenta pero a cambio mucho más realista, mostrando todos los detalles. Por otro lado, tics mayores permiten una mayor rapidez a costa aumentar la granularidad de lo representado en el área de simulación y por tanto, de perder precisión visual en la simulación. Este valor se configura en el deslizador **"Paso"**.

La forma de variar el valor y seleccionar el que deseamos es moviendo el deslizador deseado de izquierda a derecha o bien colocando el foco sobre el deslizador y usar las teclas de izquierda y derecha del teclado. El rango permitido será, como mínimo 1 nanosegundo y como máximo, el valor dado por la siguiente función:

$$MáximoPaso = \min\{MinimoDelayDeLosEnlaces, DuracionDeLaSimulacion\}$$

Aunque no se deberá preocupar por ello porque en cualquier caso este mínimo se ajusta automáticamente y el deslizador nunca le permitirá seleccionar un valor menor.

3.3.5.8. Manejando escenarios

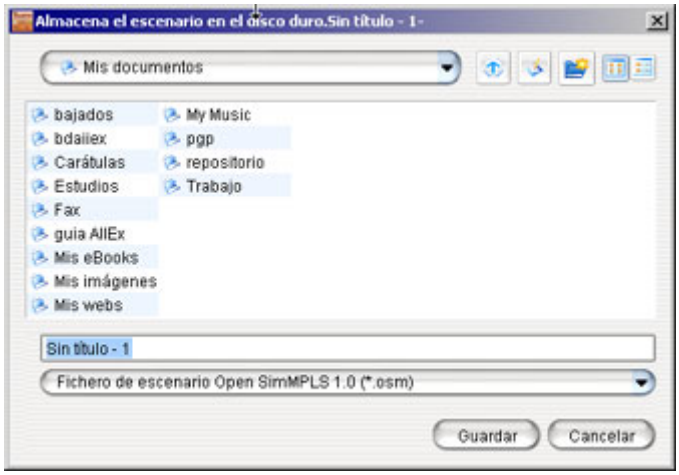
Open SimMPLS 1.0 es multiescenario. Aunque realmente el rendimiento del simulador cae cuando se intentan simular más de una red a la vez, esto depende mucho de las características del equipo donde se ejecute el simulador. De todas formas siempre se puede tener más de un escenario abierto, diseñarlos simultáneamente, etcétera. Y Open SimMPLS 1.0 ofrece acciones para manejarse entre todos los escenarios abiertos.

3.3.5.8.1. Guardando escenarios

Cuando hay varios escenarios abiertos, la operación de guardar un escenario se aplica sobre aquel que está seleccionado (activo) en ese momento. Si sólo hay un escenario, ese será el escenario activo.

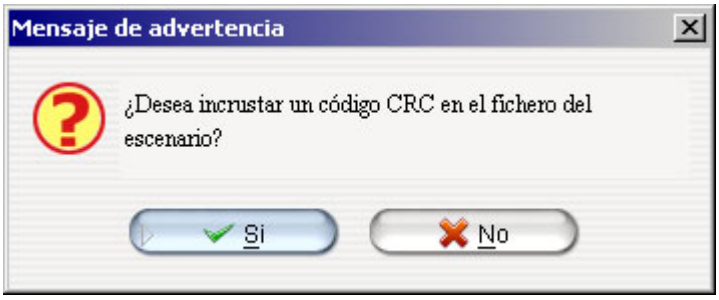
Para guardar un escenario, hay que acudir al menú principal de la aplicación y hacer clic sobre la opción **"Escenario"**. Entonces se desplegará el menú y nos permitirá elegir la opción **"Guardar"** que es la que nos interesa.

Si es la primera vez que vamos a almacenar el escenario, es decir, si no existe más que en el simulador y no hay copia de él en el disco, el simulador muestra una ventana para seleccionar el lugar donde queremos almacenar el escenario y el nombre que le queremos dar.



Si en este momento decide no almacenar el escenario, haga clic sobre el botón **"Cancelar"**; la ventana de la figura se cerrará y volverá al simulador tal y como estaba antes de intentar guardar el escenario. Si por el contrario desea almacenarlo, seleccione un directorio/carpeta para el escenario y teclee un nombre en el lugar correspondiente. Finalmente pulse **"Aceptar"**.

Esta ventana de diálogo para la selección del nombre del escenario y lugar del almacenaje, no aparecerá si el escenario ya ha sido guardado con antelación. De uno u otro modo, llegados a este punto por cualquiera de los caminos, Open SimMPLS 1.0 mostrará la siguiente ventana.



Donde nos pregunta si queremos insertar un CRC en el escenario. Un CRC es un "Código de Redundancia Cíclica", un código que insertado dentro del fichero del enlace permitirá con posterioridad averiguar si el escenario ha sido modificado manualmente. Es un medio de seguridad.

Antes de responder a esta pregunta, tenga en cuenta las siguientes observaciones.

- No es obligatorio insertar un código CRC en el escenario.
- Open SimMPLS abrirá cualquier escenario que no contenga CRC.
- Open SimMPLS abrirá cualquier escenario que contenga CRC y este demuestre que el escenario no ha sido modificado.
- Open SimMPLS **NUNCA** abrirá ningún fichero que contenga CRC y que permita demostrar que el fichero se ha modificado manualmente.

¡Nota! Bajo ciertas circunstancias el código CRC puede dar como erróneo un fichero de escenario que no lo sea, por ejemplo si el fichero se crea en un PC en español, con caracteres españoles como la eñe, los acentos, etcétera y se intenta abrir en un PC con idioma inglés, que no soporte esos caracteres. En este caso el fichero aparecerá como erróneo. Use el código CRC bajo su propio criterio.

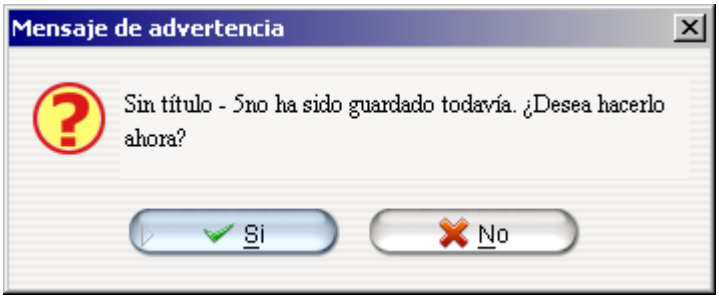
Si tras leer estas observaciones decide insertar un código CRC en el fichero de escenario, pulse **"Aceptar"**. Si no, pulse **"Cancelar"**. En ambos casos, la ventana de diálogo desaparecerá y el escenario quedará almacenado en disco; en un caso con CRC y en otro caso sin él.

3.3.5.8.2. Cerrando escenarios

Cuando hay varios escenarios abiertos, la operación de cerrar un escenario se aplica sobre aquel que está seleccionado (activo) en ese momento. Si sólo hay un escenario, ese será el escenario activo.

Para cerrar un escenario, hay que acudir al menú principal de la aplicación y hacer clic sobre la opción **"Escenario"**. Entonces se desplegará el menú y nos permitirá elegir la opción **"Cerrar"** que es la que nos interesa. También se puede invocar esta acción mediante la combinación de teclas **"Control + Z"** desde cualquier parte de la ventana principal del simulador.

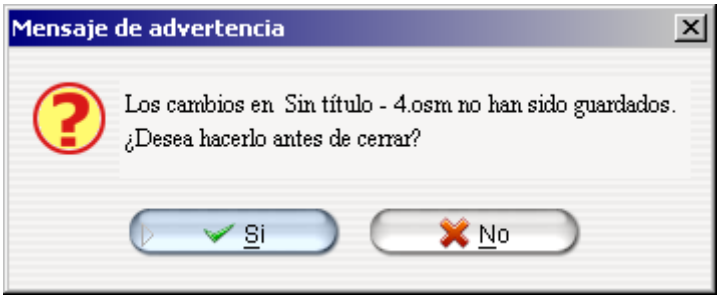
Si el escenario nunca se ha guardado, es decir, si no existe más que en el simulador y no hay copia de él en el disco, el simulador mostrará muestra la siguiente ventana.



Que indica que se va a cerrar el escenario llamado “Sin título - 5” (en este caso del ejemplo) y no se había guardado. Nos da entonces la opción de guardarlo antes de cerrarlo o simplemente olvidarnos del escenario y cerrarlo sin guardarlo. Si desea guardarlo antes de cerrar, debe hacer clic con el botón principal del ratón sobre el botón **“Si”**, y entonces comenzará el proceso de guardado que se ha descrito en el punto anterior. Si no se desea guardar el escenario, debe pulsar sobre el botón **“No”**.

En ambos casos el escenario se cerrará y desparecerla del simulador. En el primer caso se habrá almacenado antes en el disco y en el segundo caso el escenario se habrá perdido.

En el caso de que el escenario ya se haya guardado con antelación, pero una vez abierto se haya modificado, en el momento de seleccionar la opción **“Cerrar”** el simulador mostrará la siguiente ventana:



Que indica algo parecido al caso anterior. En este caso el fichero ya está almacenado en disco, pero no está actualizado con respecto al escenario que está abierto en memoria, en el simulador. Nos pregunta si queremos cerrar y obviar los últimos cambios que se han realizado con respecto al fichero del disco o si por el contrario queremos actualizar el fichero del disco con los últimos cambios antes de cerrar el escenario.

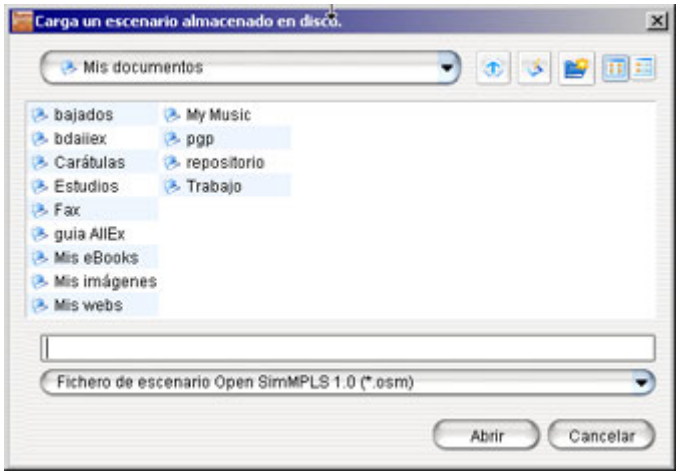
Si quiere olvidar los últimos cambios realizados en el escenario, debe pulsar sobre el botón **"No"**. En ese momento la ventana se cerrará, el escenario desaparecerá y la versión del escenario que existía en el disco quedará intacta. Si desea almacenar los cambios, entonces tiene que hacer clic con el ratón en el botón **"No"**. La ventana desaparecer e internamente se almacenarán los cambios en el disco antes de cerrar el escenario. No se nos pedirá nombre ni lugar para almacenar el escenario porque como ya ha sido almacenado con anterioridad, estos datos se conocen. Aunque si se nos preguntará si deseamos insertar un código CRC en el fichero del escenario. En el punto anterior se explica bien este concepto.

Si el escenario que está abierto está almacenado previamente en el disco y además no se ha modificado con respecto a éste (no hay cambios que poder perder), no se preguntará nada y el escenario se cerrará sin más.

3.3.5.8.3. Abriendo escenarios

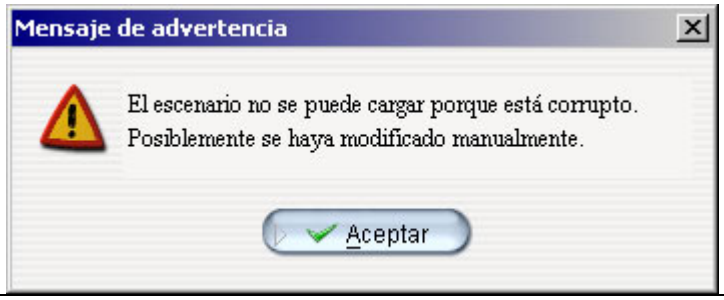
Para abrir un escenario previamente creado y guardado, hay que acudir al menú principal de la aplicación y hacer clic sobre la opción **"Escenario"**. Entonces se desplegará el menú y nos permitirá elegir la opción **"Abrir"** que es la que nos interesa.

Una vez que haya seleccionado la opción **"Abrir"**, se mostrará una ventana donde podemos buscar a través del sistema de ficheros, el escenario que se desea abrir.



Si en algún momento decide no abrir finalmente un escenario existente, haga clic con el botón principal del ratón sobre el botón **"Cancelar"**. La ventana se cerrará y volverá a la ventana principal de Open SimMPLS 1.0 como si no hubiese nunca intentado abrir un escenario. Si por el contrario está decidido, seleccione el escenario deseado y haga clic sobre el botón **"Aceptar"**. Tras esto, la ventana donde eligió el fichero, desaparecerá.

Puede entonces aparecerle una ventana de error como la que se muestra en la siguiente figura.

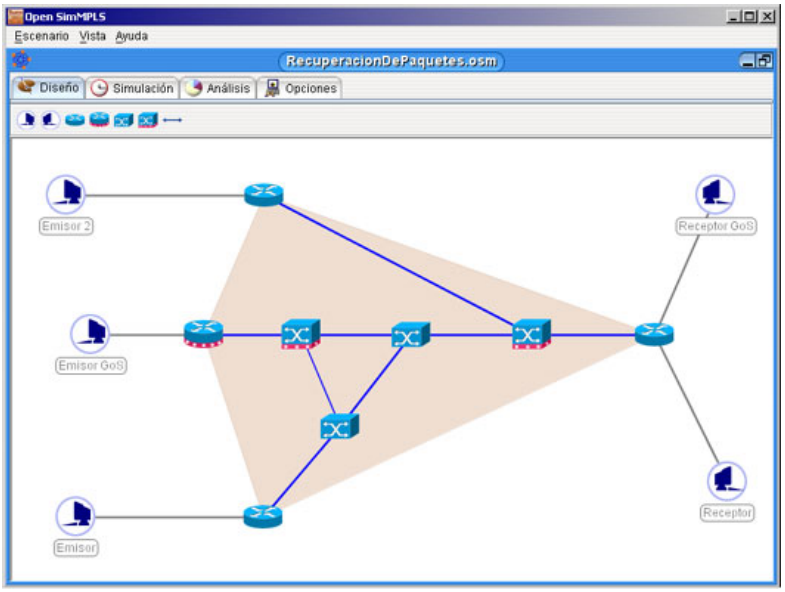


Si aparece este mensaje es porque el escenario incorpora un CRC de seguridad y, en base a él, se han demostrado que ha habido modificaciones manuales en el fichero, con lo cual no es posible asegurar que el fichero esté bien formado.

> *¡Nota! Bajo ciertas circunstancias el código CRC puede dar como erróneo un fichero de escenario que no lo sea, por ejemplo si el fichero se crea en un PC en español, con caracteres españoles como la eñe, los acentos, etcétera y se intenta abrir en un PC con idioma inglés, que no soporte esos caracteres. En este caso el fichero aparecerá como erróneo. Use el código CRC bajo su propio criterio.*

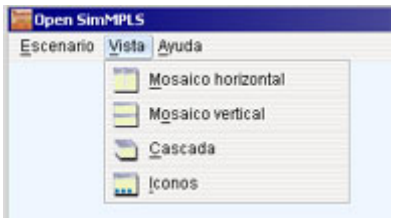
En este caso, el escenario no se abrirá. Haga clic sobre el botón **"Aceptar"**. La ventana de error desaparecer y volverá a la pantalla principal de la aplicación, que permanecerá como estaba.

Sin embargo, si esta ventana no aparece es porque todo va correctamente. Se encontrará en la ventana principal del simulador donde se habrá abierto una nueva ventana de escenario que contendrá el escenario que deseaba abrir. Ahora puede trabajar con él sin problemas.

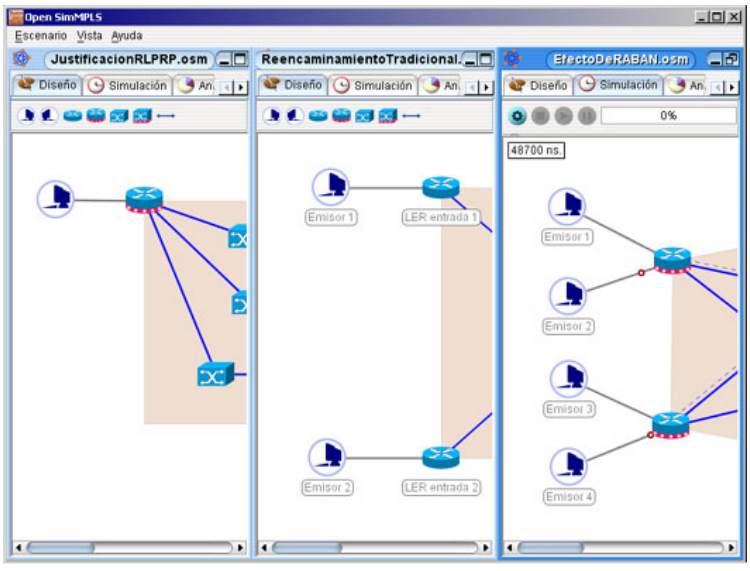


3.3.5.8.4. Disposición de los escenarios

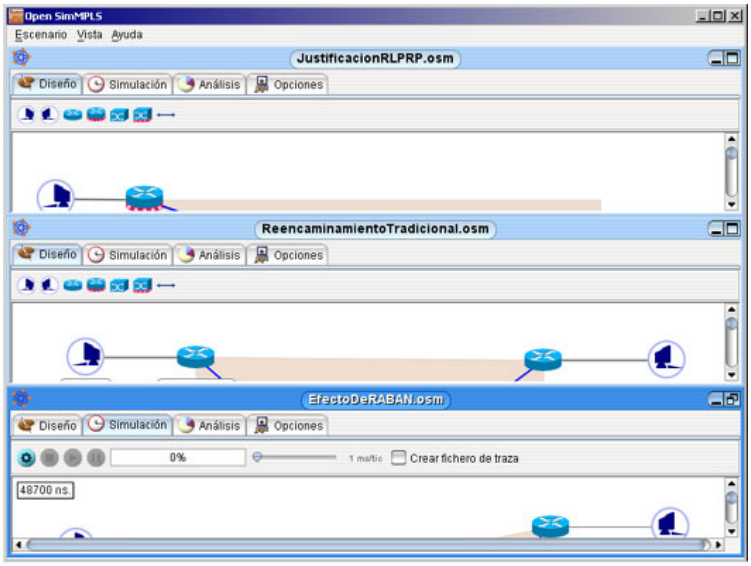
Cuando se trabaja con varios escenarios sobreviene la necesidad de poder seleccionar en cada momento el que se quiere usar. Pero si uno de los escenarios está maximizado, tedioso el poder acceder al resto. Open SimMPLS 1.0 ofrece la posibilidad de redistribuir todas las ventanas de escenarios en el área de trabajo de forma automática. Para ello es necesario que haya más de un escenario abierto. Hay que dirigirse al menú principal de la aplicación y hacer clic con el botón principal del ratón en la opción **"Vista"**. Se despliega un menú en el que podremos seleccionar cuatro opciones distintas.



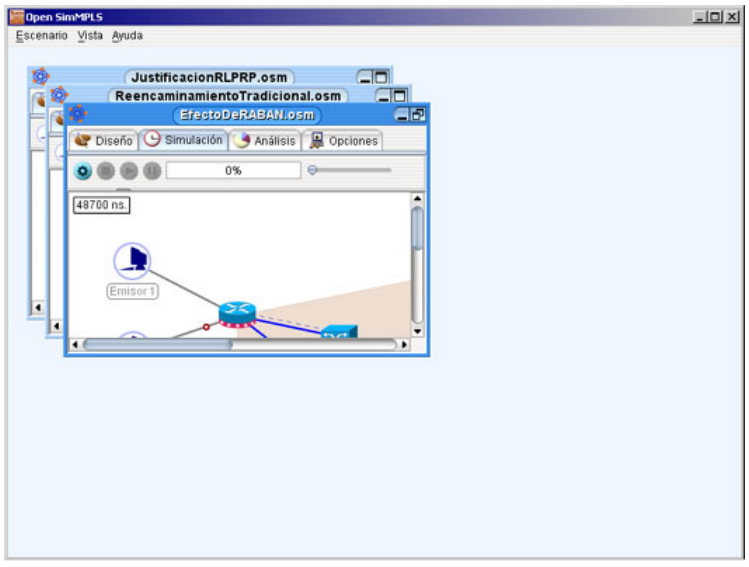
Para elegir la primera de ellas, llamada **"Mosaico horizontal"**, ha de hacer clic sobre dicha opción en el menú desplegado. Esta acción hará que los escenarios abiertos actualmente se distribuyan el tamaño del área de trabajo horizontalmente, ocupando la mayor extensión posible y quedando accesibles todos ellos, como se muestra en la siguiente figura.



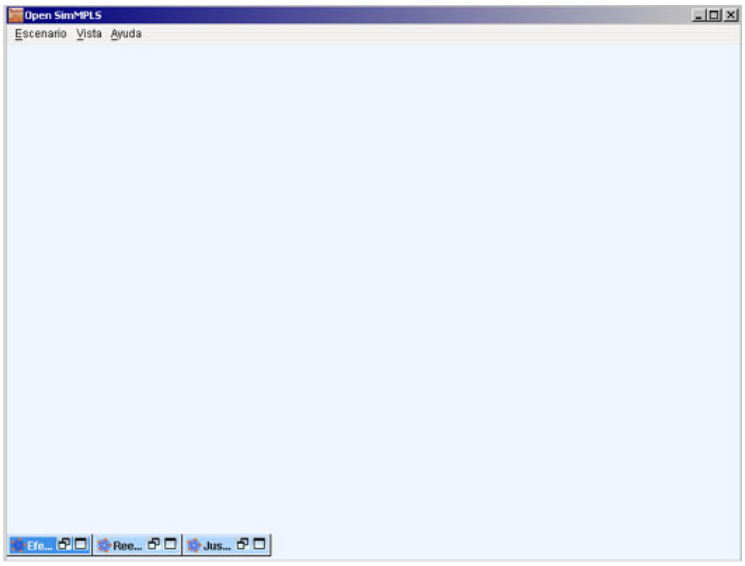
Para elegir la segunda de ellas, llamada **"Mosaico vertical"**, ha de hacer clic sobre dicha opción en el menú desplegado. Esta acción hará que los escenarios abiertos actualmente se distribuyan el tamaño del área de trabajo verticalmente, ocupando la mayor extensión posible y quedando accesibles todos ellos, como se muestra en la siguiente figura.



Para elegir la tercera, llamada **"Cascada"**, ha de hacer clic sobre dicha opción en el menú desplegado. Esta acción hará que los escenarios abiertos actualmente se distribuyan desde la esquina superior izquierda del área de trabajo, solapándose unas a otras pero quedando todos accesibles, como se muestra en la siguiente figura.



Para elegir la cuarta, llamada **"Iconos"**, ha de hacer clic sobre dicha opción en el menú desplegado. Esta acción hará que los escenarios abiertos actualmente se minimicen quedando todos con forma de iconos apilados en la parte inferior del área de trabajo y siendo todos accesibles, como se muestra en la siguiente figura.



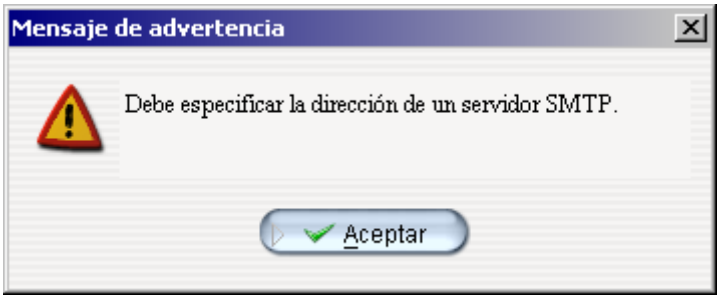
3.3.5.9. Ponerse en contacto con los autores

Open SimMPLS pretende ser un proyecto vivo. Tanto en el número y la calidad de tecnologías investigadas como en la mejora en el funcionamiento del simulador. Para ello, en la web del proyecto se encuentra una zona de retroalimentación donde se pueden enviar comentarios al respecto. Desde el propio simulador también es posible enviar estos comentarios. Para ello haga clic sobre el menú **"Ayuda"** de la ventana principal del simulador.

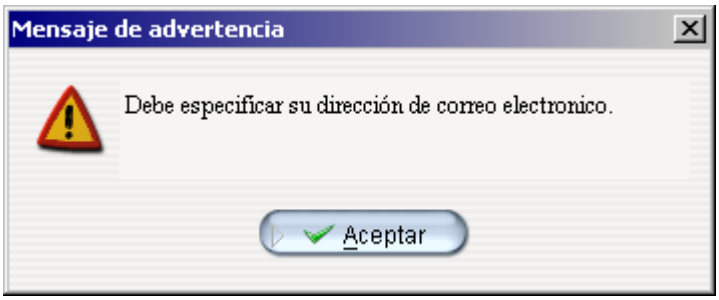
Y seleccione la opción **"Contacta con los autores"**. También puede invocar a esta opción desde cualquier lugar de la zona de trabajo, pulsando la tecla "**F1"**. Entonces aparecerá la siguiente ventana, que nos permitirá enviar un comentario a los autores del proyecto.

En el recuadro nombrado **"Nombre del servidor SMTP"**, deberá escribir el nombre de servidor de correo saliente que le haya proporcionado su proveedor de servicios de Internet. Generalmente suele tener el nombre **"smtp.unDominio.unPais"**. Pero deberá averiguar cuál es concretamente para usted el servidor SMTP a utilizar. En el recuadro **"Su dirección de correo electrónico"** deberá escribir su propia dirección de correo electrónico; No será almacenada por lo desarrolladores de Open SimMPLS, pero será utilizada para responderle si la pregunta que está realizando necesita respuesta. Y por último, en el recuadro central debe escribir el mensaje que desee hacer llegar a los autores de Open SimMPLS 1.0.

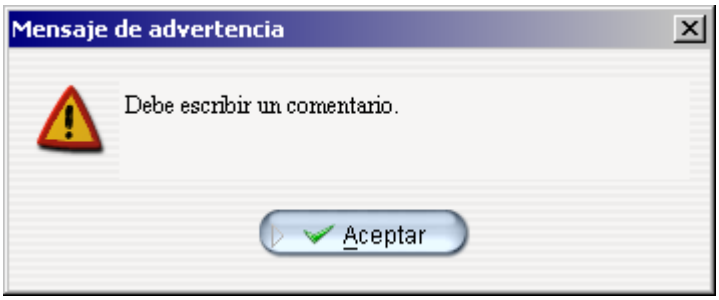
Si en cualquier momento decide no enviar el mensaje, pulse sobre el botón **"Cancelar"**. La ventana de contacto desaparecerá y todo volverá a como estaba antes. Si por el contrario ya está listo para enviar un mensaje a los autores, pulse sobre el botón **"Enviar"**. Si todo va correctamente, la ventana de contacto desaparecerá silenciosamente y eso significará que ha enviado el mensaje correctamente. Si algo falla, se mostrará la correspondiente ventana de error, como se muestra en las siguientes figuras.



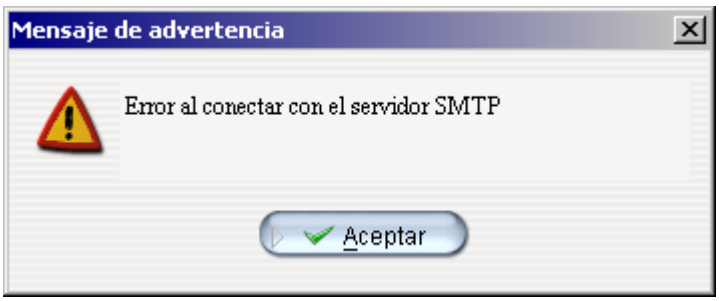
Indica que en el recuadro donde debería haber escrito el nombre del servidor SMTP encargado de enviar el mensaje a los autores, no ha escrito nada. Sin este dato Open SimMPLS no puede enviar el comentario.



Indica que en el recuadro donde debería haber escrito su dirección de correo electrónico, no ha escrito nada. Sin este dato Open SimMPLS no puede enviar el comentario, porque es necesaria su dirección para poder responderle.



Indica que en el recuadro donde debería haber escrito el comentario que desea enviar, no ha escrito nada. Open SimMPLS no enviará un comentario vacío a los autores porque carece de toda lógica.



Indica que no se ha podido contactar con el servidor de SMTP que ha indicado. Muy frecuentemente indica que ha escrito incorrectamente la dirección del servidor SMTP.

Open SimMPLS le mostrará más ventanas con mensajes de error como este, en todas las situaciones en las que la comunicación falle antes de concluir.

Por último, recuerde que para poder enviar un comentario a los autores del simulador, necesita estar conectado en ese momento a Internet.

3.3.5.10. Ayuda del programa

Actualmente la versión 1.0 de Open SimMPLS no incorpora ayuda. En su lugar, se puede acceder vía Internet a la documentación online y descargable en la web oficial del simulador. Para saber más, haga clic sobre el menú **"Ayuda"** de la ventana principal del simulador.

Y en el menú que se despliega, haga clic con el botón principal del ratón sobre la opción **"Contenidos"**. Puede obtener el mismo resultado pulsando la tecla **F1** desde cualquier lugar siempre y cuando esté situado en la ventana principal de la aplicación. Aparecerá la siguiente ventana.

Que indica precisamente eso, que la documentación se puede descargar en la web de Open SimMPLS 1.0 y que en esta versión no está disponible en la propia aplicación.

3.3.5.11. Sobre Open SimMPLS 1.0

El simulador ofrece la posibilidad de mostrar una ventana con los créditos de la aplicación. Si desea visualizarla, haga clic sobre el menú **"Ayuda"** de la ventana principal del simulador.

Y luego seleccione la opción **"Sobre"**. Tras ello, una ventana sin decoraciones aparecerá en el centro de la pantalla y mostrará los créditos, como se ve a continuación.

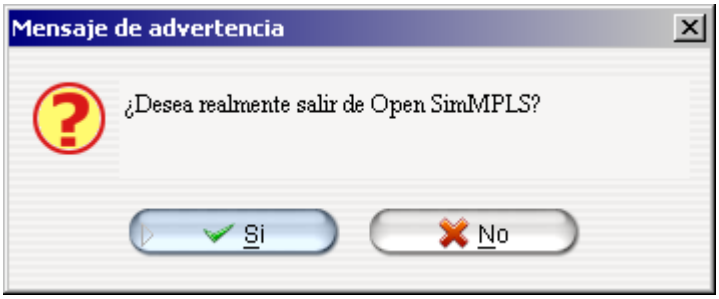
Para cerrar esta ventana, haga clic sobre ella con el botón principal del ratón. La ventana desaparecerá y el simulador volverá a quedar en el mismo estado que antes de que apareciese.

3.3.5.12. Salir de Open SimMPLS 1.0

Para salir de simulador en cualquier momento, simplemente debe acudir al menú principal de la aplicación y hacer clic sobre la opción “Escenarios” para que se despliegue dicho menú.

Y una vez desplegado, seleccionar la opción **“Salir”** mediante un clic con el botón principal del ratón sobre ella. También puede conseguir el mismo efecto pulsando la combinación de teclas **“Control+X”** desde cualquier lugar del área de trabajo.

En este momento el simulador presentará la siguiente ventana.



Que nos pregunta si estamos seguros de que deseamos salir del simulador. Si no es así y ha cambiado de idea, pulse sobre el botón **"No".** La ventana de diálogo se cerrará y usted permanecerá en el simulador. Si desea salir, pulse sobre **"Si"** y el simulador se cerrará. Es posible que al cerrarse y salir, antes pregunte si desea almacenar algún escenario que no haya guardado aún. En ese caso lea en este mismo manual cómo guardar escenarios que es donde se explica correctamente esta operación.

3.3.6. Certificación OSI

Open SimMPLS está liberado bajo licencia GPL v2.0 de la Fundación para el Software Libre. La autoridad encargada de calificar un determinado software como "Iniciativa de Software Libre" contempla esta licencia, entre muchas otras, como válida, e indica que la liberación de un software bajo licencia GPL implica que de forma implícita obtiene la certificación como "Open Software Initiative".

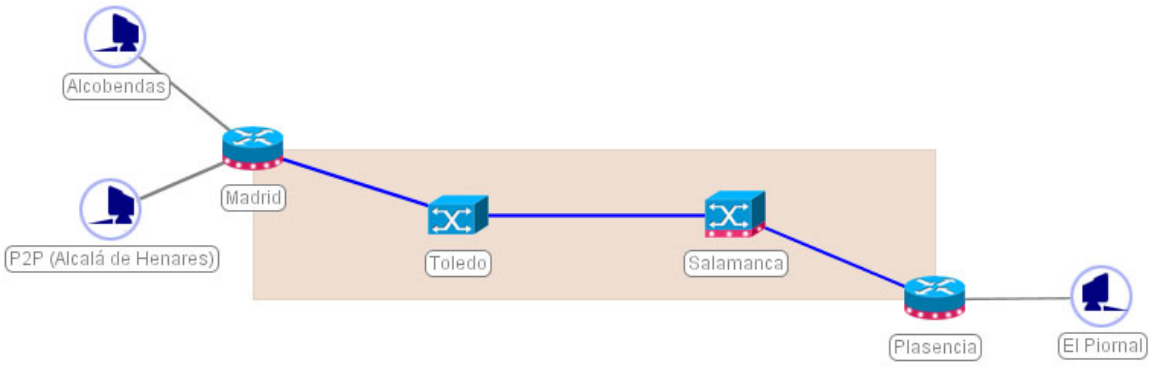
Por tanto, Open SimMPLS está certificado como Open Software Initiative y puede utilizar el distintivo que está sobre estas líneas como muestra de ello.

3.4. Estudio práctico. Resultados.

Para comprobar que el resultado de la aplicación de la tecnología propuesta es el esperado, se han realizado una serie de pruebas prácticas con el simulador, intentando aislar el todo momento el problema que se desea estudiar y estableciendo una comparativa con los escenarios antes y después de la aplicación de la propuesta de este proyecto. En las siguientes páginas se muestran estos casos y los resultados obtenidos.

3.4.1. Escenario 1: comprobación de la recuperación de paquetes

En este ejemplo se ha intentado demostrar que la propuesta que se ha planteado en este proyecto, todo el conjunto de técnicas, en general consiguen recuperar paquetes con requerimientos de GoS. Para ello se ha utilizado el siguiente escenario.



El tráfico fluye de izquierda a derecha desde los nodos emisores hasta llegar al nodo receptor. La configuración de cada elemento es la siguiente:

ENLACES		
Identificación del enlace		Retardo
Origen del enlace	Destino del enlace	
Alcobendas	Madrid	3.000 ns.
P2P (Alcalá de Henares)	Madrid	1.000 ns.
Madrid	Toledo	1.000 ns.
Toledo	Salamanca	1.000 ns.
Salamanca	Plasencia	3.000 ns.
Plasencia	El Piornal	60.000 ns.

CONMUTADORES / ENCAMINADORES			
Nombre	**Potencia conmutación**	**Tamaño del búfer**	**Tamaño DMGP**
Madrid	10 Gbps.	100 MB.	1.024 KB.
Toledo	10 Gbps.	5 MB.	N/A
Salamanca	10 Gbps.	6 MB.	100 KB.
Plasencia	6 Gbps.	1 MB.	10.240 KB.

EMISORES						
Nombre	**Tasa de generación**	**Tipo de tráfico**	**Carga útil de los paquetes**	**¿Sobre MPLS?**	**Nivel GoS**	**¿LSP de respaldo?**
Alcobendas	10 Gbps.	CONST.	100 B.	NO	3	NO
P2P (Alcalá)	10 Gbps.	CONST.	100 B.	NO	NO	NO

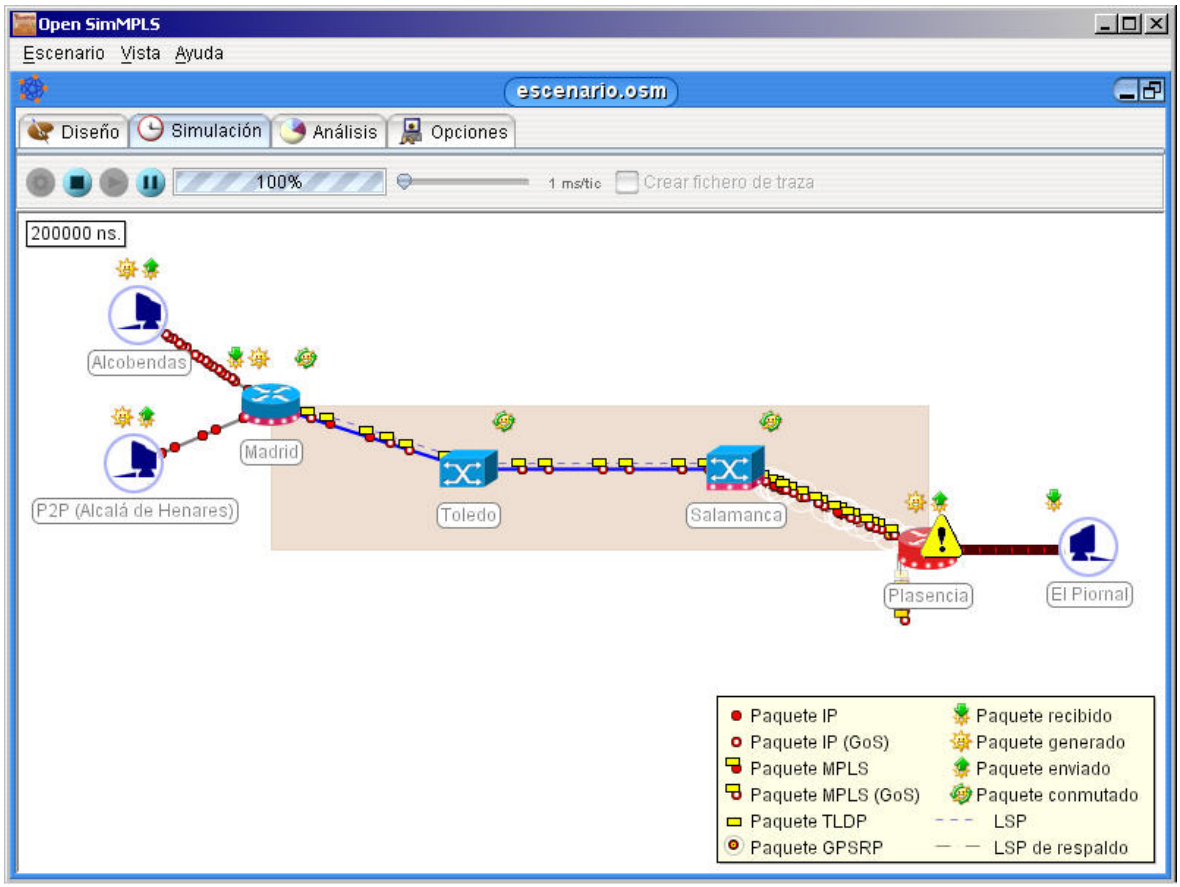
RECEPTORES
En Open SimMPLS 1.0 los receptores son ideales. No se parametrizan.

Y la configuración temporal de la simulación es como sigue:

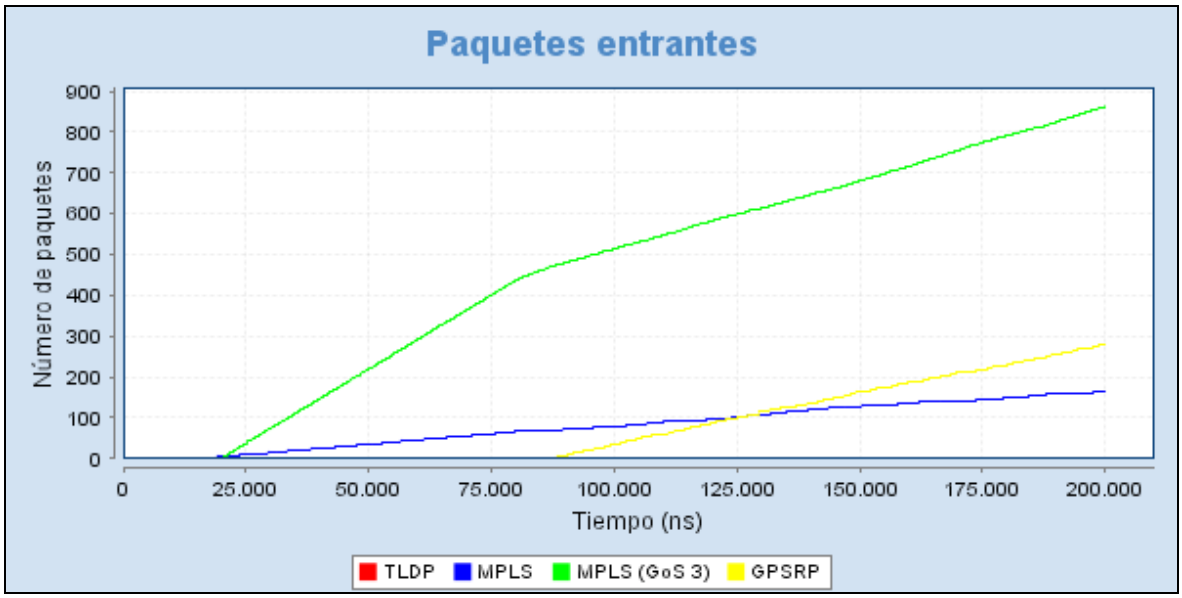
Duración	Paso
200.000 ns.	100 ns.

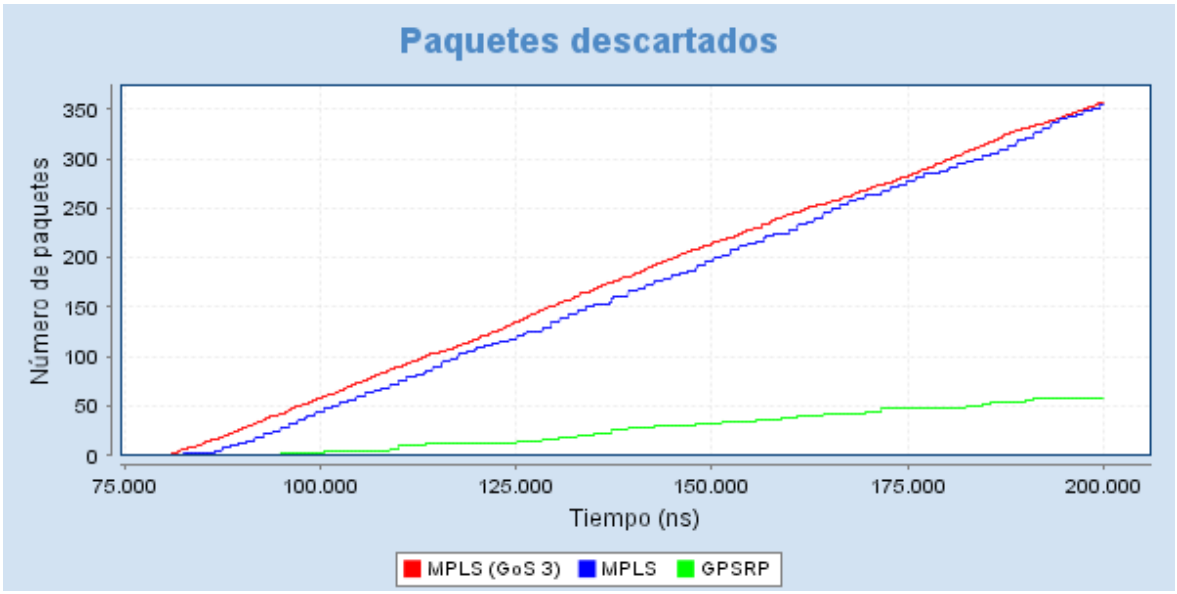
El tráfico generado, como se puede observar, es idéntico salvo que uno está marcado con GoS y el otro no. Esto está hecho así a propósito para ver, en igualdad de condiciones, cómo se trata a un tráfico y a otro por parte de la red.

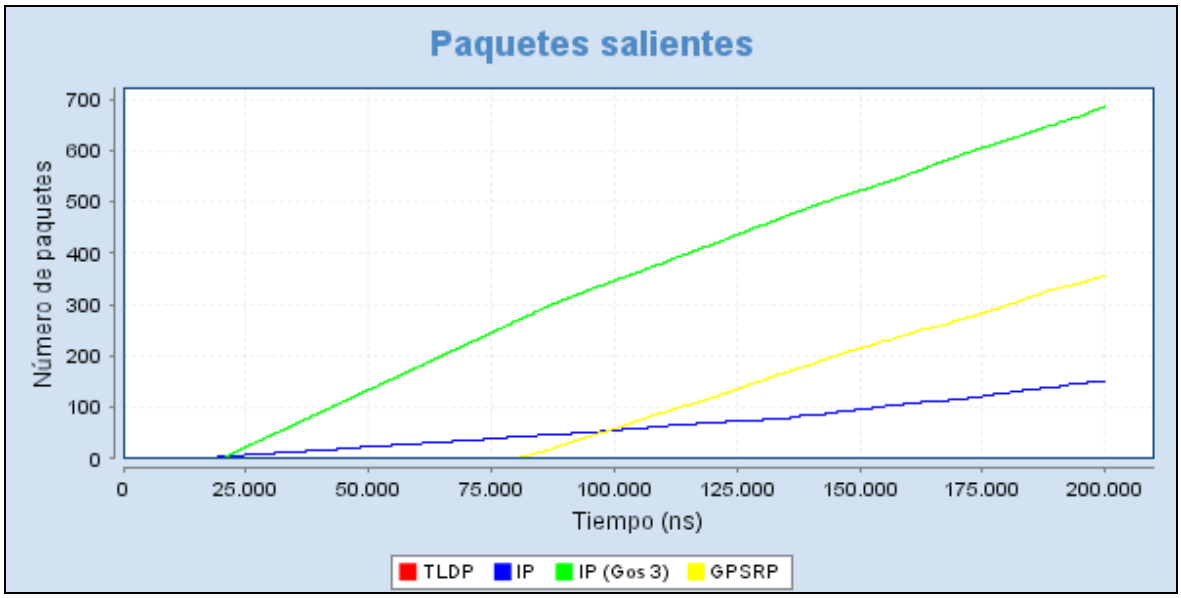
En la figura siguiente observamos una captura de la pantalla del simulador durante la simulación de este escenario.



En el instante 0 de simulación, Plasencia se congestiona artificialmente, lo que produce que parta de un 97% de congestión en el buffer. Esto es así para que provoque pérdidas en un tiempo prudencial dado el poco tiempo de duración de las simulaciones. El estudio se centra en el nodo Plasencia, que será el que se congestionará. Todas las gráficas se refieren a él. Los resultados obtenidos son los siguientes:


Paquetes entrantes
Número de paquetes
900
800
700
600
500
400
300
200
100
0
0
25.000
50.000
75.000
100.000
125.000
150.000
175.000
200.000
Tiempo (ns)
TLDP
MPLS
MPLS (GoS 3)
GPSRP


Paquetes descartados
Número de paquetes
350
300
250
200
150
100
50
0
75.000
100.000
125.000
150.000
175.000
200.000
Tiempo (ns)
MPLS (GoS 3)
MPLS
GPSRP



La gráfica de paquetes entrantes muestra el número y la distribución de paquetes recibidos en el nodo “Plasencia” durante el paso del tiempo. Por razones de escala no se visualiza bien el tráfico TLDP (sólo se reciben dos paquetes y el rango está ajustado a novecientos). En el nanosegundo 6.000 recibe un primer paquete TLDP, para crear el LSP para el tráfico de Alcalá de Henares. En torno al nanosegundo 8.000 recibe el segundo y último paquete TLDP, para crear el LSP para el tráfico de Alcobendas.

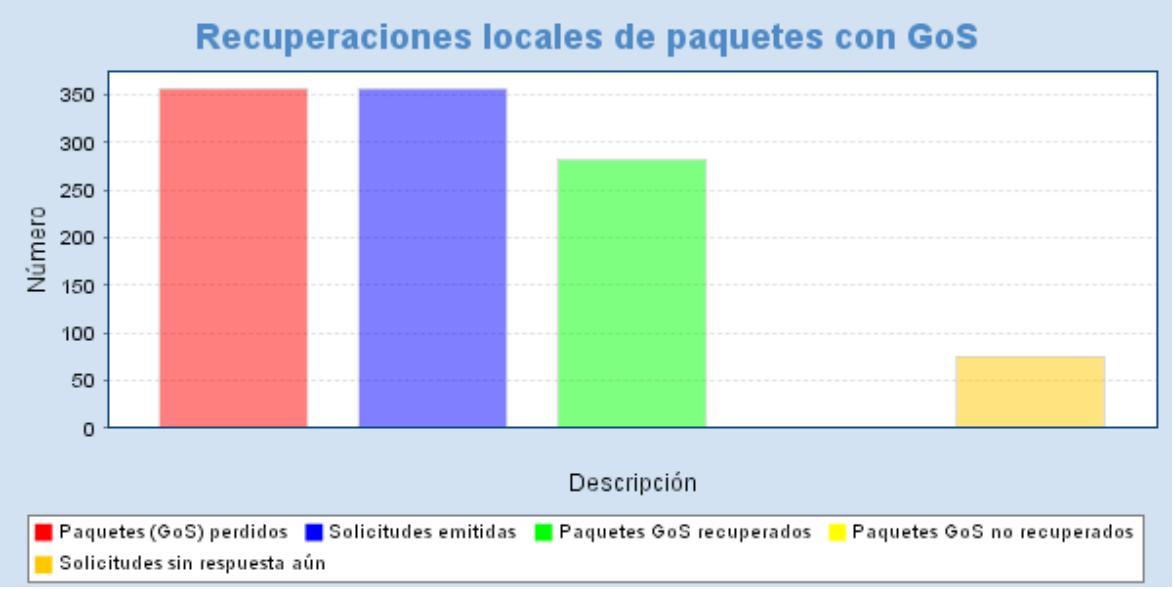
No se recibe más tráfico hasta aproximadamente el instante 18.000, cuando comienza a llegar tráfico MPLS y MPLS GoS 3. El primero en llegar es de tipo MPLS debido a que su LSP se formalizó antes (menos retardo en su enlace) y a partir de ahí comienza a llegar tráfico entremezclado de ambos tipos, siempre en mayor cantidad (aproximadamente 5-6 veces más) de MPLS GoS 3 que de MPLS tradicional. Esto es debido a que todo el tráfico pasa por Madrid y Salamanca antes de llegar a Plasencia; Madrid y Salamanca son nodos activos y como tales procesan más paquetes de tráfico con requerimientos de GoS que tráfico sin ellos; Así que en el recorrido antes de llegar a Plasencia, el tráfico MPLS tradicional va siendo paulatinamente relegado en favor del tráfico MPLS GoS 3. En este primer paso ya podemos observar cómo la existencia de nodos activos en la ruta va ejerciendo el efecto esperado.

A medida de que pasa el tiempo, el búfer de Plasencia, que comenzó al 97% de congestión, se va llenando hasta que en el instante 85.000 aproximadamente, se satura al máximo y comienza a descartar paquetes lo cual se puede observar correctamente en la gráfica de paquetes descartados. En ese momento se comienza a perder tráfico de ambos tipos

aunque, evidentemente al recibirse mayor cantidad de tráfico marcado con GoS, se pierden más paquetes de este tipo, como se observa en la gráfica de paquetes descartados.

EPCD detecta la pérdida de paquetes con requerimientos de GoS y comienza a enviar paquetes GPSRP hacia Salamanca para recuperarlos. Estas peticiones comienzan a tener respuesta en torno al instante 91.000, cuando las respuestas de Salamanca (y eventualmente el paquete retransmitido) llegan al búfer. A partir de ese momento el flujo constante de paquetes GPSRP recibidos, unido a la congestión del búfer de entrada, produce que el número de paquetes MPLS y MPLS GoS 3 recibidos decaiga un poco. Llega un momento que el número de paquetes GPSRP recibidos supera el número de paquetes MPLS recibidos; no hay que llevarse a engaños puesto que si bien el número de paquetes recibidos es mayor, los paquetes MPLS de este escenarios son de 100 octetos mientras que los paquetes GPSRP son paquetes de 9 octetos.

El resultado de la actuación GPSRP en este caso se puede observar en la siguiente gráfica.

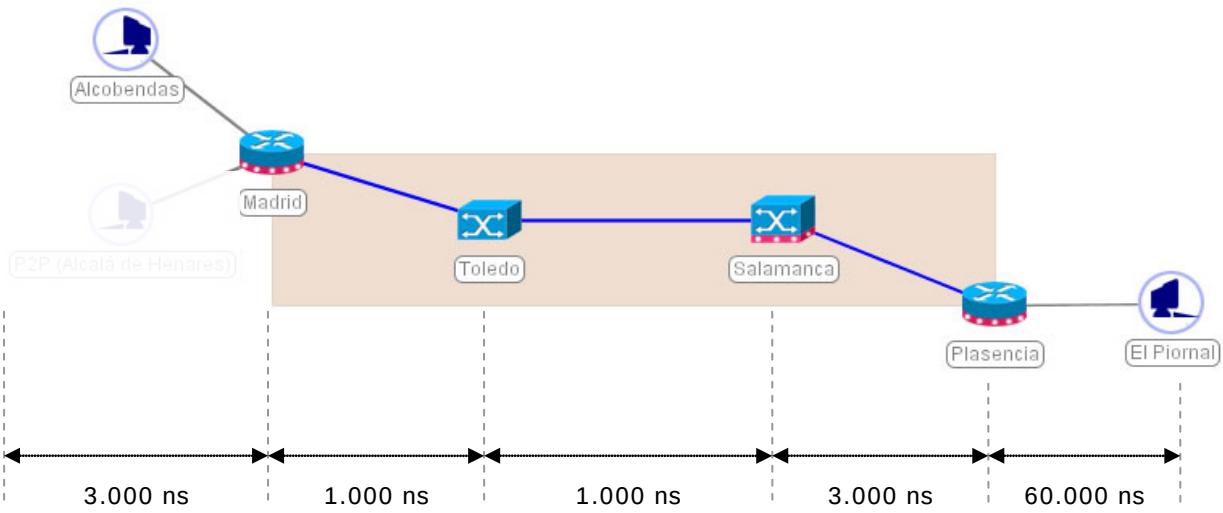


En ella se ve (primera barra) que han sido descartados sobre 360 paquetes con requerimientos de GoS. La conjunción de EPCD con GPSRP hizo que para recuperar esos 360 paquetes GoS se emitiesen el mismo número de peticiones de retransmisión de paquetes hacia Salamanca (segunda barra), el último nodo activo atravesado por los paquetes perdidos antes de llegar a Plasencia. Como fruto de estas peticiones, Salamanca

confirmó la retransmisión de 280 paquetes (tercera barra). En ningún caso contestó con una denegación (cuarta barra) sino que el resto de peticiones están aún sin contestar; están en tránsito (quinta barra).

Como conclusión de esta simulación, podemos confirmar que la propuesta de recuperación de paquetes con GoS realizada en este proyecto funciona; a decir verdad y a juzgar por los datos, funciona bastante bien. No sólo se han procesado muchos más paquetes MPLS GoS 3 que paquetes MPLS tradicionales, sino que además se han recuperado (hasta el momento) 280 paquetes de los 360 que se han descartado; esos paquetes no hubiesen sido recuperables localmente de no ser por esta propuesta; hubiesen tenido que ser recuperados extremo a extremo, con el retardo que ello supone.

Haciendo los cálculos del tiempo que se ha podido ahorrar **por cada paquete perdido** y recuperado, obtenemos los siguientes resultados.



Si Plasencia no fuese un nodo activo y allí se descartase un paquete proveniente de Alcobendas a El Piornal, tardaría en recuperarse:

Tiempo en detectar TCP trozos fuera de orden en El Piornal: X
Tiempo en solicitar retransmisión a Alcobendas: 68.000
Tiempo en llegar la retransmisión de Alcobendas a El Piornal: 68.000

Total → 136.000 + X ns.

Si el paquete perdido requiere GoS y los nodos son activos, como es el caso, si el paquete se descarta en Plasencia, tardaría en ese momento en recuperarse y llegar a El Piornal:

En el mejor de los casos (Salamanca reenvía):

Plasencia – Salamanca: 3.000
Salamanca – Plasencia: 3.000
Plasencia – El Piornal: 60.000

Total: 66.000 ns.

En el caso de tener que recurrir a Madrid para la retransmisión:

Plasencia – Salamanca: 3.000
Salamanca – Plasencia: 3.000
Plasencia – Salamanca: 3.000
Salamanca – Madrid: 2.000
Madrid – Salamanca: 2.000
Salamanca – Plasencia: 3.000
Plasencia – El Piornal: 60.000

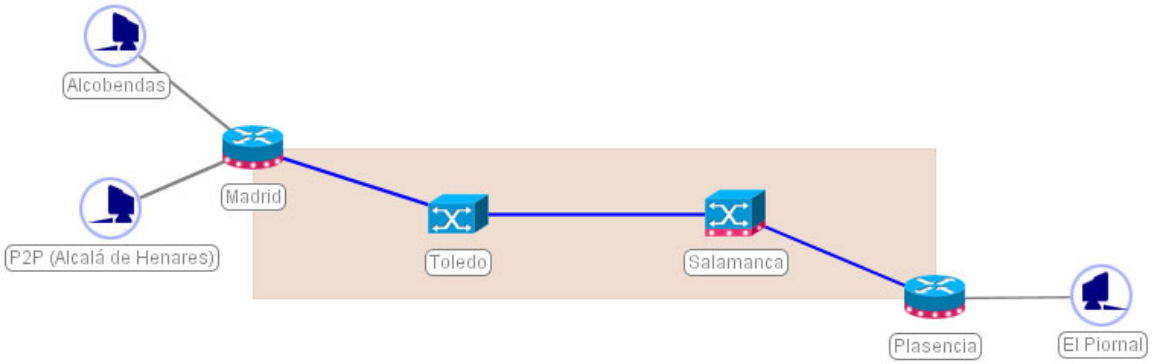
Total: 76.000 ns.

Es decir, en el peor de los casos la recuperación local ahorra un tiempo de 136.000 – 76.000 = **60.000 ns.** por cada paquete descartado.

En cualquiera de los dos casos, el tiempo de recuperación local es muy inferior al tiempo de recuperación extremo a extremo por los protocolos superiores. De cualquier forma, en el caso de no poder recuperar el paquete, los niveles superiores entran en juego.

3.4.2. Escenario 2: efecto del nivel de GoS en las recuperaciones

Para este caso, se ha simulado tres veces distintas el escenario, cambiando únicamente entre una vez y otra el nivel de GoS del tráfico generado en Alcobendas. Se intenta así estudiar el efecto que produce el nivel de GoS del tráfico en el número de paquetes que se consiguen recuperar localmente. Se ha utilizado el siguiente escenario.



El tráfico fluye de izquierda a derecha desde los nodos emisores hasta llegar al nodo receptor. La configuración de cada elemento es la siguiente:

ENLACES		
Identificación del enlace		Retardo
Origen del enlace	Destino del enlace	
Alcobendas	Madrid	3.000 ns.
P2P (Alcalá de Henares)	Madrid	1.000 ns.
Madrid	Toledo	1.000 ns.
Toledo	Salamanca	1.000 ns.
Salamanca	Plasencia	3.000 ns.
Plasencia	El Piornal	60.000 ns.

CONMUTADORES / ENCAMINADORES			
Nombre	Potencia conmutación	Tamaño del búfer	Tamaño DMGP
Madrid	10 Gbps.	100 MB.	1.024 KB.
Toledo	10 Gbps.	5 MB.	N/A
Salamanca	10 Gbps.	6 MB.	100 KB.

Plasencia	6 Gbps.	1 MB.	10.240 KB.

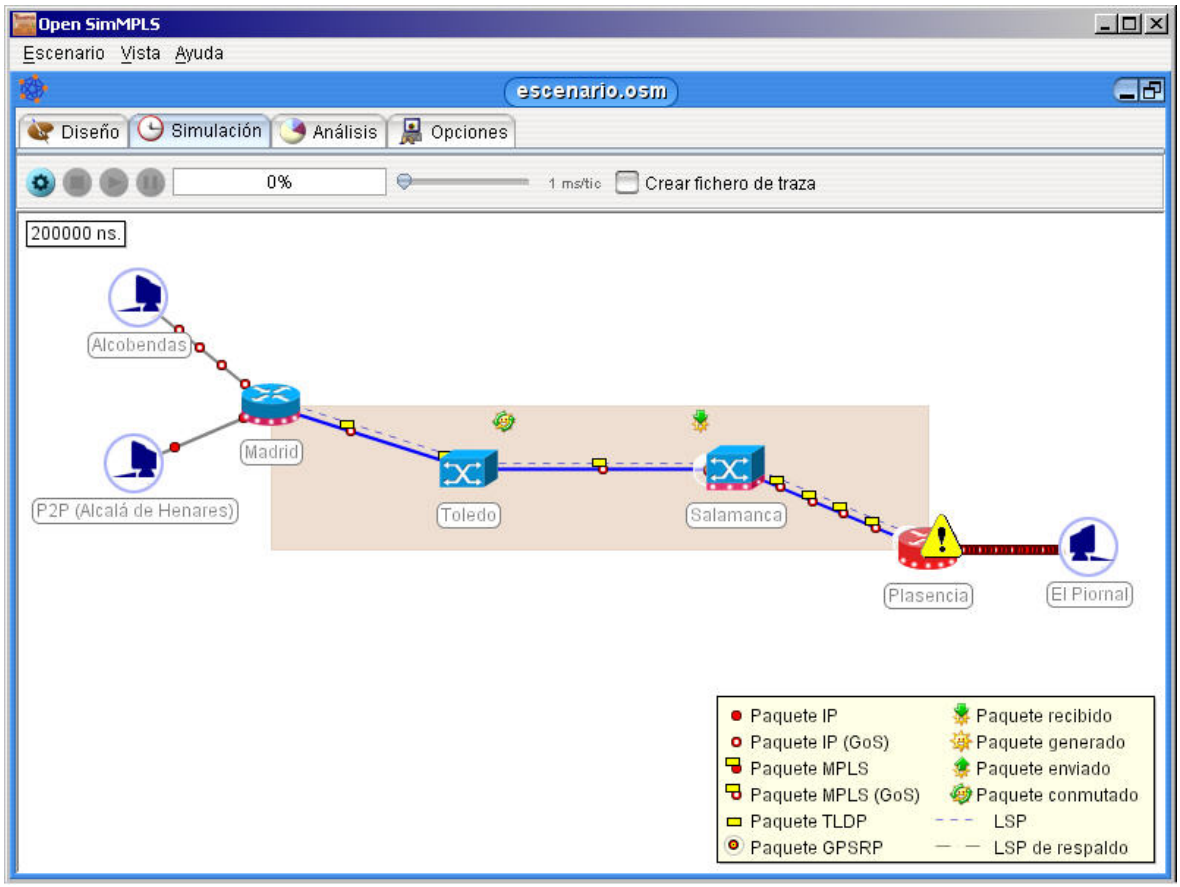
EMISORES						
Nombre	**Tasa de generación**	**Tipo de tráfico**	**Carga útil de los paquetes**	**¿Sobre MPLS?**	**Nivel GoS**	**¿LSP de respaldo?**
Alcobendas (sim. 1)	10 Gbps.	CONST.	618 B.	NO	1	NO
Alcobendas (sim. 2)	10 Gbps.	CONST.	618 B.	NO	2	NO
Alcobendas (sim. 3)	10 Gbps.	CONST.	618 B.	NO	3	NO
P2P (Alcalá)	10 Gbps.	CONST.	618 B.	NO	NO	NO

RECEPTORES
En Open SimMPLS 1.0 los receptores son ideales. No se parametrizan.

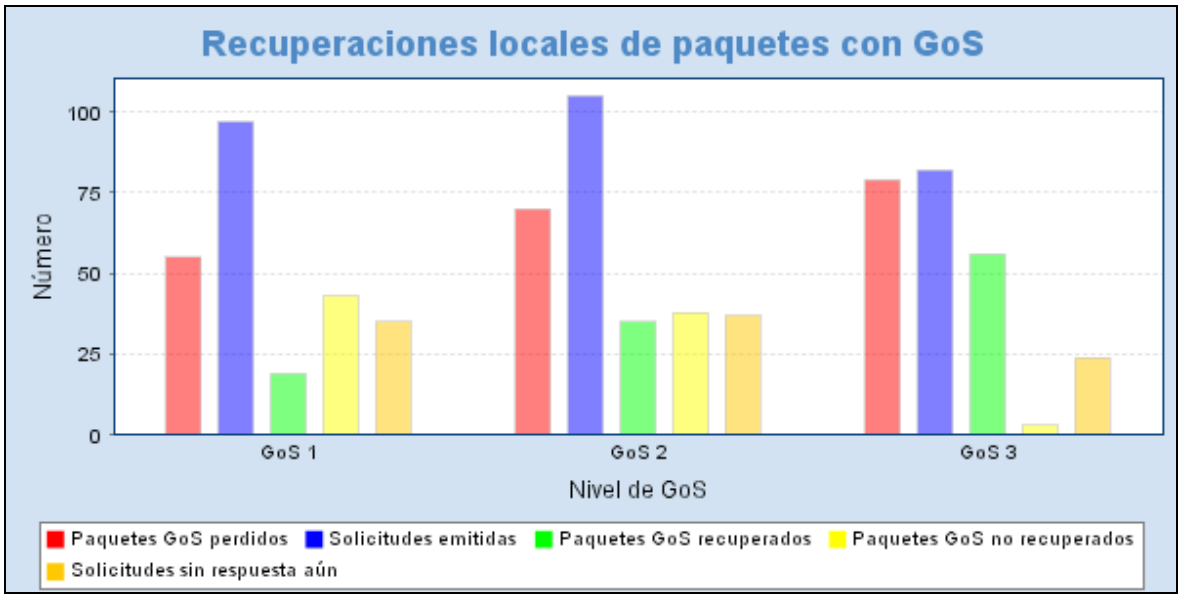
Y la configuración temporal de la simulación es como sigue:

Duración	Paso
200.000 ns.	100 ns.

El tráfico generado, como se puede observar, es idéntico salvo que uno está marcado con GoS y el otro no. Esto está hecho así a propósito para ver, en igualdad de condiciones, cómo se trata a un tráfico y a otro por parte de la red. En la figura siguiente observamos una captura de la pantalla del simulador durante la simulación de este escenario.



Y los datos que se han obtenido en Plasencia, relativos a las recuperaciones son los siguientes:



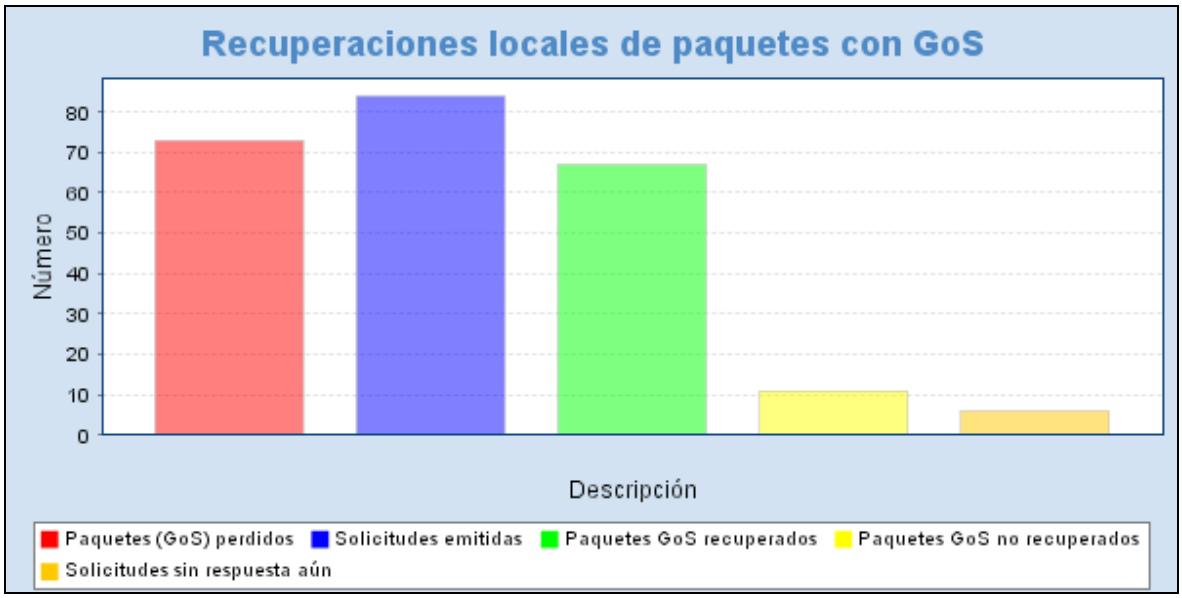
Con un nivel de GoS 1, sólo el 4% de la DMGP de los nodos es reservada para almacenar paquetes del flujo marcado con GoS 1. Se puede observar que en este caso se han descartado en Plasencia 55 paquetes y el conjunto EPCD/GPSRP ha necesitado 97 solicitudes de retransmisión para recuperar 19 de ellos. ¿Por qué más solicitudes de recuperación que paquetes que recuperar? Porque Salamanca no tiene DMGP suficiente y como sólo reserva el 4% de la misma para almacenar paquetes con GoS, le caben muy pocos. Entonces contesta a Plasencia con una negación de retransmisión y éste debe volver a emitir una petición de retransmisión para el mismo paquete, esta vez a Madrid. Madrid tiene más memoria DMGP y aunque también reserva el 4% para el flujo de Alcobendas, este 4% permite almacenar más paquetes que en el caso de Salamanca. En este caso la mayor parte de las recuperaciones se hacen vía Madrid. Aún así muchas de las peticiones realizadas llegan a Madrid cuando este ya o tiene los paquetes solicitados en su DMGP y por ello contesta con negaciones y, en este caso, los paquetes se consideran irrecuperables a todos los efectos (los niveles superiores intervendrán extremo a extremo); concretamente 43 paquetes no se han podido recuperar. Pero aún quedan 35 peticiones pendientes de contestación. En este caso se han recuperado el 34’5% de los paquetes perdidos en Plasencia.

Con un nivel de GoS 2, el 8% de la DMGP de los nodos es reservada para almacenar paquetes del flujo marcado con GoS 2. Se puede observar que en este caso se han descartado en Plasencia 70 paquetes y al conjunto EPCD/GPSRP ha necesitado 105 solicitudes de retransmisión para recuperar 35 de ellos. ¿Por qué más solicitudes de recuperación que paquetes que recuperar? Igual que en el caso anterior, porque Salamanca no tiene DMGP suficiente y como sólo reserva el 8% de la misma para almacenar paquetes con GoS, le caben muy pocos, aunque más que si hubiese reservado para GoS 1. Entonces en la mayoría de los casos contesta a Plasencia con una negación de retransmisión y éste debe volver a emitir una petición de retransmisión para el mismo paquete, esta vez a Madrid. Madrid tiene más memoria DMGP y aunque también reserva el 8% para el flujo de Alcobendas, este 8% permite almacenar más paquetes que en el caso de Salamanca. En este caso la mayor parte de las recuperaciones se hacen vía Madrid, aunque comparando con el caso de GoS 1, Salamanca recupera más en este caso. Aún así muchas de las peticiones realizadas llegan a Madrid cuando este ya o tiene los paquetes solicitados en su DMGP y por ello contesta con negaciones y, en este caso, los paquetes se consideran

irrecuperables a todos los efectos (los niveles superiores intervendrán extremo a extremo); concretamente 38 paquetes no se han podido recuperar. Pero aún quedan 37 peticiones pendientes de contestación. En este caso se han recuperado el 50% de los paquetes perdidos en Plasencia, lo que es un importante aumento con respecto a si etiquetamos el flujo con GoS 1.

Es destacable en este caso el hecho de que se pierdan más paquetes con GoS que en el caso del flujo marcado con GoS 1; resulta raro. La explicación es la siguiente; en el primer caso, prácticamente no se hacían retransmisiones desde Salamanca; casi todas se hacían desde Madrid y tampoco eran demasiadas. El tráfico de Madrid llega a Toledo y éste, por no ser activo, actúa de freno del paquete privilegiado (Round Robin sin prioridades) así que los paquetes retransmitidos se detienen en Toledo que "regulariza" la cadencia de tráfico que sigue llegando con la rapidez normal a Salamanca y después Plasencia. En el caso de GoS 2, Salamanca retransmite más paquetes a Plasencia que en el primer caso y lo hace directamente; esto provoca que se genere un aumento local del tráfico que se añade al tráfico proveniente de Toledo y provoca que Plasencia, que está saturado, tenga que descartar más paquetes. Pero en la mayor parte de los casos los paquetes descartados son los propios paquetes retransmitidos, para los que se vuelve a solicitar retransmisión. En cualquier caso, el porcentaje de recuperaciones aumenta. Como aclaración, hay que saber que este efecto negativo es producto de que la simulación se está realizando con tráfico constante que no permiten recuperarse al nodo una vez está congestionado. El tráfico a ráfagas y de tamaño variable de Internet, con el cual se ha probado este escenario, permite resultados mejores en el número de recuperaciones sin ese aumento de las pérdidas porque las mismas ráfagas que pueden congestionar un nodo, pueden producir el efecto contrario, permitiendo la entrada en el búfer de los paquetes retransmitidos.

Como ejemplo ilustrativo, aquí está la gráfica de Plasencia cuando establecemos el tráfico de Alcalá de Henares como variable, dejando todo lo demás intacto y el tráfico de Alcobendas como GoS 2. Este resultado, como el tráfico, es variable.



En este caso el número de pérdidas no se puede comparar con las gráficas anteriores; responde a la naturaleza variable del tráfico real de Internet. Pero el hecho de que haya ráfagas posibilita que el nivel de congestión fluctúe y las recuperaciones tengan cabida en el búfer del nodo; En este caso (y en muchas otras pruebas con tráfico real variable) se consigue un porcentaje de recuperaciones superior al 90% (del 91'7%).

Con un nivel de GoS 3, el 12% de la DMGP de los nodos es reservada para almacenar paquetes del flujo marcado con GoS 3. Se puede observar que en este caso se han descartado en Plasencia 79 paquetes y al conjunto EPCD/GPSRP ha necesitado 82 solicitudes de retransmisión para recuperar 56 de ellos. ¿Por qué más solicitudes de recuperación que paquetes que recuperar? Igual que en el caso anterior, porque Salamanca no tiene DMGP suficiente para dar servicio a todas las peticiones de retransmisión y como sólo reserva el 12% de la misma para almacenar paquetes con GoS, le caben muy pocos, aunque más que si hubiese reservado para GoS 2 y, en este caso, casi lo suficiente para poder responder a todas las peticiones. En algún caso contesta a Plasencia con una negación de retransmisión y éste debe volver a emitir una petición de retransmisión para el mismo paquete, esta vez a Madrid. Madrid tiene más memoria DMGP y aunque también reserva el 12% para el flujo de Alcobendas, este 12% permite almacenar más paquetes que en el caso de Salamanca. En este caso una parte mínima de las recuperaciones se hacen vía Madrid; Salamanca retransmite casi todos los paquetes en este caso. Aún así de las pocas peticiones que llegan a Madrid alguna llega cuando este ya o tiene los paquetes solicitados

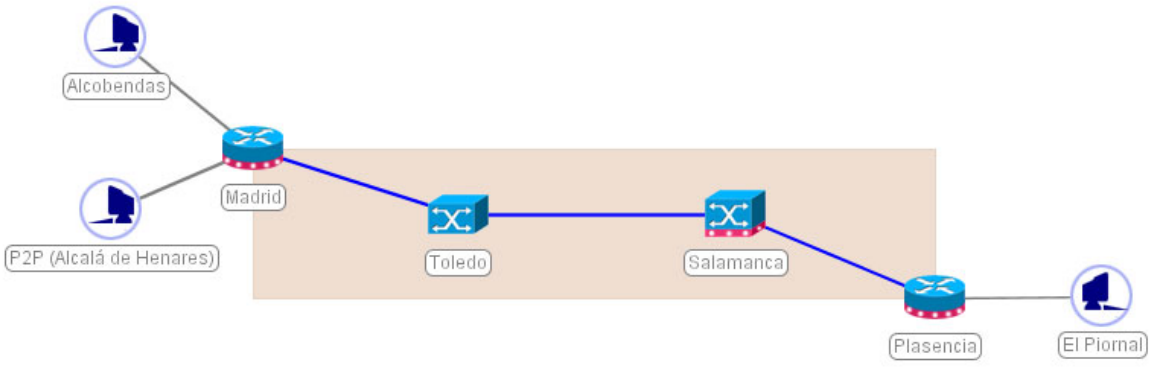
en su DMGP y por ello contesta con negaciones y, en este caso, los paquetes se consideran irrecuperables a todos los efectos (los niveles superiores intervendrán extremo a extremo); concretamente 2 paquetes no se han podido recuperar. Pero aún quedan 24 peticiones pendientes de contestación. En este caso se han recuperado el 70'8% de los paquetes perdidos en Plasencia, lo que es un importante aumento con respecto a si etiquetamos el flujo con GoS 2 y más con respecto a GoS 1.

El aumento del número de pérdidas a medida que la GoS aumenta se ha justificado unos párrafos más arriba.

Como conclusión, tenemos que con GoS 1 se recuperan el 34'5% de los paquetes, con GoS 2, el 50% de los paquetes y con GoS 3, el 70'8. Podemos afirmar que tras las pruebas, la propuesta de este PFC recupera paquetes y lo hace en mayor medida cuando los paquetes están marcados con un grado mayor de GoS.

3.4.3. Escenario 3: efecto del tamaño de la DMGP en las recuperaciones

Para este caso, se ha simulado tres veces distintas el escenario, cambiando únicamente entre una vez y otra el tamaño de la memoria DMGP de Salamanca. Se intenta así estudiar el efecto que produce el tamaño de la DMGP del nodo al que se solicitan retransmisiones en el número de paquetes que se consiguen recuperar localmente. Se ha utilizado el siguiente escenario.



El tráfico fluye de izquierda a derecha desde los nodos emisores hasta llegar al nodo receptor. La configuración de cada elemento es la siguiente:

ENLACES		
Identificación del enlace		Retardo
Origen del enlace	Destino del enlace	
Alcobendas	Madrid	3.000 ns.
P2P (Alcalá de Henares)	Madrid	1.000 ns.
Madrid	Toledo	1.000 ns.
Toledo	Salamanca	1.000 ns.
Salamanca	Plasencia	3.000 ns.
Plasencia	El Piornal	60.000 ns.

CONMUTADORES / ENCAMINADORES			
Nombre	Potencia conmutación	Tamaño del búfer	Tamaño DMGP
Madrid	10 Gbps.	100 MB.	1 KB.
Toledo	10 Gbps.	5 MB.	N/A
Salamanca (sim. 1)	10 Gbps.	6 MB.	32 KB.
Salamanca (sim. 2)	10 Gbps.	6 MB.	64 KB.
Salamanca (sim. 3)	10 Gbps.	6 MB.	96 KB.
Plasencia	6 Gbps.	1 MB.	10.240 KB.

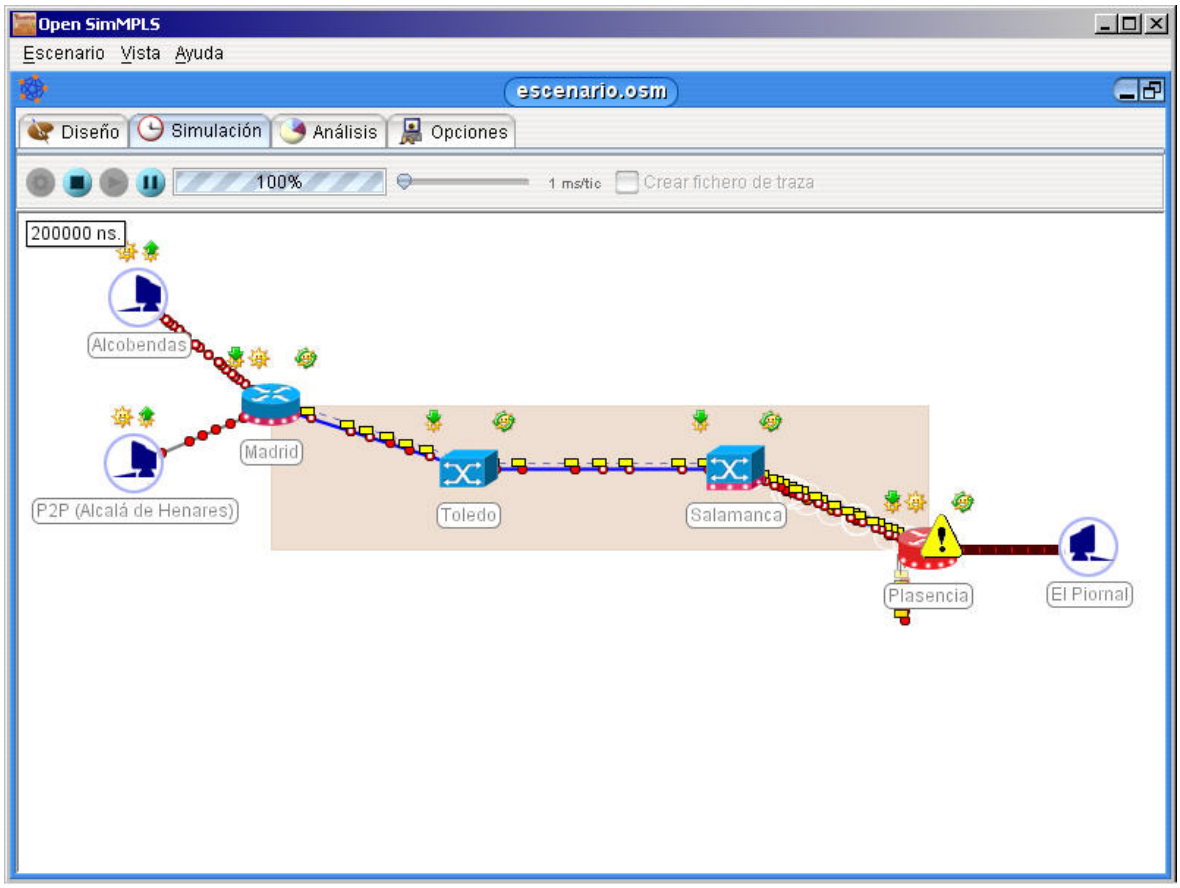
EMISORES						
Nombre	Tasa de generación	Tipo de tráfico	Carga útil de los paquetes	¿Sobre MPLS?	Nivel GoS	¿LSP de respaldo?
Alcobendas	10 Gbps.	CONST.	100 B.	NO	3	NO
P2P (Alcalá)	10 Gbps.	CONST.	100 B.	NO	NO	NO

RECEPTORES
En Open SimMPLS 1.0 los receptores son ideales. No se parametrizan.

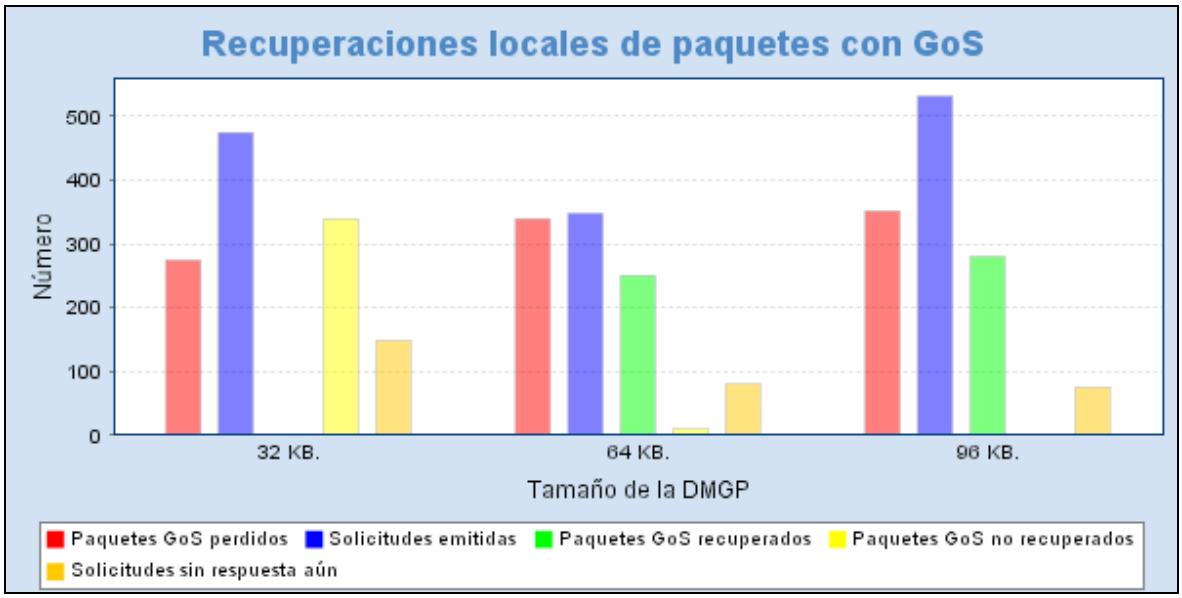
Y la configuración temporal de la simulación es como sigue:

Duración	Paso
200.000 ns.	100 ns.

El tráfico generado, como se puede observar, es idéntico salvo que uno está marcado con GoS y el otro no. Esto está hecho así a propósito para ver, en igualdad de condiciones, cómo se trata a un tráfico y a otro por parte de la red. Además, la DMGP de Madrid se ha establecido a 1 KB. Que en la práctica le impedirá resolver las peticiones que Salamanca no pueda resolver para Plasencia. Así estudiamos aisladamente el efecto del tamaño de la DMGP en Salamanca. En la figura siguiente observamos una captura de la pantalla del simulador durante la simulación de este escenario.



Y los datos que se han obtenido en Plasencia, relativos a las recuperaciones son los siguientes:



En el primer caso se producen 275 descartes de paquetes con GoS en Plasencia y para recuperarlos EPCD/GPSRP inician 475 solicitudes de retransmisión de paquetes. Al final de todo el proceso no se consiguen recuperar ninguno de los paquetes perdidos; es más 340 se han dado por irrecuperables y aunque aún quedan 148 peticiones por resolver, todo augura que no se recuperarán. El proceso es el siguiente: por cada paquete que se pierde en Plasencia, se envía una solicitud de retransmisión a Salamanca. Salamanca tiene una memoria DMGP que le impide almacenar más de unos pocos paquetes del flujo privilegiado y cuando le llegan las solicitudes, los paquetes a los que hacen referencia ya no se encuentran en su DMGP; así que contesta a Plasencia con denegaciones y Plasencia debe volver a solicitar, esta vez a Madrid, la retransmisión de los paquetes. Madrid se ha configurado con 1 KB. De DMGP deliberadamente para que siempre conteste con negaciones. Así que, en este caso, todo paquete que no es capaz de recuperar Salamanca, tampoco se podrá recuperar por Madrid. Con 32 KB. Y esta configuración se han recuperado el 0% de lo paquetes GoS descartados en Plasencia.

En el segundo caso se producen 340 descartes de paquetes con GoS en Plasencia y para recuperarlos EPCD/GPSRP inician 348 solicitudes de retransmisión de paquetes. Al final de todo el proceso se consiguen recuperar 251 de los paquetes perdidos; 10 se han dado por irrecuperables y aún quedan 80 peticiones por resolver. El proceso es el siguiente: por cada paquete que se pierde en Plasencia, se envía una solicitud de retransmisión a Salamanca. Salamanca tiene una memoria DMGP que le impide almacenar demasiados paquetes del

flujo privilegiado (64 KB.) aunque en este caso, son suficientes para dar un buen servicio. Cuando le llegan las solicitudes de retransmisión, alguno de los paquetes a los que hacen referencia ya no se encuentran en su DMGP; así que contesta a Plasencia con denegaciones y Plasencia debe volver a solicitar, esta vez a Madrid, la retransmisión de los paquetes. Madrid se ha configurado con 1 KB. De DMGP deliberadamente para que siempre conteste con negaciones. Así que, en este caso, todo paquete que no es capaz de recuperar Salamanca, tampoco se podrá recuperar por Madrid; por tanto todos los paquetes recuperados han sido gracias a retransmisiones de Salamanca. Con 64 KBde DMGP y esta configuración se han recuperado el 73’82% de lo paquetes GoS descartados en Plasencia.

Es destacable en este caso el hecho de que se pierdan más paquetes con GoS que en el caso de la DMGP de 32 KB; resulta raro. La explicación es la siguiente; en el primer caso, prácticamente no se hacían retransmisiones desde Salamanca (ni desde Madrid). En el caso de la DMGP de 64 KB, Salamanca retransmite muchos más paquetes a Plasencia que en el primer caso y lo hace directamente; esto provoca que se genere un aumento local del tráfico que se añade al tráfico proveniente de Toledo y provoca que Plasencia, que está saturado, tenga que descartar más paquetes. Pero en la mayor parte de los casos los paquetes descartados son los propios paquetes retransmitidos, para los que se vuelve a solicitar retransmisión. En cualquier caso, el porcentaje de recuperaciones aumenta. Como aclaración, hay que saber que este efecto negativo es producto de que la simulación se está realizando con tráfico constante que no permiten recuperarse al nodo una vez está congestionado. El tráfico a ráfagas y de tamaño variable de Internet, con el cual se ha probado este escenario, permite resultados mejores en el número de recuperaciones sin ese aumento de las pérdidas porque las mismas ráfagas que pueden congestionar un nodo, pueden producir el efecto contrario, permitiendo la entrada en el búfer de los paquetes retransmitidos.

En el tercer caso se producen 352 descartes de paquetes con GoS en Plasencia y para recuperarlos EPCD/GPSRP inician 352 solicitudes de retransmisión de paquetes. Al final de todo el proceso se consiguen recuperar 280 de los paquetes perdidos; ninguno se han dado por irrecuperables y aún quedan 75 peticiones por resolver. El proceso es el siguiente: por cada paquete que se pierde en Plasencia, se envía una solicitud de retransmisión a Salamanca. Salamanca tiene una memoria DMGP suficiente (96 KB.) para dar servicio a todas las peticiones de Plasencia. Cuando le llegan las solicitudes de retransmisión, alguno

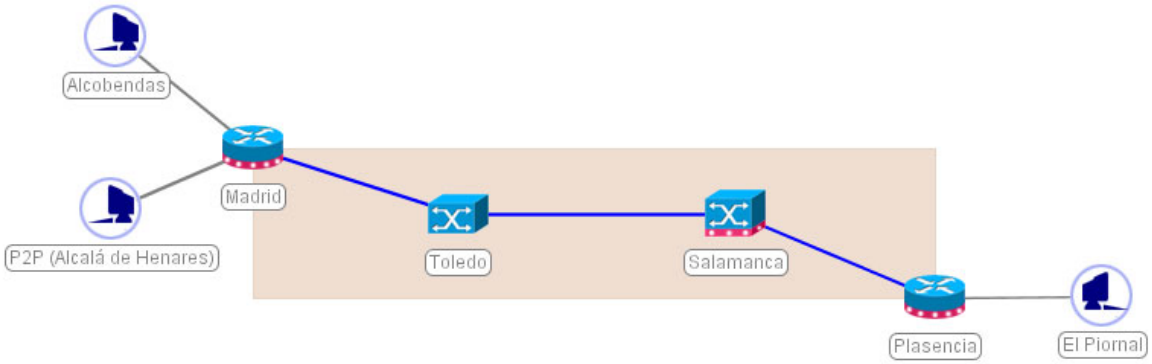
de los paquetes a los que hacen referencia ya no se encuentran en su DMGP; así que contesta a Plasencia con confirmaciones y retransmisiones de paquetes. Con 96 KB de DMGP y esta configuración se han recuperado el 79’54% de lo paquetes GoS descartados en Plasencia.

El aumento del número de pérdidas a medida que la GoS aumenta se ha justificado unos párrafos más arriba.

Como conclusión, tenemos que con DMGP de 32 KB. se recuperan el 0% de los paquetes, con DMGP de 64 KB, el 73’82% de los paquetes y con DMGP de 96 KB, el 79’54 (y el 0% de paquetes irrecuperables). Podemos afirmar que tras las pruebas, la propuesta de este PFC recupera paquetes y lo hace en mayor medida cuando el tamaño de la DMGP de los nodos a los que se solicita retransmisiones es mayor.

3.4.4. Escenario 4: efecto del tamaño de los paquetes en las recuperaciones

Para este caso, se ha simulado tres veces distintas el escenario, cambiando únicamente entre una vez y otra el tamaño de los paquetes generados en los emisores. Se intenta así estudiar el efecto que produce el tamaño de los paquetes en el número de paquetes que se consiguen recuperar localmente. Se ha utilizado el siguiente escenario.



El tráfico fluye de izquierda a derecha desde los nodos emisores hasta llegar al nodo receptor. La configuración de cada elemento es la siguiente:

ENLACES		
Identificación del enlace		Retardo
Origen del enlace	Destino del enlace	
Alcobendas	Madrid	3.000 ns.
P2P (Alcalá de Henares)	Madrid	1.000 ns.
Madrid	Toledo	1.000 ns.
Toledo	Salamanca	1.000 ns.
Salamanca	Plasencia	3.000 ns.
Plasencia	El Piornal	60.000 ns.

CONMUTADORES / ENCAMINADORES			
Nombre	Potencia conmutación	Tamaño del búfer	Tamaño DMGP
Madrid	10 Gbps.	100 MB.	1 KB.
Toledo	10 Gbps.	5 MB.	N/A
Salamanca	10 Gbps.	6 MB.	64 KB.
Plasencia	6 Gbps.	1 MB.	10.240 KB.

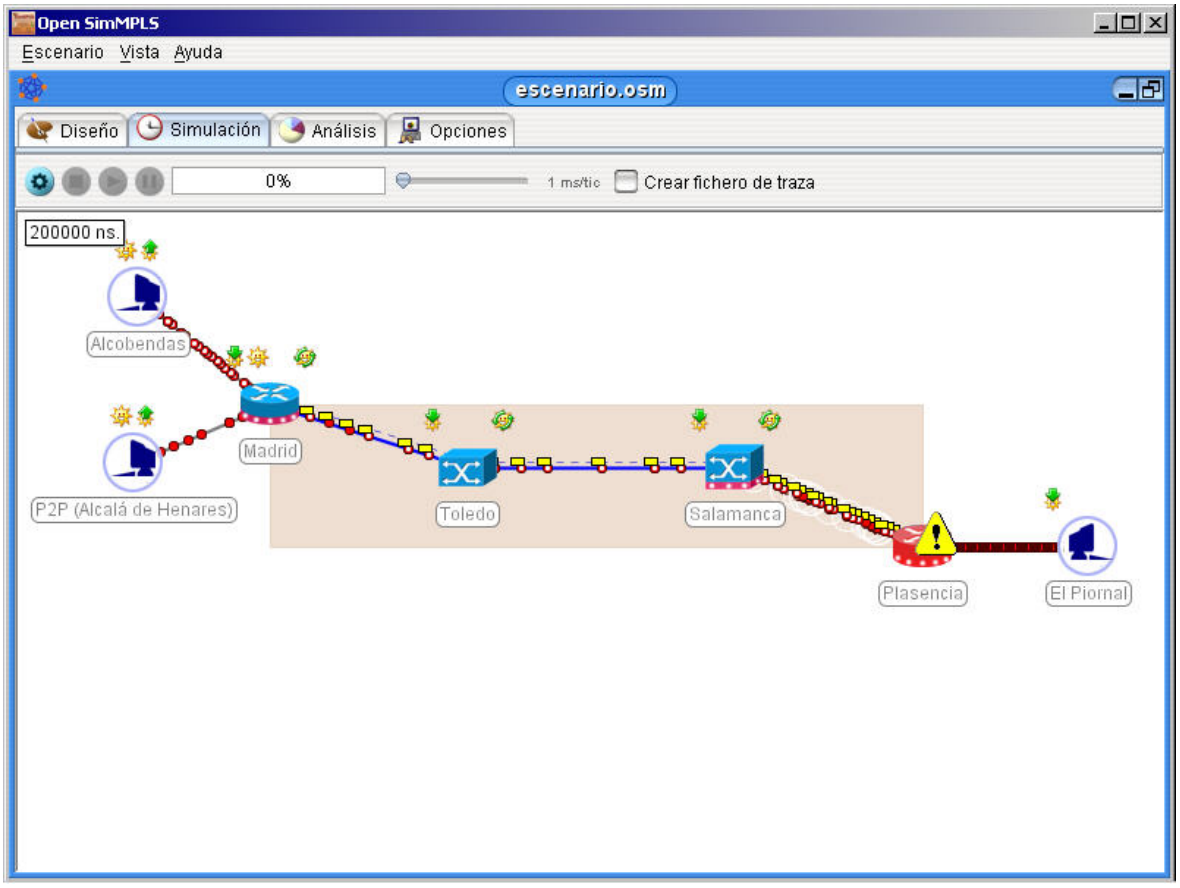
EMISORES						
Nombre	Tasa de generación	Tipo de tráfico	Carga útil de los paquetes	¿Sobre MPLS?	Nivel GoS	¿LSP de respaldo?
Alcobendas (sim. 1)	10 Gbps.	CONST.	10 B.	NO	3	NO
Alcobendas (sim. 2)	10 Gbps.	CONST.	100 B.	NO	3	NO
Alcobendas (sim. 3)	10 Gbps.	CONST.	1000 B.	NO	3	NO
P2P (Alcalá) (sim. 1)	10 Gbps.	CONST.	10 B.	NO	NO	NO
P2P (Alcalá) (sim. 2)	10 Gbps.	CONST.	100 B.	NO	NO	NO
P2P (Alcalá) (sim. 3)	10 Gbps.	CONST.	1000 B.	NO	NO	NO

RECEPTORES
En Open SimMPLS 1.0 los receptores son ideales. No se parametrizan.

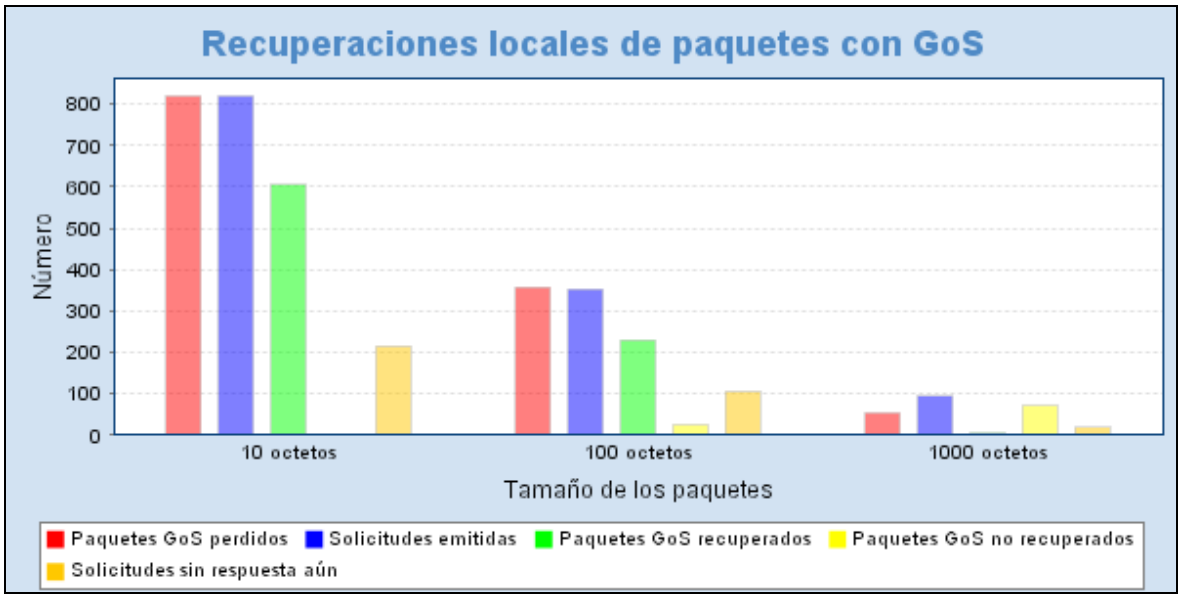
Y la configuración temporal de la simulación es como sigue:

Duración	Paso
200.000 ns.	100 ns.

El tráfico generado, como se puede observar, es idéntico salvo que uno está marcado con GoS y el otro no. Esto está hecho así a propósito para ver, en igualdad de condiciones, cómo se trata a un tráfico y a otro por parte de la red. Además, la DMGP de Madrid se ha establecido a 1 KB. Que en la práctica le impedirá resolver las peticiones que Salamanca no pueda resolver para Plasencia. Así estudiamos aisladamente el efecto del tamaño de los paquetes. En la figura siguiente observamos una captura de la pantalla del simulador durante la simulación de este escenario.



Y los datos que se han obtenido en Plasencia, relativos a las recuperaciones son los siguientes:



En el primer caso se producen 820 descartes de paquetes con GoS en Plasencia y para recuperarlos EPCD/GPSRP inician 820 solicitudes de retransmisión de paquetes. Al final de todo el proceso se consiguen recuperar 605 de los paquetes perdidos; ninguno se ha dado aún por irrecuperables y aunque aún quedan 215 peticiones por resolver, todo augura que se recuperarán. El proceso es el siguiente: por cada paquete que se pierde en Plasencia, se envía una solicitud de retransmisión a Salamanca. El reducido tamaño de los paquetes hace que se puedan almacenar en la DMGP de Salamanca un gran número de paquetes del flujo privilegiado y cuando le llegan las solicitudes, todos los paquetes a los que hacen referencia se encuentran en su DMGP; así que contesta a Plasencia con confirmaciones y retransmisiones de paquetes. Con paquetes de 10 octetos y esta configuración se han recuperado el 73,78% de lo paquetes GoS descartados en Plasencia.

En el segundo caso se producen 348 descartes de paquetes con GoS en Plasencia y para recuperarlos EPCD/GPSRP inician 353 solicitudes de retransmisión de paquetes. Al final de todo el proceso se consiguen recuperar 230 de los paquetes perdidos; 25 se han dado por irrecuperables y aún quedan 105 peticiones por resolver. El proceso es el siguiente: por cada paquete que se pierde en Plasencia, se envía una solicitud de retransmisión a Salamanca. Debido a que el tamaño de los paquetes no es excesivamente grande, Salamanca puede almacenar en su DMGP casi todos los paquetes necesarios para dar servicio a las peticiones procedentes de Plasencia, aunque en este caso, no son suficientes para dar un servicio del 100. Cuando le llegan las solicitudes de retransmisión, alguno de los paquetes a los que hacen referencia ya no se encuentran en su DMGP; así que contesta

a Plasencia con denegaciones y Plasencia debe volver a solicitar, esta vez a Madrid, la retransmisión de los paquetes. Madrid se ha configurado con 1 KB. De DMGP deliberadamente para que siempre conteste con negaciones. Así que, en este caso, todo paquete que no es capaz de recuperar Salamanca, tampoco se podrá recuperar por Madrid; por tanto todos los paquetes recuperados han sido gracias a retransmisiones de Salamanca. Con tamaño de paquetes de 100 octetos y esta configuración se han recuperado el 66,09% de lo paquetes GoS descartados en Plasencia.

En el tercer caso se producen 54 descartes de paquetes con GoS en Plasencia y para recuperarlos EPCD/GPSRP inician 94 solicitudes de retransmisión de paquetes. Al final de todo el proceso se consiguen recuperar 5 de los paquetes perdidos; 71 se han dado por irrecuperables y aún quedan 18 peticiones por resolver. El proceso es el siguiente: por cada paquete que se pierde en Plasencia, se envía una solicitud de retransmisión a Salamanca. Debido al gran tamaño de los paquetes (1000 octetos), Salamanca sólo puede almacenar en su DMGP unos pocos paquetes pertenecientes al flujo privilegiado. Cuando le llegan las solicitudes de retransmisión, la mayor parte de los paquetes a los que hacen referencia ya no se encuentran en su DMGP; así que contesta a Plasencia con denegaciones y Plasencia debe volver a solicitar, esta vez a Madrid, la retransmisión de los paquetes. Madrid se ha configurado con 1 KB. De DMGP deliberadamente para que siempre conteste con negaciones. Así que, en este caso, todo paquete que no es capaz de recuperar Salamanca, tampoco se podrá recuperar por Madrid; por tanto todos los paquetes recuperados han sido gracias a retransmisiones de Salamanca. Con tamaño de paquetes de 1000 octetos y esta configuración se han recuperado el 9,25% de lo paquetes GoS descartados en Plasencia.

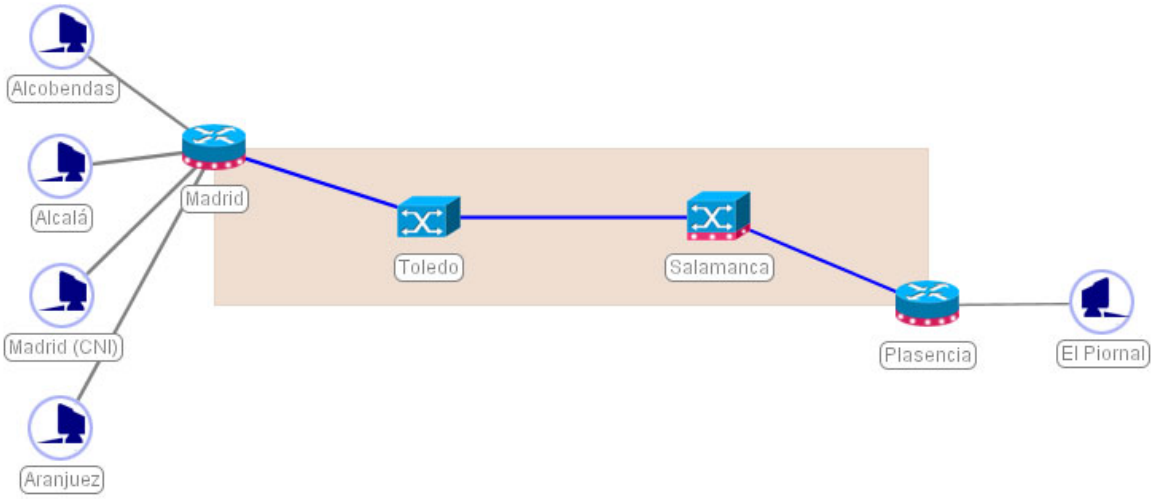
Como aclaración, comentar que la gran diferencia de pérdidas de una prueba a otra se debe a que se han dejado intactas las configuraciones de los nodos. Así, los emisores siguen poseyendo la misma tasa de generación de tráfico, pero con paquetes mayores, el número de paquetes totales generados decrece y con tamaños menores, se generan más paquetes. Más paquetes circulando por la red explican que haya más pérdidas.

Como conclusión, tenemos que con paquetes de 10 octetos, se recuperan el 73,78% de los paquetes, paquetes de 100 octetos, el 66,09% de los paquetes y con paquetes de 1000

octetos, el 9’25%. Podemos afirmar que tras las pruebas, la propuesta de este PFC recupera paquetes y lo hace en mayor medida cuando el tamaño de los paquetes es menor.

3.4.5. Escenario 5: efecto del nivel de GoS en el tratamiento de los paquetes

En este ejemplo se conectan cuatro generadores de tráfico con la misma capacidad y generando flujos idénticos. Los flujos generados sólo difieren en el nivel de GoS de cada uno de ellos. Los cuatro flujos tienen como destino el mismo receptor. Con este ejemplo queremos probar si el tratamiento que se da a los paquetes en el transcurso del camino hacia el destino depende del nivel de GoS de los mismos o no. Se ha utilizado el siguiente escenario.



El tráfico fluye de izquierda a derecha desde los nodos emisores hasta llegar al nodo receptor. La configuración de cada elemento es la siguiente:

ENLACES		
Identificación del enlace		Retardo
Origen del enlace	Destino del enlace	
Alcobendas	Madrid	1.000 ns.
Alcalá	Madrid	1.000 ns.
Madrid (CNI)	Madrid	1.000 ns.
Aranjuez	Madrid	1.000 ns.
Madrid	Toledo	1.000 ns.

Toledo	Salamanca	1.000 ns.
Salamanca	Plasencia	3.000 ns.
Plasencia	El Piornal	60.000 ns.

CONMUTADORES / ENCAMINADORES			
Nombre	**Potencia conmutación**	**Tamaño del búfer**	**Tamaño DMGP**
Madrid	10 Gbps.	100 MB.	1 KB.
Toledo	10 Gbps.	5 MB.	N/A
Salamanca	10 Gbps.	6 MB.	64 KB.
Plasencia	6 Gbps.	1 MB.	10.240 KB.

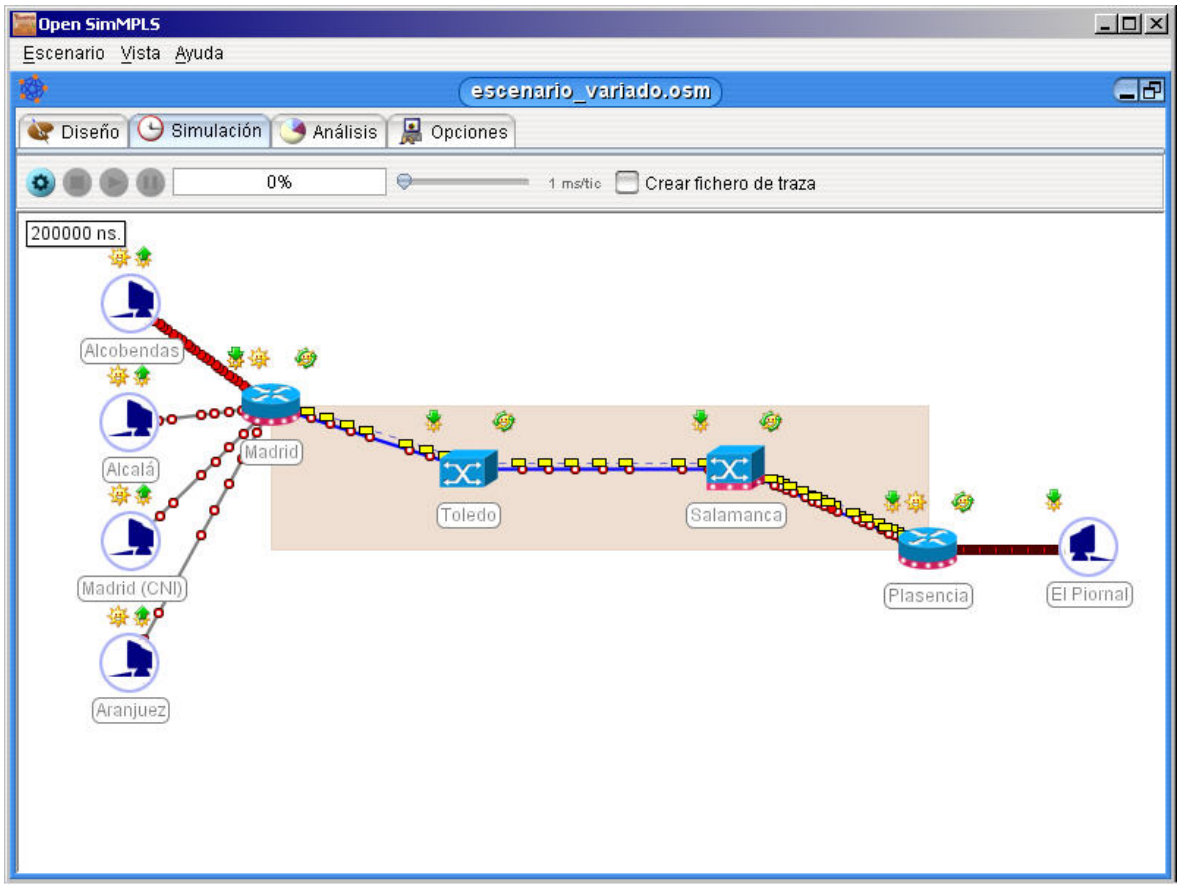
EMISORES						
Nombre	**Tasa de generación**	**Tipo de tráfico**	**Carga útil de los paquetes**	**¿Sobre MPLS?**	**Nivel GoS**	**¿LSP de respaldo?**
Alcobendas	10 Gbps.	CONST.	100 B.	NO	NO	NO
Alcalá	10 Gbps.	CONST.	100 B.	NO	1	NO
Madrid (CNI)	10 Gbps.	CONST.	100 B.	NO	2	NO
Aranjuez	10 Gbps.	CONST.	100 B.	NO	3	NO

RECEPTORES
En Open SimMPLS 1.0 los receptores son ideales. No se parametrizan.

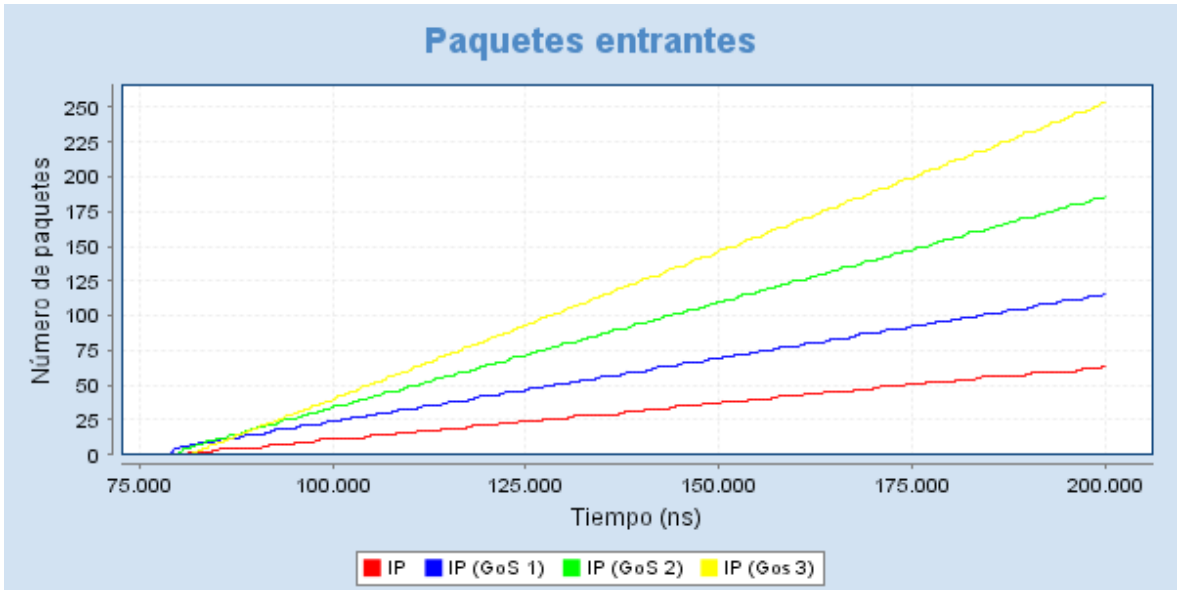
Y la configuración temporal de la simulación es como sigue:

Duración	Paso
200.000 ns.	100 ns.

En la figura siguiente observamos una captura de la pantalla del simulador durante la simulación de este escenario.

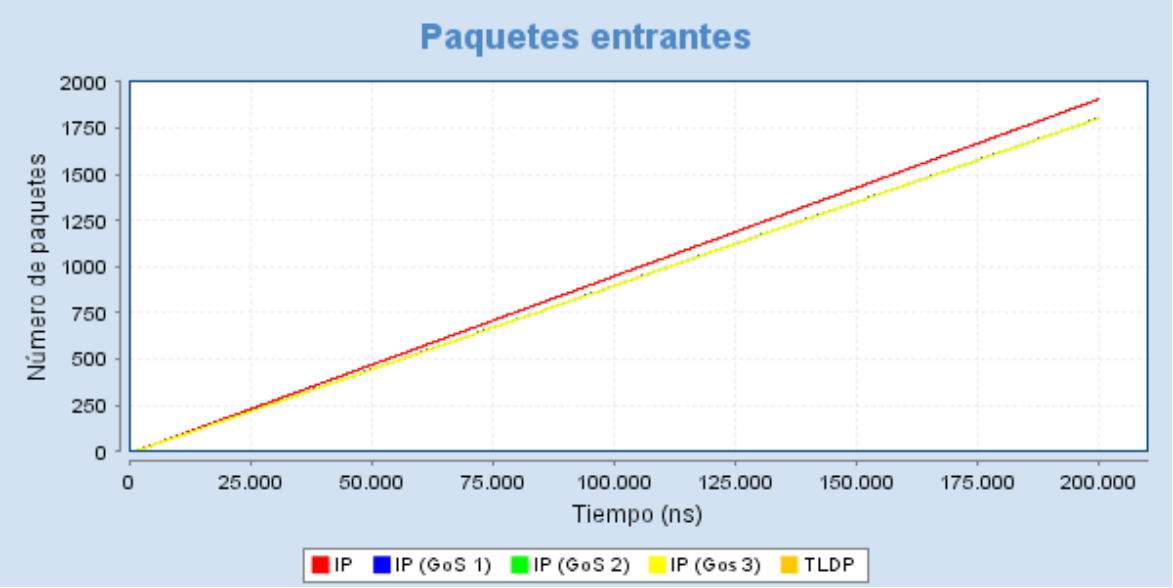


Y los datos que se han obtenido en El Piornal, relativos a los paquetes recibidos son los siguientes:



En el receptor, como se puede observar, la llegada de tráfico está estratificada. Los paquetes que han llegado en mayor cantidad son los paquetes IP GoS 3, seguidos por IP GoS 2, IP GoS 1 y por último IP sin GoS. En un ámbito normal, no sería esperable esta gráfica, puesto que el tráfico está siendo generado en forma idéntica; al receptor deberían llegar en la misma cantidad. Sin embargo esta es una de las características más importantes de los nodos activos, el algoritmo Round Robin con Prioridades (que une características de SFQ y CBQ) unido a la arquitectura de los puertos activos y el sistema de prioridades está pensado precisamente para que en los puertos activos se procesen más paquetes de aquellos tráficos con mayor prioridad y en el caso de IP o MPLS, los paquetes tienen mayor prioridad cuanto mayor es grado de GoS que portan. Por tanto, la gráfica anterior seria la esperada en un entorno donde existen estos cuatro tipos de gráficos y han de pasar por nodos activos.

Para quedar más convencidos, veamos la gráfica del tráfico entrante en Madrid, proveniente directamente de los cuatro emisores:



Las líneas correspondientes a IP GoS 1, IP GoS 2 e IP GoS 3 coinciden, por lo que aparecen como una línea amarilla con tintes de azul. Esto significa que han entrado el mismo número de paquetes de estos tres tráficos en Madrid; y de ahí hasta el receptor sólo hay un posible camino que han debido llevar todos estos paquetes. Con respecto al tráfico IP sin GoS, se ve que a Madrid llegan más paquetes que de tráfico marcado con GoS,

aunque muy poco más. Esto es debido a que todos los emisores están configurados para generar tráfico a la misma velocidad, con la misma tasa; Y aunque la carga útil de los paquetes se ha configurado en todos los emisores a 100 octetos, los paquetes IP marcados con garantía de servicio son unos octetos más grandes debido a que deben almacenar en el campo opciones los atributos correspondiente a GoS. No es mucho, pero es lo suficiente para que con a lo largo del tiempo, con igual tasa de generación de tráfico, se generen más paquetes IP sin GoS.

Como conclusión, podemos confirmar que simplemente por el hecho de que el tráfico esté marcado con distinto nivel de GoS, es tratado de forma distinta por los nodos activos: favoreciendo aquellos paquetes cuyos requerimientos de GoS son mayores, que serán procesados en mayor cantidad en los búferes de los puertos activos.

3.4.6. Conclusiones

Al comentar las investigaciones teóricas de las que partió este proyecto, dije que se había conseguido encontrar sobre el papel un conjunto de técnicas, protocolos y estructuras que en conjunto permitían que el tratamiento del tráfico marcado con GoS sea privilegiado, más cuanto más nivel de GoS se hubiese asignado, que era el objetivo inicial del proyecto; Dejé para este momento el comprobar si además de correctas teóricamente, las propuestas de este proyecto eran correctas en la práctica. Pues bien, a la vista de los resultados anteriores, creo que se puede decir tranquilamente que el proyecto en los requisitos, la teoría y la práctica cumple con lo que se esperaba de él. Funciona y da resultado a los problemas que se perseguía paliar.

3.5. Web del proyecto

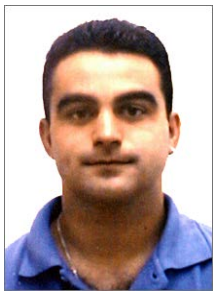
Desde el comienzo se ha pretendido que tanto la propuesta técnica presentada en este proyecto como el simulador de la misma - Open SimMPLS 1.0 - estuviesen liberados como documentación libre y como software libre, respectivamente. Además se ha pretendido siempre que tras la presentación del proyecto, este siguiese vivo. Para que esto

suceda, se ha creado una página web del Proyecto Final de Carrera, bilingüe, en la siguiente dirección:

- http://patanegra.unex.es/opensimmpls

En ella se podrán encontrar las últimas actualizaciones, se podrá colaborar el desarrollo del proyecto tras su presentación, se podrá consultar la documentación existente y se podrá descargar sin coste alguno todo el material existente. Además, desde ella se podrá poner en contacto con el autor del mismo (y con quien, eventualmente, pudiese hacerse cargo del proyecto).

4. Autor del proyecto



Manuel Domínguez Dorado <ingeniero@ManoloDominguez.com> nació en Zafra, provincia de Badajoz (Extremadura, ESPAÑA) en 1.977. Ha dedicado y dedica gran parte de su formación como Ingeniero en Informática e Ingeniero Técnico en Informática de Sistemas al estudio de las redes y protocolos avanzad@s de comunicación y a los grandes sistemas de software distribuido. Tiene especial predilección por el análisis de los flujos de información en los entornos empresariales y el diseño de soluciones de ingeniería sencillas para problemas informáticos complejos. Cree en la ingeniería como único método eficaz para conseguir software adecuado, robusto, de calidad, fiable y competitivo. Es miembro de la *Asociación de Ingenieros Informáticos de Extremadura* (AIIEx, http://webepcc.unex.es/aiiex), de la *asociación española de usuarios de Linux* (Hispalinux, http://www.hispalinux.es), es enlace del *ACM Crossroads student magazine* y ha pertenecido durante algunos años a otras asociaciones relacionadas con el software libre como GuPLe, GULEx y Sinuh. Desde hace algún tiempo y de forma continua trabaja escribiendo artículos para distintas publicaciones del sector informático. Hasta 2.004 ha creado más de 40 aplicaciones informáticas licenciadas como software libre y ha escrito más de 60 documentos calificados como documentos públicos (incluidos este Proyecto Final de Carrera y su documentación).

5. Agradecimientos

Quisiera agradecer, al término de mis estudios como Ingeniero Informático, a todas aquellas personas que me han ayudado a llevar a buen término mi carrera, en general, y concretamente este Proyecto Final de Carrera.

- En primer lugar, a mis padres, **Manolo** y **Agustina**, y al resto de mi **familia**; por su incondicional apoyo tanto moral como económico, incluso bajo circunstancias muy desfavorables, sin el cual ahora mismo no estaría a las puertas de ser ingeniero.

- En segundo lugar, a **Elena**, mi pareja desde hace… ¡muuuchos años! Al lado de la cual comencé esta andadura y al lado de la cual la termino; por el tiempo que ha sacrificado para que yo pueda desarrollar los estudios que quería. A nuestros **amigos** también, por acompañarla en tantas ocasiones en las que yo no he estado.

- En tercer lugar, a **José Luis**, mi director del Proyecto Final de Carrera. Principalmente por haberme dado muchísimo como persona; amistad, confianza y posibilidades de desarrollar mis ideas; por escucharme con atención, por aconsejarme en el ámbito personal y profesional y por haber saltado tantas veces ese límite que separa la relación profesor-alumno e introducirse en el terreno de la amistad. Y por ser un profesor que no sólo durante el Proyecto sino en las distintas asignaturas que me ha impartido (e incluso fuera de ellas) ha conseguido despertar en mí el afán de superación que tras los tres primeros años de carrera parecía haberse apagado. Porque no me ha puesto nunca nada fácil, porque no me ha dado nunca nada hecho y porque ha conseguido que hasta la última neurona de mi cerebro sude.

- En cuarto lugar, a todos los **profesores** que durante mi carrera han hecho que su labor investigadora se refleje en sus clases en pro de sus alumnos, trabajando con dedicación para ellos, preparando bien las clases, renovándose, creando la complicidad necesaria para que el ambiente universitario brote, poniendo todos los medios posibles al servicio de sus alumnos y sabiendo entender las particularidades de cada uno de ellos.

- Y por último a todos aquellos **autores** que han publicado sus trabajos de forma libre y gratuita en Internet. Sus aportaciones han supuesto el 90% de las fuentes bibliográficas que he consultado sin las cuales, sin duda, no hubiese podido realizar este proyecto.

6. Fuentes consultadas

Para la realización de este proyecto he tenido que consultar multitud de fuentes. A continuación se detalla una relación de todas ellas.

6.1. Bibliografía

6.1.1. Java

- ***"JSDK 1.4.1 API Documentation"***
  Sun Microsystems, 2003.

- ***"Java 2. Manual de referencia"***
  Herbert Schildt.
  Osborne – McGraw Hill, 2001.

6.1.2. MPLS

- ***"Multiprotocol Label Switching: enhancing routing the new public network"***
  Chuck Semeria.
  Juniper Networks, 1999.

- ***"Redes de banda ancha"***
  José Manuel Caballero.
  Marcombo, 1998.

- ***"Redes de computadoras, 3ª edición"***
  Andrews S. Tanenbaum.
  Pearson, 1997.

- ***"Alta velocidad y calidad de servicio en Redes IP"***
  Jesús García Tomás, José Luis Raya Cabrera y Víctor Rodrigo Raya.
  Ra-ma, 2002.

- ***"MPLS: an introduction to Multiprotocol Label Switching"***
  Nortel Networks, 2001.

- ***"Implementing Multiprotocol Label Switching with Altera PLD's"***
  Altera Corporation, 2001.

- ***"MPLS class of service"***
  Cisco, 2001.

- ***"Multiprotocol Label Switching (MPLS)"***
  International Engineering Consortium, 2000.

- ***"Network Simulator with MPLS"***
  Miguel González Martín, Ángel Bejarano Borrega.
  Escuela Politécnica de Cáceres, 2002.

- ***"A guide to MPLS"***
  Nortel Networks, 2001.

- ***"Solución de redes privadas virtuales sobre MPLS"***
  HP Networks, 2001.

- ***"LDP conformance implementation agreement"***
  MPLS Forum, 2002.

- ***"Análisis de la integración entre tráfico IP y redes ATM. Simulador MPLS"***
  Miguel Ángel Martín Tardío, Miguel Gaspar Rodríguez, José Luis González Sánchez.
  Escuela Politécnica de Cáceres, 2001.

- ***"MPLS: una arquitectura de backbone para la Internet del siglo XXI"***
  José barberá, Telia Iberia, 2001

- ***"MPLS Traffic Engineering"***
  Raj Jain.
  The Ohio State Univerity, 1999.

- ***"Traffic Engineering with Multiprotocol Label Switching"***
  Avici Systems, 2000.

- ***"Deply MPLS y MAN's"***
  Marc Lasserre.
  River Stone Networks, 2001.

- ***"Quality of service using traffic engineering over MPLS: an analysis"***
  Praveen Bhaniramka, Wei Sun, Raj Jain.
  The Ohio State University, 1999.

- ***"Redes IP multiservicio"***
  José Miguel Vargas González, Avelino Bellón Gutiérrez.
  Telefónica de España, 1999.

- ***"Routing in a MPLS network featuring pre-emption mechanisms"***
  François Blanchy, Laurent Mélon, Guy Leduc.
  University of Liège, 1999.

- ***"Providing quality of service in the Internet"***
  Xipeng Xiao.
  Michigan State University, 2000.

- ***"MPLS & GMPLS"***
  Li Yin.
  UC Berkeley, 2002.

- ***Los siguientes Request For Comments (RFC) del IETF:***
  - RFC 1483: *"Multiprotocol Encapsulation over ATM Adaptation Layer 5"*.
  - RFC 2547: *"BGP/MPLS VPNs"*.
  - RFC 2702: *"Requirements for Traffic Engineering Over MPLS"*.
  - RFC 2917: *"A Core MPLS IP VPN Architecture"*.
  - RFC 3031: *"Multiprotocol Label Switching Architecture"*.
  - RFC 3032: *"MPLS Label Stack Encoding"*.
  - RFC 3034: *"Use of Label Switching on Frame Relay Networks Specification"*.
  - RFC 3035: *"MPLS using LDP and ATM VC Switching"*.
  - RFC 3036: *"LDP Specification"*.
  - RFC 3037: *"LDP Applicability"*.
  - RFC 3038: *"VCID Notification over ATM link for LDP"*.
  - RFC 3063: *"MPLS Loop Prevention Mechanism"*.
  - RFC 3107: *"Carrying Label Information in BGP-4"*.
  - RFC 3209: *"RSVP-TE: Extensions to RSVP for LSP Tunnels"*.
  - RFC 3210: *"Applicability Statement for Extensions to RSVP for LSP-Tunnels"*.
  - RFC 3212: *"Constraint-Based LSP Setup using LDP"*.
  - RFC 3213: *"Applicability Statement for CR-LDP"*.
  - RFC 3214: *"LSP Modification Using CR-LDP"*.
  - RFC 3215: *"LDP State Machine"*.
  - RFC 3270: *"Multi-Protocol Label Switching (MPLS) Support of Differentiated Services"*.
  - RFC 3353: *"Overview of IP Multicast in a Multi-Protocol Label Switching (MPLS) Environment"*.
  - RFC 3443: *"Time To Live (TTL) Processing in Multi-Protocol Label Switching (MPLS) Networks"*.
  - RFC 3469: *"Framework for Multi-Protocol Label Switching (MPLS)-based Recovery"*.
  - RFC 3477: *"Signalling Unnumbered Links in Resource ReSerVation Protocol - Traffic Engineering (RSVP-TE)"*.
  - RFC 3478: *"Graceful Restart Mechanism for Label Distribution Protocol"*.
  - RFC 3479: *"Fault Tolerance for the Label Distribution Protocol (LDP)"*.

- RFC 3480: *"Signalling Unnumbered Links in CR-LDP"*.
- RFC 3612: *"Applicability Statement for Restart Mechanisms for the Label Distribution Protocol (LDP)"*.

- ***Los siguientes borradores (Drafts) del IETF:***
  - *"Graceful Restart Mechanism for BGP with MPLS"*.
  - *"Link Bundling in MPLS Traffic Engineering"*.
  - *"Multiprotocol Label Switching (MPLS) Traffic Engineering Management Information Base for Fast Reroute"*.
  - *"Multiprotocol Label Switching (MPLS) FEC-To-NHLFE (FTN) Management Information Base"*.
  - *"Multiprotocol Label Switching (MPLS) Forwarding Equivalence Class To Next Hop Label Forwarding Entry (FEC-To-NHLFE) Management Information Base"*.
  - *"Encapsulating MPLS in IP or Generic Routing Encapsulation (GRE)"*.
  - *"LDP DoD Graceful Restart"*.
  - *"Definitions of Managed Objects for the Multiprotocol Label Switching, Label Distribution Protocol (LDP)"*.
  - *"MTU Signalling Extensions for LDP"*.
  - *"Graceful Restart Mechanism for LDP"*.
  - *"Detecting MPLS Data Plane Failures"*.
  - *"Multi Protocol Label Switching Label Distribution Protocol Query Message Description"*.
  - *"Multiprotocol Label Switching (MPLS) Label Switching Router (LSR) Management Information Base"*.
  - *"Multiprotocol Label Switching (MPLS) Label Switching Router (LSR) Management Information Base"*.
  - *"Multiprotocol Label Switching (MPLS) Management Overview"*.
  - *"Definition of an RRO node-id subobject"*.
  - *"OAM Requirements for MPLS Networks"*.
  - *"Fast Reroute Extensions to RSVP-TE for LSP Tunnels"*.
  - *"MPLS Traffic Engineering Soft preemption"*.

- *"Definitions of Textual Conventions for Multiprotocol Label Switching (MPLS) Management"*.
- *"Definitions of Textual Conventions for Multiprotocol Label Switching (MPLS) Management"*.
- *"Improving Topology Data Base Accuracy with Label Switched Path Feedback in Constraint Based Label Distribution Protocol"*.
- *"Traffic Engineering Link Management Information Base"*.
- *"Multiprotocol Label Switching (MPLS) Traffic Engineering Management Information Base"*.
- *"Multiprotocol Label Switching (MPLS) Traffic Engineering Management Information Base"*.
- *"Time to Live (TTL) Processing in MPLS Networks"*.

6.1.3. GMPLS

- ***"Generalized Multiprotocol Label Switching: An Overview of Routing and Management Enhancements"***

  Ayan Banerjee, John Drake, Jonathan P. Lang, Brad Turner, Kireeti Kompella y Yakov Rekhter.

  IEEE Communications Magazine, Enero 2001.

- ***"Experiments in Fast Restoration using GMPLS in Optical / Electronic Mesh Networks"***

  Guangzhi Li, Jennifer Yates, Robert Doverspike y Dongmei Wang, 2001.

- ***"Generalized Multiprotocol Label Switching (GMPLS)"***

  International Engineering Consortium, 2001.

- ***"Generalized MPLS - Signaling Functional Description"***

  Lou Berger y Peter Ashwood-Smith, Julio de 2000.

- ***"GMPLS Configuration"***

  Juniper Networks, 2002.

- ***"Generalized Multiprotocol Label Switching"***
  Gavin Gandhi, Febrero 2002.

- ***"Generalized Multiprotocol Label Switching: An Overview of Signaling Enhancements and Recovery Techniques"***
  Ayan Banerjee, John Drake, Jonathan Lang, Brad Turner, Daniel Awduche Lou Berger, Kireeti Kompella y Yakov Rekhter.
  IEEE Communications Magazine, Julio 2001.

- ***"Generalized MPLS-Based Distributed Control Architecture for Automatically Switched Transport Networks"***
  Yangguang Xu, Patrice N. Lamy, Eve L. Varma, and Ramesh Nagarajan.
  Bell Labs Technical Journal, Enero - Julio 2001.

- ***Los siguientes Request For Comments (RFC) del IETF:***
  - RFC 3471: *"Generalized Multi-Protocol Label Switching (GMPLS) Signalling Functional Description"*.
  - RFC 3472: *"Generalized Multi-Protocol Label Switching (GMPLS) Signaling Constraint-based Routed Label Distribution Protocol (CR-LDP) Extensions"*.
  - RFC 3473: *"Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions"*.

- ***Los siguientes borradores (Drafts) del IETF:***
  - *"Generalized MPLS Signaling - Implementation Survey"*.
  - *"Generalized Multiprotocol Label Switching Extensions to Control Non-Standard SONET and SDH Features"*.
  - *"Generalized Multiprotocol Label Switching (GMPLS) Traffic Engineering Management Information Base"*.
  - *"Link Bundling in MPLS Traffic Engineering"*.
  - *"Generalized Multi-Protocol Label Switching (GMPLS) Signaling Constraint-based Routed Label Distribution Protocol (CR-LDP) Extensions"*.

- *"Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions"*.
- *"Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description"*.
- *"LSP Hierarchy with Generalized MPLS TE"*.
- *"OSPF Restart Signaling"*.

6.1.4. Conmutación óptica

- ***"Integración de IP sobre redes ópticas"***
  Alejandro Campoy Ruiz.
  Escola Universitària Politécnica de Mataró

- ***"Redes de banda ancha"***
  José Manuel Caballero.
  Marcombo, 1998.

- ***"Redes de computadoras, 3ª edición"***
  Andrews S. Tanenbaum.
  Pearson, 1997.

- ***"Alta velocidad y calidad de servicio en Redes IP"***
  Jesús García Tomás, José Luis Raya Cabrera y Víctor Rodrigo Raya.
  Ra-ma, 2002.

- ***"Evolution to All-optical switching"***
  Krishna Bala, 2001.

- ***"Comunicación óptica: pasado, presente y futuro"***
  Guadalupe Ortiz Bellot, 2003.

- ***"10 Gigabit Ethernet, Metro WDM, MPLS, MPLS Traffic Engineering"***
  Walter Dey.

29th Speedup Workshop on 29th Speedup Workshop on Distributed Computing and High Distributed Computing and High-Speed Networks.
Berne University Switzerland March, 2001

- ***"Multicasting in WDM Networks".***
  Jingyi He, S.-H. Gary Chan y Danny H. K. Tsang.
  IEEE Communications Surveys, 2002.

- ***"IP-Centric Control and Management of Optical Transport Networks".***
  Greg M. Bernstein, Jennifer Yates, Debanjan Saha.
  IEEE Communications Magazine, Octubre 2000.

- ***"Proyecto EMPIRICO: Anillo Metropolitano Gigabit Ethernet DWDM Configurable Dinámicamente Mediante un Plano de Control Óptico basado en GMPLS".***
  Raül Muñoz, Carolina Pinart, Gabriel Junyent, 2002.

- ***"Controlling Optical Networks".***
  Greg Bernstein, Bala Rajagopalan y Dan Spears.
  Optical Internetworking forum (OIF), Octubre 2002.

- ***"Creating Convergence Between the IP and the Optical Layers".***
  Krishna Bala.
  Tellium, 2000.

- ***"Comunicación óptica".***
  Yolanda Sánchez Muñoz. Ana María Luna Bueno, 2.003.

- ***"Transport Networks & Technologies".***
  John Strand.
  AT&T, 2000.

6.1.5. Redes activas y sistemas multiagente

- ***"Protocolo Activo para Transmisiones Garantizadas sobre una Arquitectura Distribuida y Multiagente en Redes ATM".***
  José Luis González-Sánchez, Mayo de 2001.

- ***"Más allá de las redes de nueva generación: últimos avances en el campo de las redes activas".***
  Pedro Andrés Aranda Gutiérrez.
  Telefónica I+D, Noviembre 2001.

- ***"Sistemas multiagente para la asignación de frecuencias en redes celulares".***
  Francesc Comellas y Javier Ozón.
  AT&T, 2000.

- ***"Providing Authentication & Authorization Mechanisms for Active Service Charging".***
  Marcelo Bagnulo, Bernardo Alarcos, María Calderón, Marifeli Sedano, 2002.

- ***"Seguridad En Redes Activas".***
  Alberto Díaz Freire, 2002.

- ***"Interacción entre Agentes: Utilidad, Coaliciones y Negociación".***
  Mª Victoria Belmonte, José Luis Pérez de la Cruz, Francisco Triguero, 2003.

- ***"Agentes en Internet".***
  Daniela Godoy, 2002.

- ***"Agentes móviles. Aplicaciones telemáticas".***
  Roberto Tejero Ujado, 2002.

- ***"Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents".***
  Stan Franklin y Art Graesser.
  Proceedings of the Third International Workshop on Agent Theories, Architectures,

and Languages, Springer-Verlag, 1996.

- ***"Desarrollo de Sistemas Multi-Agentes".***
  Analía Amandi
  Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.13 (2001).

- ***"Report on Active Network Activity".***
  John Guttag, Marzo 1997.

- ***"Active network vision and reality: lessons from a capsule-based system".***
  David Wetherall.
  17th ACM Symposium on Operating Systems Principles (SOSP '99).

- ***"Service Introduction in an Active Network".***
  David J. Wetherall.
  Degree of Doctor of Philosophy (MIT), 1999.

- ***"Implementation of an Active Networking Architecture".***
  Samrat Bhattacharjee, Kenneth L_ Calvert, Ellen W_ Zegura, 2003.

- ***"AntNet Routing Algorithm for Data Networks based on Mobile Agents".***
  Benjamín Barán y Rubén Sosa.
  Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No. 12 (2001).

- ***"Self_Organizing Wide_Area Network Caches".***
  Samrat Bhattacharjee, Kenneth L_ Calvert, Ellen W_ Zegura, 2003.

- ***"Towards an Active Network Architecture".***
  David L. Tennenhouse y David J. Wetherall.
  Keynote session of Multimedia Computing and Networking, San Jose, CA, January 1996.

- ***"Metodologías orientadas a agentes".***

Carlos A. Iglesias, Mercedes Garijo, José C. González, 2003.

- ***"Directions in Active Networks".***
  Kenneth L_ Calvert y Samrat Bhattacharjee Ellen Zegura, 2003.

- ***"Active Agents Algorithm for Differentiated Services in Networks with Bursty Traffic".***
  Tibor Gyires.
  CSNDSP, 2002.

- ***"Inteligencia artificial distribuida en entornos de tiempo real".***
  Emilio Vivancos, Luis Hernández, Vicente Botti, 2003.

- ***"Módulo de Planificación en Sistemas Multiagente para dominios normados".***
  Cora B. Excelente y Christian Lemaître.
  ENC'97 proceedings.

- ***"Multicast Congestion Control for Active Network Services".***
  Arturo Azcorra, María Calderón, Marifeli Sedano y José Ignacio Moreno.
  ETT'99.

- ***"Network Support for Multicast Video Distribution".***
  Samrat Bhattacharjee, Kenneth L_ Calvert y Ellen W_ Zegura,
  GIT-CC 98/16.

- ***"Performance of Application Specific Buffering Schemes for Active Networks".***
  Samrat Bhattacharjeey y Martin W. McKinnon, 2003.

- ***"An Architecture for Active Networking".***
  Samrat Bhattacharjee, Kenneth L. Calvert, Ellen W. Zegura, 2003.

- ***"Active Networking and the End_to_End Argument".***
  Samrat Bhattacharjee, Kenneth L. Calvert, Ellen W. Zegura, 2003.

- ***"Reasoning About Active Network Protocols".***
  Samrat Bhattacharjee, Kenneth L. Calvert, Ellen W. Zegura, 2003.

- ***"Active Networks Support for Multicast Applications".***
  María Calderón, _Marifeli Sedano, _Arturo Azcorra, Cristian Alonso
  IEEE Networks Magazine, 2003.

- ***"A Survey of Active Network Research".***
  David L. Tennenhouse, Jonathan M. Smith, W. David Sincoskie, David J. Wetherall, Gary J. Minden.
  IEEE Communications Magazine, 1997.

- ***"Performance of Active Multicast Congestion Control".***
  Marifeli Sedano, Arturo Azcorra, María Calderón.
  IWAN'00.

- ***"Caracterización de los enlaces de Internet utilizando tecnología deRedes Activas".***
  Marifeli Sedano, Bernardo Alarcos, María Calderón y David Larrabeiti.
  JITEL'01.

- ***"Principios y Aplicaciones de las Redes Activas".***
  María Calderón Pastor, Marifeli Sedano Ruiz y Santiago Eibe García.
  JITEL'99.

- ***"Metodologías para el desarrollo de sistemas multi-agente".***
  Jorge J. Gómez Sanz
  Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.18 (2003).

- ***"Estudio de métodos de desarrollo de sistemas multiagente".***
  Vicente J. Julián, Vicente J. Botti, 2003.

- ***"Mobile Agents and Active Networks: Complementary or Competing Technologies?".***
  Martin Collier
  Technical report of the Broadband Switching & Systems Laboratory, 1999.

- ***"Next Century Challenges: RadioActive Networks".***
  Vanu Bose, David Wetherall y John Guttag.
  MOBICOM'99.

- ***"Los Sistemas MultiAgente para el modelado de la actuación en organizaciones humanas".***
  Enrique Paniagua Arís, José Tomás Palma Méndez, Fernando Martín Rubio, 2001.

- ***"Redes Activas con IPv6".***
  D. Larrabeiti, M. Calderón, A. Azcorra, A. García y J. E. Kristensen.
  COMWORLD'01.

- ***"Ad Hoc Networks with Active Technology: A Synthesis Study".***
  Alex Galis, Alvin Tan, Joan Serrat y Julio Vivero, 2001.

- ***"ROSA: Realistic Open Security Architecture for active networks".***
  Marcelo Bagnulo, Bernardo Alarcos, María Calderón, Marifeli Sedano.
  IWAN'02.

- ***"The Price of Safety in an Active Network".***
  D. Scott Alexander, Paul B. Menage, Angelos D. Keromytis, William A. Arbaugh, Kostas G. Anagnostakis, and Jonathan M. Smith, 2001.

- ***"An Active Network Approach to Support Multimedia Relays".***
  Manuel Urue~na1, David Larrabeiti, Marría Calderón, Arturo Azcorra, Jens E. Kristensen, Lars Kroll Kristensen, Ernesto Exposito, David Garduno y Michel Diaz.
  PROMS'02.

- ***"Software Agents: An Overview".***
  Hyacinth S. Nwana, 2001.

- ***"Secure Quality of Service Handling: SQoSH".***
  D. Scott Alexander, William A. Arbaugh, Angelos D. Keromytis, Steve Muir y Jonathan M. Smith.
  IEEE Communications Magazine, Abril 2000.

- ***"Tésis doctoral: definición de una metodología para el desarrollo de sistemas multiagentes".***
  Carlos Ángel Iglesias Fernández, 1998.

- ***"Liquid Software: A New Paradigm for Networked Systems".***
  John Hartman, Udi Manber, Larry Peterson y Todd Proebsting.
  Téchnical Report 96/11.

- ***"An Adaptive Network Routing Strategy with Temporal Differences".***
  Yván Túpac Valdivia, Marley M. Vellasco, Marco A. Pacheco, 2001.

- ***"RAP: Protocol for Reliable Transfers in ATM Networks with Active Switches".***
  José Luis González-Sánchez y Jordi Domingo-Pascual, 1999.

- ***"SwitchWare: Accelerating Network Evolution (White Paper)".***
  J. M. Smith, D. J. Farber, C. A. Gunter, S. M. Nettles,D. C. Feldmeier y W. D. Sincoskie.
  White paper, 1996.

- ***"Una plataforma abierta para la investigación: Software de enrutadores".***
  Carolina Franco Espinosa, 2003.

- ***"Active Network Encapsulation Protocol (ANEP)".***
  D. Scott Alexander, Bob Braden, Carl A. Gunter, Alden W. Jackson, Angelos D. Keromytis, Gary J. Minden, David Wetherall, 1997.

- ***"Active Network Overlay Network (ANON)".***

- ***"COPS client-type for Active Networks".***
  Yacine El Mghazli y Olivier Marce, 2002.

6.1.6. Otros

- ***"Dirección y gestión de proyectos: un enfoque práctico"***
  Alberto Domingo Ajenjo.
  Ra-ma, 2000.

6.2. Internet

6.2.1. Java

- ***Web oficial de la tecnología java.***
  http://java.sun.com.

- ***Web oficial de JavaHispano.***
  http://www.javahispano.org.

- ***Web de programación en Castellano.***
  http://www.programacion.com.

6.2.2. MPLS

- ***Centro de recursos MPLS.***
  http://www.mplsrc.com.

- ***Forum MPLS.***
  http://www.mplsforum.org

- ***Consorcio Internacional de Ingeniería.***
  http://www.iec.org/online/tutorials/mpls

- ***Conferencia Internacional sobre MPLS, 2004.***
  http://www.isocore.com/mpls2004

- ***Web del IETF.***
  http://www.ietf.org

6.2.3. GMPLS

- ***Centro de recursos GMPLS.***
  http://www.polarisnetworks.com/gmpls

- ***Consorcio Internacional de Ingeniería.***
  http://www.iec.org/online/tutorials/gmpls

- ***Web del IETF.***
  http://www.ietf.org

6.2.4. Conmutación óptica

- ***Web oficial de Cisco System.***
  http://www.cisco.com

- ***Consorcio Internacional de Ingeniería.***
  http://www.iec.org/online/tutorials/opt_switch

- ***Web del IETF.***

http://www.ietf.org

6.2.5. Redes activas y sistemas multiagente

- ***Red académica española; REDIRIS.***
  http://www.rediris.es

- ***Página sobre redes activas del MIT.***
  http://www.sds.lcs.mit.edu/activeware

- ***Páginas web sobre sistemas multiagentes de la Universidad de Vigo.***
  http://agentes.ei.uvigo.es

6.2.6. Otros

- ***Grupo de News.***
  es.comp.ingenieria.software

7. Herramientas utilizadas

Para la creación del simulador Open SimMPLS he utilizado una gran cantidad de herramientas. El listado completo se puede ver bajo estas líneas.

- **NetBeans IDE 3.6**
  Completo entorno de programación para Java. Software libre, gratuito y multiplataforma.

- **Java 2 Software Development Kit 1.4.1**
  Conjunto de desarrollo de Sun Microsystems para el desarrollo de aplicaciones multiplataforma.

- **Photoshop 7.1**
  Programa de retoque fotográfico de Adobe, para Windows.

- **The GIMP 1.2.3**
  Programa de retoque fotográfico creado por Spencer Kimball y Meter Mattis, para Linux y Windows.

- **WinCVS 1.3.13.2**
  Cliente de CVS creado por Karl-Heinz Truenen, para Windows.

- **gCVS 1.0**
  Cliente de CVS creado por Alexandre Parenteau, para Linux.

- **XML Spy 5.0 Enterprise Edition**
  Software para la generación de documentos XML, XML Schema, XSL y otros, de Altova. Para Windows.

- **Word XP**
  Procesador de textos de Microsoft para Windows.

- **PDFCreator 0.7.2**

  Impresora virtual para la generación de documentos PDF, creado por Philip Chinery. Software libre y gratuito para Windows.

- **Microangelo 5.5**

  Editor de Iconos, de Impact Software, para Windows.

- **SnagIt 6.0**

  Capturador de pantalla de TechSmith, para Windows.

- **PowerPoint XP**

  Software de diseño de presentaciones, de Microsoft. Para Windows.

- **Dreamweaver MX 2.004**

  Software para la creación de sitios web, de Macromedia. Para Windows.

- **Nero Burning ROM 5.5.10.28**

  Software para la grabación de CD, de Ahead Software. Para Windows.

- **Acrobat Reader 5.5**

  Software para el visionado de documentos PDF, de Adobe. Para Windows.

- **KSnapshot 1.01.00**

  Software de captura de pantalla de KDE, creado por Richard M. Moore. Software libre y gratuito. Para Linux.

- **SimSwitch**

  Simulador de la implementación del protocolo TAP en ATM. Proyecto final de carrera de Juan Luis Vidal Arrojo y Guillermo Paniagua López-Ibarra.

- **TextPad 4.6.2**

  Procesador de textos, de Helios Software Solutions. Para Windows.

- **KWrite 4.0**

  Procesador de textos para KDE, creado por Christoph Cullmann. Software libre y gratuito para Linux.

- **Open Office.org Writer 1.0.1**

  Procesador de textos de Sun Microsystems. Software Libre y gratuito para Linux y Windows.

- **FindBugs 0.8.2**

  Analizador de código Java para búsqueda de patrones desencadenantes de fallos en la aplicación final. Está creado por William Pugh y David Hovemeyer, profesores en la Universidad de Maryland. Es software libre bajo licencia LGPL 2.1 con versiones para Windows y Linux entre otros.

8. Licencia de documentación libre de GNU

GNU Free Documentation License
Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine- generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non commercially, provided that this License, the copyright notices, and the license notice saying this License appliesto the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to live them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which

should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalents are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or simply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number.
Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other Documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this

License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later

version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.