Escola Politècnica Superior
d'Enginyeria de Vilanova i la Geltrú
UNIVERSITAT POLITÈCNICA DE CATALUNYA

PROJECTE FI DE CARRERA

TÍTOL:

Simulador para redes ópticas semitransparentes

AUTOR: Jordi Ferrer Martí

TITULACIÓ: E.T.T especialitat sistemes electrònics

DIRECTOR: Sergio Sánchez López

DEPARTAMENT: Arquitectura de computadors

DATA: 8 de febrer de 2008

TÍTOL:
Simulador para redes ópticas semitransparentes

COGNOMS:Ferrer Martí **NOM: Jordi**

TITULACIÓ: Enginyeria Tècnica Telecomunicació

ESPECIALITAT:Sistemes eletrònics **PLA: 95**

DIRECTOR: Sergio Sánchez López

DEPARTAMENT: Arquitectura de computadors

QUALIFICACIÓ DEL PFC

TRIBUNAL

PRESIDENT J.V. Castells SECRETARI Vicenç Peris VOCAL Jordi Fornés

DATA DE LECTURA:08 de Febrer de 2008

Aquest Projecte té en compte aspectes mediambientals: □ Sí □ No

PROJECTE FI DE CARRERA

RESUM (màxim 50 línies)

El crecimiento del tráfico de datos en las redes, ha provocado la búsqueda de nuevas tecnologías par poder satisfacer esa demanda.
Las redes de fibra óptica tienen la característica de ofrecer un gran ancho de banda y todos los esfuerzos van dirigidos hacia esa tecnología.
Por tanto los operadores de telecomunicaciones necesitan, entre otras cosas, de herramientas para poder desplegar las redes troncales de fibra óptica.
En este proyecto ofrecemos un simulador que ayuda a la planificación en el despliege de una red óptica, ya que nos da a conocer su comportamiento y tomar decisiones de ingeniería.
La arquitectura de red que predomina en las OTN's (redes ópticas de transporte) es DWDM, que consiste en la multiplexación de longitudes de onda en una fibra.
DWDM es un plano de transporte eficaz que nos puede dar un ancho de banda para comunicaciones con un límite casi desconocido, pero necesitamos de un plano de control que lleve la gestión de esa red. EL protocolo que se encaragará de ellos es GMPLS/MPLS.
Por otro lado los dispositivos que ejecutarán las funciones en una OTN son los Óptical Cross-Connect (OXC) y nos permiten conmutar *lamdas*.
En un principio todo el control y gestión de la OTN se hacia en el dominio eléctrico, llamada redes opacas, pero los investigadores tratan de desarrollar tecnología para que todo el control se haga también en el dominio óptico y tener redes transparentes.
En este momento estamos a mitad de camino. Esto nos da que parte del control de la red, como la conmutación si se hace en el dominio óptico, pero la gestión de protocolos de "routing" aún es en el dominio eléctrico, éstas son las redes ópticas semitransparentes

Paraules clau (màxim 10):

Long. de onda	Fibra óptica	Llamadas	Protocolo
Redes troncales	Enrutamiento	Erlangs	OXC
Planificación	"Lightpath"		

Contenido

Capítulo 1

Estado del arte.

1.1. Introducción.

A través de este trabajo vamos a estudiar las redes ópticas troncales, su estado actual, su importancia en el mundo de las comunicaciones y hacia donde evolucionan.
Los diseñadores de las primeras redes así como los operadores que las construyen poco imaginaban que las necesidades de interconectar dispositivos crecerían tanto. El crecimiento de Internet, gracias al gran impulso de las empresas y de las necesidades de la gente, ha sido muy grande. Dichas necesidades han ido creciendo desde las primeras ideas de interconectar ordenadores para intercambiar ficheros y datos, hasta las necesidades actuales que nos llevan a ocupar las redes con datos, voz e imágenes, que incrementan exponencialmente la capacidad que tienen que ofrecer una red troncal. Otro hecho que ha provocado la gran demanda de capacidad de la Red ha sido el hecho que los equipos cada vez han rebajado más y más sus precios con lo cual, esta tecnología cada vez ha sido más accesible a más y más gente. La suma de demanda de servicios más el incremento de equipos hace que el crecimiento sea mayor cada vez, esto también añadido al hecho que en el siglo XXI las tecnologías que interconectan a los usuarios a la Red no sólo se registran en el primer mundo sino que cada vez más llegan a países en vías de desarrollo.
El mayor grado de exigencia de las aplicaciones unido al gran número de usuarios ha planteado un serio problema a los proveedores de servicios, son necesarias redes de transporte más rápidas y con mayor capacidad. A partir de este punto las redes ópticas toman una importancia grande. Este tipo de redes ofrecen una gran capacidad y gran velocidad de transmisión, hasta el punto que las redes antiguas de cobre no podrían nunca igualarlas, ni siquiera tecnologías inalámbricas o de microondas. La única desventaja es que la tecnología óptica continúa a un nivel económico bastante alto.
Las redes ópticas de primera generación surgieron para sustituir el cobre físicamente, por tener un ancho de banda y un alcance también mayor sin necesidad de componentes activos (amplificadores, por ejemplo). De todas maneras la conmutación y el procesado de las tramas

de bits se hace aún en el dominio eléctrico. Un primer ejemplo de este tipo de redes fueron las redes FDDI, dentro de la conocida capa OSI estas primerizas redes ópticas están situadas en el nivel la capa física.

El desarrollo tecnológico no ha parado y se han conseguido redes ópticas de segunda generación llamadas WDM o redes de multiplexación por división de *lamdas*. El gran cambio que incorporan estas redes de segunda generación es la posibilidad de incorporar todo el proceso de conmutación dentro del dominio óptico, y es a este nivel donde se encuentra el desarrollo actual.

Las redes de comunicaciones se iniciaron, de un principio, con redes totalmente opacas, que significa que todos los procesos de la red se encontraba en el dominio eléctrico, simplemente el soporte por los que corrían los datos era óptico, hasta llegar a redes totalmente transparentes en que todo el comportamiento de una red troncal se encuentra en el dominio óptico. Estas redes de segunda generación han sido diseñadas para trabajar con el concepto de *lightpath* o camino óptico, que será ofrecido a las capas superiores y gestionado por el plano de control.

Concepto *lightpath*: los caminos ópticos son circuitos virtuales creados sobre una longitud de onda entre dos nodos, uno origen y otro destino, por donde se cursará una llamada.

Son los operadores de servicios de comunicaciones los que aplicarán estas tecnologías y las pondrán al servicio de sus clientes.

Los operadores de comunicaciones, principales interesados en el desarrollo de tecnologías de redes troncales, cada vez tienen procesos de planificación más precisos a la hora de desplegar nuevas redes. Es necesario cuantificar el volumen y calidad de servicio que se quiere entregar, los recursos de que disponemos y como se va a desarrollar la inversión. Para ello primero habrá un estudio de mercado, una búsqueda de los recursos que nos ofrecen los fabricantes de tecnologías y para acabar, según las partidas presupuestarias, como se irán aplicando los planes a la red. En toda planificación siempre va implícita una previsión de crecimiento futuro, incluida en el estudio de mercado.

De aquí que nos ayudemos de herramientas de simulación para ajustar mejor el despliegue de una red óptica.

Estas herramientas nos darán la posibilidad de hacer una previsión y comprobar su rendimiento. Lo más importantes de los simuladores es que nos permiten probar posibles errores en nuestro diseño sin poner en riesgo ni recursos materiales ni económicos. (En algunos casos humanos, no es el nuestro).

1.2 Objetivos y justificación:

Para estudiar como se comportan las redes ópticas cuando éstas empiezan a estar cargadas con tráfico, es necesario un simulador que consigue tratar con las variables más relevantes de una red óptica. Nuestro simulador nos da la posibilidad de estudiar diferentes parámetros dentro de las redes ópticas WDM y poder descubrir su comportamiento.
La posibilidad de variar los parámetros del simulador nos dará resultados que los operadores podrán aplicar a sus planificaciones, incluso pueden saber cuales son los mejores parámetros aplicables a los dispositivos que actúan en su red.
El trabajo sobre nuestro simulador empieza con la creación de un protocolo de "routing" para buscar los caminos existentes entre un origen y un destino, para luego elegir uno de ellos. Diseñaremos un protocolo de "routing", basándonos en el OSPF (Open Shortest Path Fist) RFC 1247, pero con varias puntualizaciones orientadas a las características de una red óptica.
Después la selección de una *lamda* libre en el cada uno de los enlaces nos dará como resultados el camino óptico para establecer el curso de llamadas. Una vez acabada la llamada la liberamos y la dejamos disponible para llamadas futuras. Más tarde en nuestro simulador incorporaremos sistemas con regeneración (conmutación) de *lamdas*.
Este último apartado nos dará la posibilidad de estudiar la mejor posición de nuestros conmutadores dentro de la topología de la red.
Para acabar usaremos el simulador para comprobar el comportamiento de una red semitransparente, utilizando para ello unas topologías de redes reales como modelos.
Es imprescindible actualmente para un operador, poder simular el resultado que obtendrán a la hora de desplegar su red. Podemos analizar con el simulador que protocolo utilizar para elegir el mejor camino óptico, como darle más inteligencia a la red para mejorar esta elección y por ultimo localizar los cuellos de botella que tenemos dentro de nuestra red.

1.3 Actualidad:

Las "handicap" de la mayoría de redes de comunicaciones que están construidas por los operadores actualmente, limitan el crecimiento, la versatilidad y adaptabilidad del equipamiento y la gestión del mismo. Cualquiera de estas prestaciones se puede obtener, pero a un coste elevado y en procesos muy lentos. Esto se debe a diversos factores: existen múltiples tipos arquitecturas de redes dependiendo del tipo de servicio y siempre son complejas, por eso la reubicación y ampliación de ancho de banda es muy lenta y costosa incluso a veces imposible.
La evolución natural de las redes ópticas exige que las nuevas redes sean adaptables al tipo y clase de servicio, den integración ("grooming") a las diferentes tecnologías ya existentes (IP,

ATM, SONET/SDH), que el ancho de banda bajo demanda pueda ser proporcionado de manera rápida y eficiente, que sean redes de alta disponibilidad, de mayor capacidad y flexibilidad, de crecimiento sencillo y con una perspectiva de red global.

Las recomendaciones que se pueden extraer de la evolución actual y los estudios simulados concluyen, que la solución son: las redes malladas; interconectadas por nodos de conmutación óptica reconfigurables dinámicamente y gestionadas por un plano de control automatizado, que aporta la inteligencia a la red.

Las redes ópticas semitransparentes es la solución intermedia, ya que por el punto de evolución tecnológica en la que nos encontramos en la actualidad aún es imposible poder trabajar con redes completamente ópticas. Esto quiere decir que para poder tratar el tráfico a través de una red óptica en algún momento tenemos que pasar la información de óptico a eléctrico y viceversa.

El elemento clave en este tipo de redes es un conmutador óptico que sea capaz de conmutar circuitos ópticos de manera dinámica, según las necesidades y requerimientos de las llamadas entrantes a la red.

Concretando:

Estado actual: Las redes ASON (Automatic Switched Optical Networks, ITU-T G.8080) son la evolución natural de las redes ópticas de transporte (OTN) actuales. En ellas conviven múltiples protocolos. Existe un tipo de red para cada tipo de servicio. Las redes clásicas, no malladas, normalmente anillos, son de aprovisionamiento mucho más complejo, son más rígidas, requieren más recursos de red y son de difícil crecimiento e interoperabilidad. Estos factores determinan que el coste del hardware de red y los costes operacionales sean sensiblemente superiores.

El mañana: Las redes del mañana deben tener mayor capacidad y flexibilidad, deben poder reubicar ancho de banda de un modo rápido y eficiente, deben ser redes multiservicio y deben ser redes adaptables a cada tipo y clase de servicio. Finalmente, deben ser concebidas con un carácter global.

Capítulo 2

Redes ópticas de transporte (OTN)

2.1 Evolución

Como clasificación general se pueden encontrar redes ópticas pasivas (ITU-T recommendations G.983.3, G.983.4 & G.983.5) y activas. En las redes pasivas solo utilizaban dos *lamdas* una para transmisión y otra para la recepción (G.983.3). Aunque en el nuevo borrador (G.983.4 & 5) se mejora el ajuste de ancho de banda asignado a los clientes frente al fijado antes, con lo cual se puede ajustar a la demanda, se incrementan las *lamdas* utilizadas hasta ahora con una nueva ventana, en la fig.2.1 pueden verse las ventanas de trabajo, que permitirá asignar a alguna de estas *lamdas* servicios específicos y otras a servicios de broadcasting y multicasting.

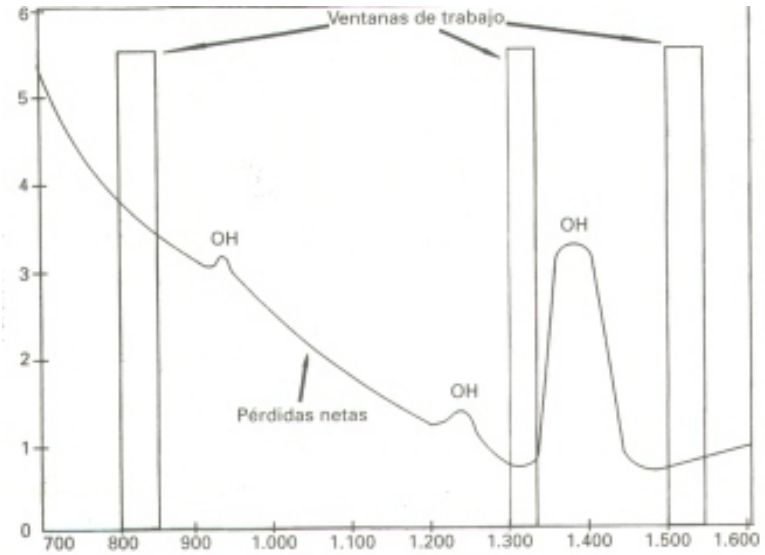


Fig. 2.1: Ventanas de trabajo de la fibra óptica.

En el caso de que la red no sea muy extensa no será necesario el uso de amplificadores por lo que la red será de tipo pasiva. Si la red tiene una gran extensión se hará necesario el uso de amplificadores, la red será de tipo activa. Las redes de transporte se diseñan para poder interconectar subredes clientes. Los nodos que limitan la red se encargan de convertir la información que llega desde las subredes en forma electrónica a señales ópticas.

A nivel físico una red óptica de transporte requiere conmutadores ópticos unidos mediante fibra óptica.

Los conmutadores y la fibra deberán cumplir diferentes condiciones: como red de transporte debe soportar el transporte de tráfico que proviene de otras redes con diferente arquitectura, y por lo tanto diferentes protocolos.

Es por ello que los grupos de trabajo han desarrollad el estándar GMPLS para poder soportar redes de múltiple naturaleza (eléctrica y óptica), redes asíncronas y síncronas (ATM, SDH), múltiples protocolos (IP, PPP, HDLC, Frame Relay).

2.2 OTN

Las redes de transporte están especificadas por la ITU-T en varias recomendaciones G.872 para la arquitectura, el G.709 para los formatos y tramas, G.798 para funciones y procesos. En el dominio óptico las OTN están basadas en la tecnología DWDM y el estándar provee métodos de supervisión y administración de la red. La jerarquía de Redes de Transporte Óptico está basado en el camino óptico (OCh – Optical Channel) dónde la carga de datos es soportada por una *lamda*, incluyéndose en esta tecnología canales de supervisión y de administración para estableces la señalización de la red.

En un principio solo conexiones punto a punto y anillos WDM eran posibles pero gracias a sistemas ópticos de conmutación a nivel óptico han permitido que se realicen sólo pequeñas conversiones óptico-eléctricas y viceversa empleadas para la regeneración de la información.

Redes de transporte con conmutadores eléctricos puedes ser cambiados por *cross-connects* ópticos, lo dicho nos ahorra la conversión de la señal a eléctrico con el inconveniente de caros *transceivers* y la penalización de rendimiento que ello conlleva.

Para hacer un símil entre las OTN y la capa OSI que hace de referencia a los desarrolladores, de redes LAN IP entre otras, las redes ópticas también están compuestas de capas. Aquí las veremos como planos.

Los principales son:

- el plano de control es el que se encarga de establecer, supervisar, mantener y liberar las conexiones y las llamadas.
- El plano de gestión es el encargado de la supervisión, configuración, seguridad y facturación del sistema.
- Por último, el plano de transporte, es el encargado de la transferencia de información de los usuarios de un lugar a otro, ya sea unidireccional o bidireccionalmente.

Otra de las características que han sido incorporadas en las redes ópticas de transporte es la capacidad de mapeo y adaptación de señales. Como comentábamos antes las redes actuales son muy diversas y utilizan diferentes tecnologías y tipos de tramas, las redes de transporte

ópticas tienen que tener la capacidad de poder incluir tramas de un amplio aspecto y forma. Y como se introdujo antes GMPLS(RFC 4139) será el protocolo que nos arrope.
El mapeo de las señales de la capa superior a la que realiza el transporte como la capa 2 en el nivel OSI, está basado en el *Generis Framing Procedure*. Con este tipo de mapeo los bloques de información que nos llegan son decodificados y mapeados en bloques fijos de tramas GFP, y después se transmiten inmediatamente sin esperar a recibir todo el paquete de información. Esto permite que topologías de red como LAN o SAN tengan una muy baja latencia.
En conclusión cuando se complete la introducción de redes ópticas de transporte, los operadores de servicios podrán considerar las ventajas y avances que esta tecnología. Por ahora prácticamente todas estas tecnologías son abastecidas al cliente bajo demanda.

2.3 ASON

Como ya comentamos antes, los grupos de trabajo están desarrollando una evolución de estas redes para que los procesos asignación de recursos y servicios se automaticen, aquí aparecen las ASON (Automatically Switched Optical Network).
Los trabajos sobra ASON podrían englobarse, dentro de OSI, en la Capa 2 y 3.

2.3.1 Descripción

La arquitectura de las redes ASON está determinada por la topología y los elementos de transmisión que conforman los nodos. Las topologías malladas requieren menos recursos de red, le otorgan mayor flexibilidad, facilitan su crecimiento.
Las redes clásicas, basadas en anillos, son de aprovisionamiento mucho más complejo, son más rígidas, requieren más recursos de red y son de difícil crecimiento e interoperabilidad. Estos factores determinan que el coste del hardware de red y los costes operacionales sean sensiblemente superiores. Los elementos clave que conforman los nodos en las redes malladas son los conmutadores ópticos (OXC o ROADM) y los transmisores a longitud de onda DWDM sintonizable. La operación combinada de ambos dispositivos permite la provisión dinámica y en tiempo real de caminos ópticos y de ancho de banda, mediante la extracción e inserción de portadoras ópticas entre redes para el establecimiento de los enlaces ópticos. El control y la inteligencia de red de transporte residen en el estándar GMPLS (Generalized Multiprotocol Label Switching).

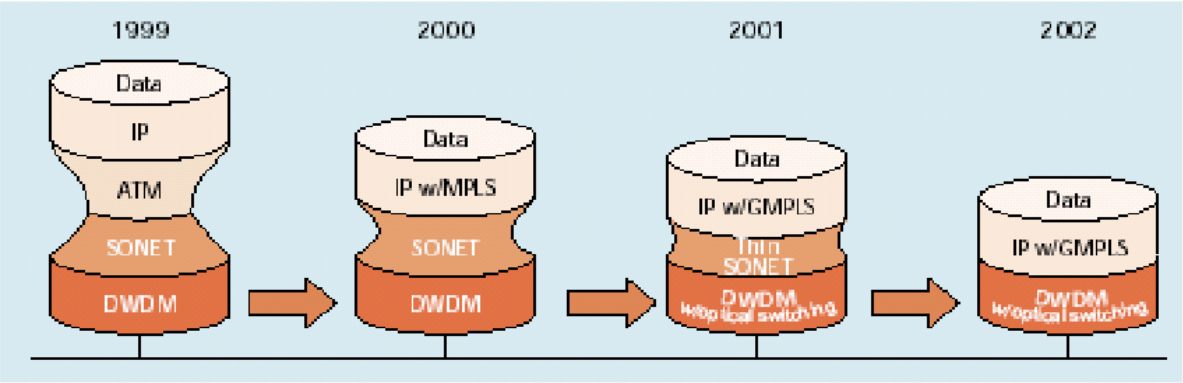


Fig.2.2 Evolución de las pilas de protocolos

ASON no deja de ser un futuro inmediato, es decir, todas las teorías que podemos exponer son más experimentales y metafísicas que realmente evaluables en un entorno empresarial y de operadoras. Podemos ver en la fig.2.2 la evolución que han tenido las pilas de protocolos.
Esta red estará orientada a conexión, ofrecerá QoS, está formada por nodos ópticos que nos darán la capacidad de encaminar longitudes de onda dinámicamente, descubrimiento de nodos vecinos, descubrimiento de la topología y de recursos.
Las primeras redes ópticas fueron concebidas para ser manejadas por sistemas de control centralizado, pero esto aun generaba grandes dificultades, en su capacidad de procesamiento y en la asignación rápida de recursos.
Es así como surge la idea de crear un sistema distribuido, basado igualmente en redes ópticas, encargado del enrutamiento, señalización, establecimiento de recursos y facturación. Este es el concepto básico de las “Redes Ópticas Conmutadas Automáticamente” ASON (Automatic Switched Optical Network). En otras palabras, la OTN tradicional emigrará de redes controladas por una arquitectura tipo TMN (plano de gestión único) hacia un nuevo concepto distribuido como se puede observar en la figura 2.3, donde su ente fundamental lo constituye el Plano de Control.
Cuando hablamos de las OTN, ya describimos los planos que implementa, ASON, como evolución, hereda estos conceptos.
El estándar ASON define de una manera general el plano de control y las interrelaciones básicas con la capa de transporte y la capa de gestión. También define las interfaces de los diversos planos.

2.3.2 Características de ASON

1) Capacidad para introducir nuevos servicios ópticos
Entre estos servicios, se destacan dos: “Servicio de Ancho de Banda BW bajo Demanda” (BODS) y “Redes Privadas Virtuales Ópticas” OVPN.

El BODS es implementado básicamente por las conexiones conmutadas, y está dirigido a usuarios con gran demanda de capacidad y que necesitan nuevas conexiones o reconexiones por períodos cortos. ASON puede proveer nuevas conexiones en segundos, en lugar de días u horas que tardaba cuando se realizaba la petición vía TMN.

El servicio de OVPN debe cumplir con los requerimientos de los operadores, en el sentido de permitirle al usuario tener visibilidad y un control más o menos limitado sobre los recursos de la red reservados para este fin. Se le puede dar un control limitado a los usuarios, obviamente el Operador de ASON mantiene total control sobre todos los otros recursos.

2) Capacidad de Enrutamiento Dinámico

Son factores fundamentales de este punto, los siguientes:

2.1) Auto detección de "vecinos": Cada Nodo reconoce a sus nodos adyacentes.

2.2) Auto detección de enlaces de conexión: Los nodos ASON son capaces de chequear automáticamente la disponibilidad de enlaces.

2.3) Auto detección de topología: Tan pronto los OXC (Optical Cross Connect), (elementos que se encargan de enlazar los diferentes caminos)

2.4) La arquitectura de enrutamiento está basado en *áreas* (Routing Áreas – RAs) subdividiendo la red por zonas de descubrimiento y rebajando el coste de actualización.

3) Aumento de la estabilidad y escalabilidad en los sistemas de gestión de operaciones.

Dado que la red ASON es capaz de enrutar y restaurar las conexiones automáticamente en una forma mucho más tolerante a fallas, el requerimiento de mantener la consistencia de la base de datos de topología es satisfecho.

El plano de control de ASON se encuentra distribuido sobre todos los elementos de la red, lo que lo hace más escalable la red.

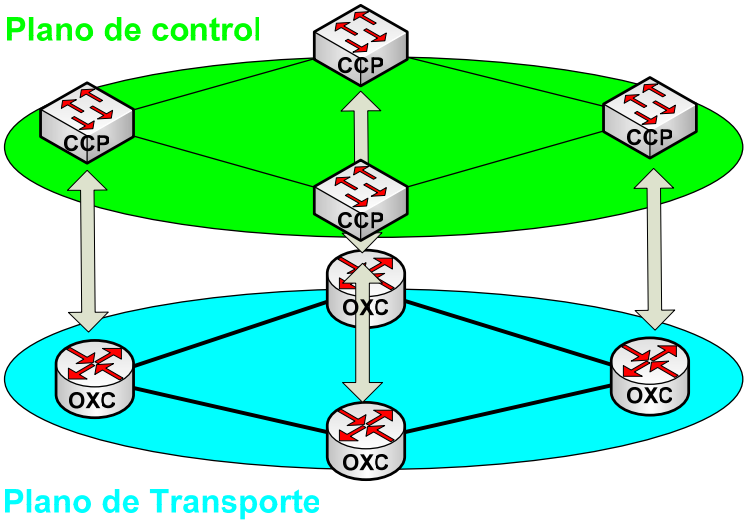


Fig.2.3 Representación del control distribuido

4) Restauración más eficiente de servicios.

Antiguamente, cuando ocurría una falla en un sistema, la base de datos de topología, adyacencia y enrutamiento requería de algún tiempo para restablecer la comunicación, dado que se requerían de cálculos realizados por una pieza de software para reestructurar las rutas de conexión. Ahora ASON ofrece una restauración que es descentralizada.

La actualización de la base de datos de la red es lograda virtualmente en tiempo real. Esto permite recalcular los caminos de restauración con más precisión.

2.3.3 Planos de ASON

Las redes de conmutación automática, tal como se conciben en el estándar ASON, están constituidas por tres planos: el de transporte, el de control y el de gestión. En la figura 2.4 se muestra un enfoque macro de la interacción entre estos tres planos.

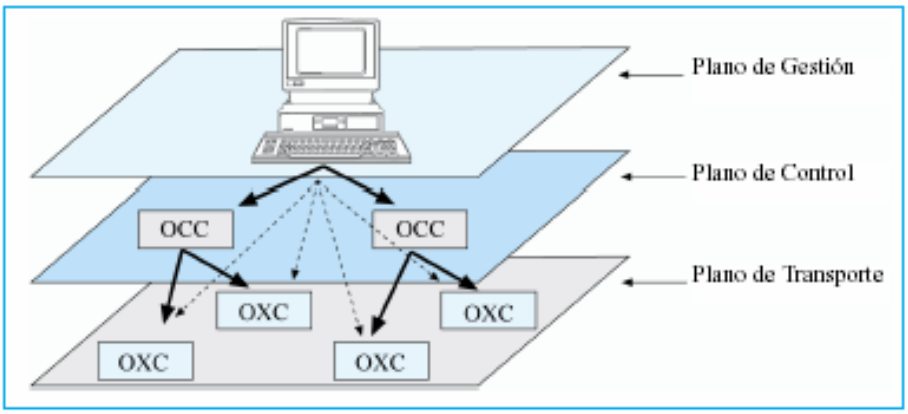


Fig. 2.4 Dibujo descriptivo de los planos de las OTN

Su enfoque fundamental está dirigido a proveer a las redes ópticas con un plano de control inteligente, que incorpore aprovisionamiento dinámico de la red combinado con funciones de supervisión, protección y restauración de las conexiones.

- Plano de transporte: unidad lógica y física que hace el transporte de los datos, así como la amplificación y regeneración de la señal óptica.
- Plano de control: es la entidad lógica que da inteligencia a la red, tales como enrutamiento, señalización y restauración de caminos
- Plano de gestión: unidad lógica que permite al operador de la red gestionar su comportamiento.

ASON fue diseñada en su concepción inicial para soportar múltiples clientes y diferentes tecnologías. Esta diversidad crea los diferentes dominios de cada plano. La conexión intra-dominios e inter-dominios dentro de la capa de control, se realiza a través de las interfaces

I-NNI (Internal Network to Network Interface), E-NNI (External Network to Network Interface) respectivamente y I-NMI (Internal Network to Managment Interface).

Adicionalmente existe otra interfaz en la capa de control, es la que enlaza el dominio de los usuarios con la red de los Proveedores de servicio, y se conoce por las siglas UNI (User to Network Interface).

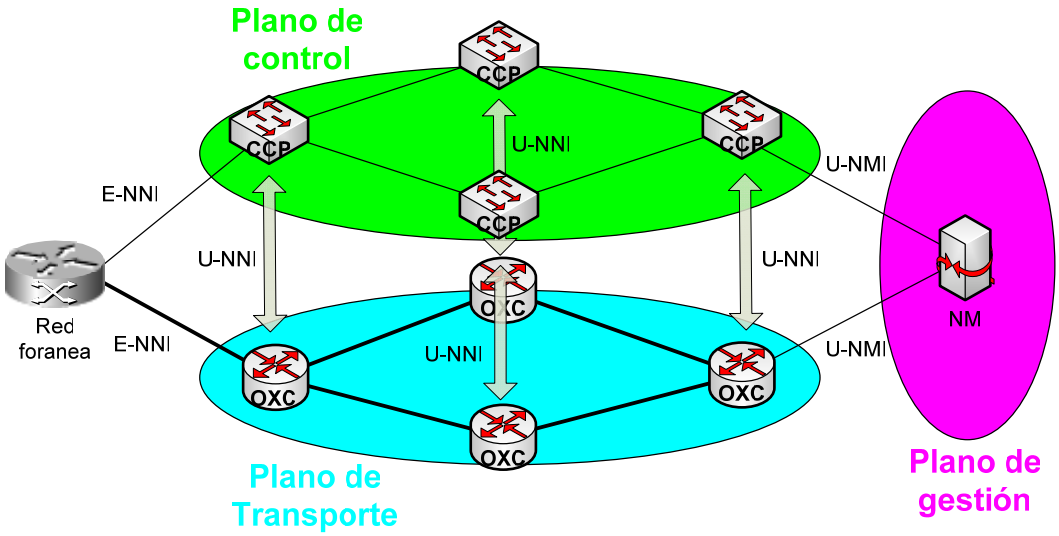


Fig. 2.5 Visión de las interficies que comunican los diferentes planos

El plano de transporte contiene todos los elementos de transporte de red (switches y enlaces) que hacen posible la conexión.

Las conexiones extremo a extremo son establecidas dentro del plano de transporte bajo el control del plano de control de ASON, siendo este elemento la principal característica de interrelación entre estos planos.

Los elementos básicos que conforman el plano de transporte son:

– Conmutadores Ópticos:

- OXC Conmutadores ópticos/eléctrico/ópticos
- PXC Conmutadores ópticos/ópticos

– Topología de red tipo malla, de fibra óptica

– LMP Protocolo de Capa de Enlace, Link Management Protocol RFC(4394)

ASON define una arquitectura para el Plano de Control que permite el establecimiento y desconexión de las sesiones como resultado de requerimientos de los usuarios. Para lograr una cobertura global y el soporte de múltiples tipos de clientes, es que se describe esta arquitectura en términos de componentes y de un conjunto de reglas y puntos de referencia que se deben aplicar en los puntos de interfaz entre los clientes y la red, y entre las propias redes en sí.

Una arquitectura del plano de control bien diseñada debe dar a los proveedores de servicio, un mejor control de su red

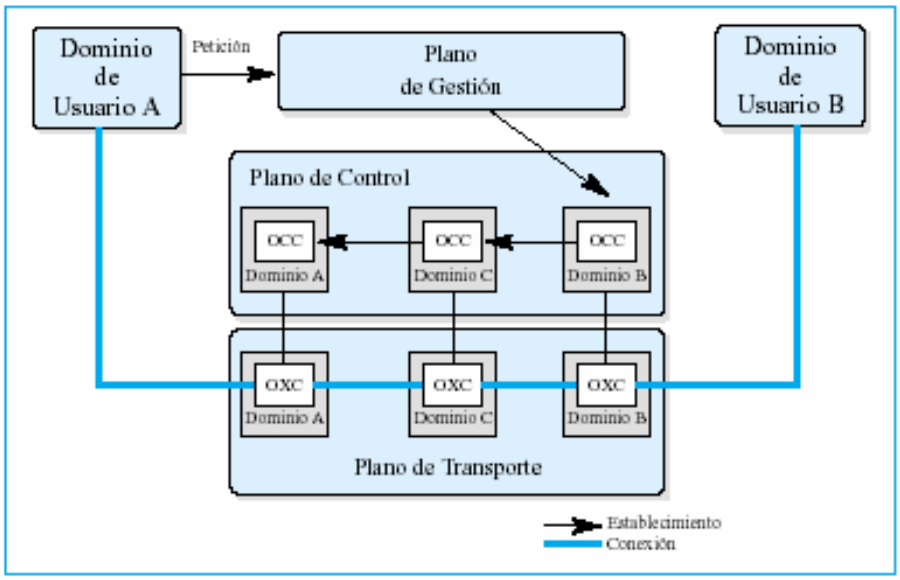


Fig.2.6 Visión general de los bloques de las ASON

2.3.4 Enrutamiento y Señalización

La arquitectura de ASON trata separadamente las llamadas y su control de conexión. Esto permite la introducción de servicios mejorados, en donde una simple llamada puede estar compuesta de más de una aplicación.

Esta característica brinda beneficios a las áreas de mantenimiento y restauración.

Las redes ópticas son capaces de entregar conexiones de banda muy ancha a través de los *lightpaths*.

La arquitectura de enrutamiento de las ASON asume que la red será subdividida en *áreas* (RAs). Puede soportar múltiples protocolos de enrutamiento pero el elemental será OSPF. Cada *área* tendrá Controladores de Enrutamiento (RCs) y de backup por si los primero caen, para el intercambio de información de routing entre *áreas*. La estructura de los nodos de la red ASON es jerárquica, habiendo unos nodos padres, y otros hijos. La comunicación entre controladores de *área* será de padre a padre nunca de un hijo de un *área* a otro hijo de otra *área*, se escalará siempre.

2.3.5 Ventajas e inconvenientes de las ASON:

2.3.5.1 Cuestiones favorables

La capacidad de las redes ASON que permiten a las operadoras dar soporte a los clientes es una gran ventaja, ya que la integración de todos los clientes sobre una misma plataforma reduce mucho los costos y simplifica el manejo de la red.
La utilización de protocolos generalistas, actualizables y escalables, además de la posibilidad de conmutar canales ópticos, permiten gran movimiento en la estructura de la red.
El plano de control permite una gran supervisión de los recursos libres de la estructura, pudiendo gestionarlos sacando un rendimiento óptimo.
El operador podrá garantizar un servicio a un cliente aplicando sus políticas de control sobre los recursos, pudiendo de esta manera ajustar el precio.
La separación del plano de control del plano de transporte nos permitirá dar los siguientes servicios:

- Canales semipermanentes conmutados automáticamente.
- Canales permanentes automáticos.
- Redes ópticas virtuales.
- Asignación de longitudes de onda a flujos entre nodos extremos.

2.3.5.2 Cuestiones críticas

A pesar de estas ventajas que promete ASON tiene puntos no tan favorables.
La granularidad de las longitudes de onda, hace que la asignación de una *lamda* a un cliente sea improbable mientras el coste de ésta sea tan elevado. No es posible asignar *lamdas* a tráficos exclusivos ya que tiene una elevada capacidad y se estaría desperdiciando.
La solución a esto es TDM, que permite conmutar muchos flujos una misma longitud de onda. El problema está en que deshomegeneizamos nuestra tecnología.

2.4 Protocolos para OTN's

No es nuevo en las redes de comunicaciones la necesidad de protocolos para control y gestión de los equipos. Será necesario establecer las comunicaciones, liberarlas. También será necesario la escalabilidad y la tolerancia a fallos como describimos anteriormente, pero en esta sección vamos a entrar a concretar algo más en los protocolos que realizan estas tareas.
MPLS es un tipo particular de conmutación mediante etiquetas diseñado específicamente para dar a las redes características orientadas a conexión, como por ejemplo IP (Internet) a nivel 2.
Los protocolos de distribución de etiquetas diseñados para funcionar con MPLS son LDP

(G.7713.3) y RSVP(G.7713.2). Mediante el uso de LDP (RFC 3475) o RSVP (RFC 2205)se pueden establecer LSPs (Label Switched Path). El nexo entre las etiquetas y los *lightpath* reside en que las etiquetas actúan como banderas que marcan el *lightpath*. Un *lightpath* puede ser permanente o conmutado. Los *lightpaths* conmutados se establecen y eliminan de forma dinámica en tiempo real gracias a la señalización utilizada. Este tipo de *lightpaths* pueden mantenerse establecidos largos o cortos períodos de tiempo.

2.4.1 MPLS/GMPLS

El desarrollo de MPLS se llevó a cabo a mediados de 1990 para conseguir mayor eficiencia en redes ATM que transportan tráfico IP. La implementación de MPLS sobre una red IP tradicional introduce las siguientes ventajas:

- El encaminado se realiza de acuerdo a etiquetas de longitud corta y fija de nivel 2 en la torre OSI. Esto permite que el encaminado sea más eficiente que con las cabeceras tradicionales de nivel 3 y longitud variable.
- Ofrece la posibilidad de encaminado explícito, se crea un camino completo examinando un solo paquete. Los demás paquetes con el mismo destino no deberán ser examinados.
- Permite una administración de tráfico más flexible y eficiente.
- Ofrece soporte QoS, mediante el establecimiento de rutas explícitas.

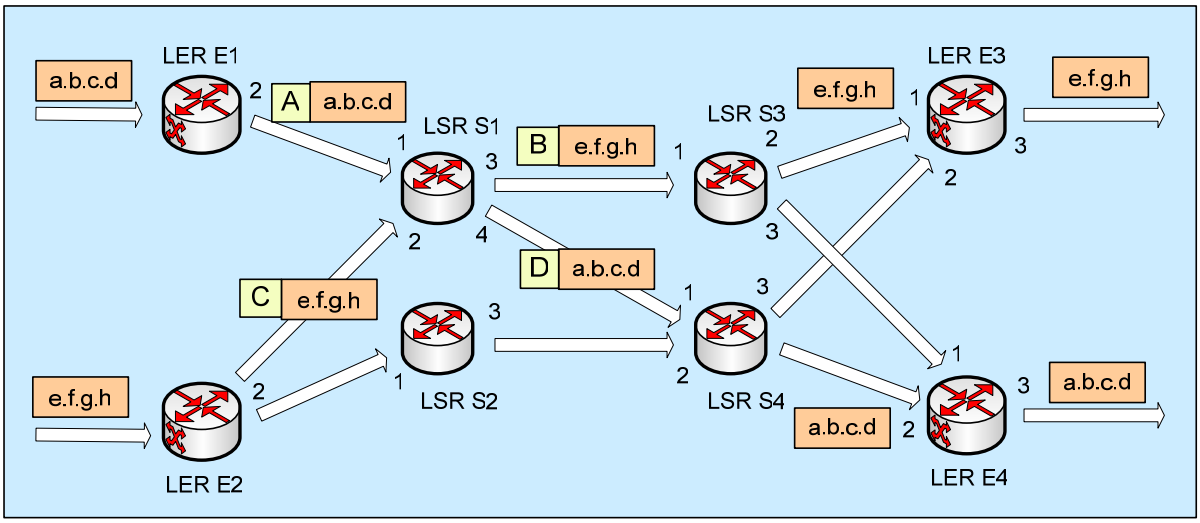
Utilizando MPLS se distinguen entre dos tipos de nodo: nodos límite y nodos del núcleo de la red. Los nodos límite reciben el nombre de LER (Label Edge Router). Un LSP (Label Switched Path) es una conexión unidireccional que comienza en un LER de entrada. Los nodos de núcleo reciben el nombre de LSR (Label Switching Router). Un LSR realiza el encaminado de los paquetes mirando sólo la etiqueta.

Se almacena en el componente llamado RIB (Routing Information Base) la información referente al enrutamiento de la red externa al dominio MPLS. Un LER utiliza la información contenida en RIB para crear la información que guarda en el componente llamado FIB (Forwarding Information Base). En el plano de control el componente de un LER interactúa con el de un LSR mediante el protocolo de señalización MPLS. La información resultante de esta comunicación se almacena en LIB (Label Information Base). En el LIB hay la asociación de etiquetas negociada con otros nodos MPLS. Un LER puede encaminar paquetes del dominio externo, añadir una etiqueta a un paquete externo (label push) o eliminar una etiqueta de un paquete (label pop). Un LSR no puede encaminar paquetes externos, sin embargo también puede añadir o modificar y eliminar etiquetas de paquetes previamente etiquetados.

La información en NHLFE (Next Hop Label Forwarding Entry) determina la acción que debe realizar el LSR o LER sobre la etiqueta del paquete. En la información dentro de la tabla NHLFE también puede haber aspectos del nivel de enlace. MPLS realiza un proceso llamado eliminado en el penúltimo salto (penultimate hop popping). En este proceso es el penúltimo nodo a lo largo del LSP quien elimina la etiqueta permitiendo así que el último nodo reciba un paquete y no uno etiquetado.

En ejemplo:

FEC	Puerto Out	NHLFE	Puerto In	Etiq In	Puerto Out	NHLFE	Puerto In	Etip In	Puerto Sou	NHLFE	FEC	Puerto Salida	NHLFE
a.b.c.d	**2**	**Mas A**	**1**	**A**	**4**	**D**	**1**	**B**	**2**	**Elim**	**e.f.g.h**	**3**	
			2	**C**	**3**	**B**							



FEC	Puerto Out	NHLFE	Puerto In	Etiq In	Puerto Out	NHLFE	Puerto In	Etip In	Puerto Sou	NHLFE	FEC	Puerto Salida	NHLFE
e.f.g.h	**2**	**Mas C**					**1**	**D**	**4**	**Elim**	**a.b.c.d**	**3**	

Fig. IP sobre MPLS .Plan de enrutamiento MPLS

Un protocolo de encaminamiento, del tipo no orientado a conexión como IP, cada router toma la decisión del siguiente salto. Cada router se basa en el contenido de la cabecera del paquete recibido junto con la información de la topología. La información de la topología la recibe a través del protocolo de encaminado. Normalmente una red calcula el camino más corto basándose en la métrica configurada en su protocolo interior de estado de red. Sin embargo a menudo el problema de encaminado necesita basarse en otros criterios, por ejemplo QoS, a este se lo llama encaminamiento con restricciones. El encaminado basado en restricciones

hace la elección óptima de ruta dependiendo del estado de la red en el momento de la petición. Para que la elección sea óptima los nodos siempre deben tener información actualizada del estado de la red. Este hecho da lugar a mayor complejidad al sistema.

Debido a que MPLS utiliza la técnica de separar el control del encaminado es capaz de operar sobre diferentes protocolos de nivel de enlace. Esto significa que MPLS puede ser implementado como un router más switch integrados.

Después de la integración de MPLS en las redes no orientadas a conexión como IP se trato de hacer las redes más homogéneas tratando de subir un nivel más en los protocolos de control y diseñar uno que pueda englobar la otra gran jerarquía de red como es SDH, sugió GMPLS y con él podemos dar soporte a redes de conmutación de lambas (WDM), de paquetes (IP) y circuitos (SDH).

GMPLS soporta cinco interfaces: interfaz de conmutado de paquetes, interfaz de conmutado a nivel 2, interfaz de multiplexado por división de tiempo, interfaz de conmutado por longitud de onda y interfaz de conmutado de fibra.

Un interfaz de conmutado de paquetes reconoce los límites del paquete y puede encaminar paquetes basándose en la cabecera IP. Un interfaz de conmutado de nivel 2 reconoce los límites de una célula o “frame” y puede encaminar los datos basándose en el contenido de la cabecera de la célula o “frame”. El ejemplo de ATM que encaminan células basándose en su valor VPI/VCI o switches Ethernet que encaminan el tráfico basándose en la información de MAC. Un interfaz de multiplexado por división de tiempo encamina datos basándose en las ranuras temporales que forman tramas, “frames” en el caso SONET/SDH. Un interfaz de conmutado por longitud de onda encamina señales ópticas de una longitud de onda entrante a otra saliente. Como ejemplo los OXCs que operan a nivel de longitud de onda individual. Interfaz de conmutado por fibra encamina señales de una o más fibras de entrada a una o más fibras de salida. Como ejemplo los OXCs que operan a nivel de fibra.

En GMPLS se utiliza la distribución de etiquetas ya comentamos que es una evolución de MPLS, Algunas formas nuevas de etiquetas son necesarias para soportar la amplia visión de GMPLS en el dominio óptico y en el multiplexado por división temporal. La nueva etiqueta no sólo permite que las etiquetas tradicionales viajen junto con el paquete asociado también permite que las etiquetas identifiquen ranuras temporales, longitudes de onda o fibras. Los protocolos de distribución de etiquetas LDP y RSVP. Los protocolos interiores ISIS y OSPF también han sido extendidos para poder utilizarse con las tecnologías ópticas. También se ha desarrollado un protocolo administrar el nivel de enlace en redes ópticas, el protocolo LMP (Link Management Protocol).

2.4.2 LDP

LDP es un protocolo nuevo y diseñado para cumplir sólo con la distribución de etiquetas. El RFC 3037 describe las aplicaciones de LDP siendo útil en redes de encaminado salto por salto donde la eficiencia sea un parámetro muy importante.
Utiliza los siguientes tipos de mensajes para intercambiar información:

- **Mensajes de descubrimiento:** se realiza un intercambio periódico de mensajes “Hello” para anunciar y verificar la existencia de LSRs conectados tanto directamente como indirectamente.
- M**ensajes de sesión:** se utilizan para establecer, negociar parámetros, inicializar, mantener y terminar sesiones LDP.
- M**ensajes de anuncio:** se utilizan para crear, cambiar, o eliminar asignaciones de etiquetas

El anuncio LDP de asignado de una etiqueta utiliza dos mensajes: el mensaje de mapeo de etiqueta (label mapping) y el mensaje de petición de etiqueta (label request). Una vez los conmutadores se han descubierto y han establecido una sesión, intercambian mensajes de mapeo de etiquetas. LDP también define mensajes para situaciones específicas. Un mensaje (label withdraw) permite a un conmutador pedir que un peer deje de utilizar un asignado de etiqueta específico. El mensaje liberar etiqueta (label release) indica que una etiqueta pedida o recibida previamente ya no es necesaria. Se define también el mensaje de petición de aborto de etiqueta (label abort request), la utilidad de este mensaje es acabar con un mensaje de petición de etiqueta pendiente.

2.4.3 RSVP

RSVP tiene muchos mecanismos para realizar la señalización de la distribución de etiquetas. La función principal de RSVP es establecer reservas para flujos unidireccionales de paquetes. En el RFC 2205 se definen tres tipos de mensajes: establecimiento de reserva, liberación de reserva (tear down) y error. Cada mensaje esta compuesto por varios objetos, están descritos en el RFC 3209. La extensión de RSVP, RSVP-TE añade otro mensaje: el mensaje "Hello".
La especificación de este protocolo utiliza el concepto de reserva basada en el receptor, donde el emisor primero envía un mensaje “Path” que identifica el flujo y las características del tráfico. El mensaje contiene un identificador de sesión, una petición de etiqueta, un campo de especificación de tráfico, en el que se especifican la velocidad de pico, la velocidad de "troughput", tamaño de la ráfaga y amaño máximo del paquete y por último un campo de ruta explícita.

El funcionamiento de RSVP es básico. Un nodo origen que quiere establecer un enlace para cursar una llamada con un destino, le envía a éste un mensaje “Path” y el destino responderá con un mensaje “Resv” si desea iniciar el asignado de etiqueta pedido en el mensaje “Path”. El mensaje “Resv” contiene el mismo identificador de sesión que contenía “Path”, incluido va un objeto donde se almacena la ruta seguida. Se encuentran dos estilos de reserva: estilo de filtro fijo y estilo explícito compartido. En el estilo de filtro fijo cada par emisor-receptor tiene asignado una etiqueta y un campo de especificación de tráfico. En el estilo explícito compartido se asigna una etiqueta distinta a cada emisor pero los emisores comparten explícitamente las mismas especificaciones de reserva de flujo.

RSVP-TE define dos mensajes para liberar un camino: liberación de “Path” y liberación de “Resv”. Ambos mensajes de liberación se envían en la dirección opuesta al correspondiente mensaje “Path” y “Resv”. Los mensajes de liberación eliminan la asociación que creo su mensaje opuesto.

Hay mensajes de notificación de error para los mensajes “Path” y “Resv” además de un mensaje opcional de confirmación “Resv”. Esto quiere decir que si se trata de trazar una ruta hacia un destino pero alguno de los saltos no cumple con los requisitos necesarios se descarta la reserva y se envía un mensaje “ResvErr” y se envía hacia el router.

El mensaje opcional “Hello” está definido en el RFC 3209 para RSVP-TE. Este mensaje permite a un LSR detectar más rápidamente que un vecino ha fallado. En el caso de utilizar RSVP el refresco se realiza en intervalos.

Trataremos de ilustrar un poco las lineas escritas anteriormente en la figura 2.7.

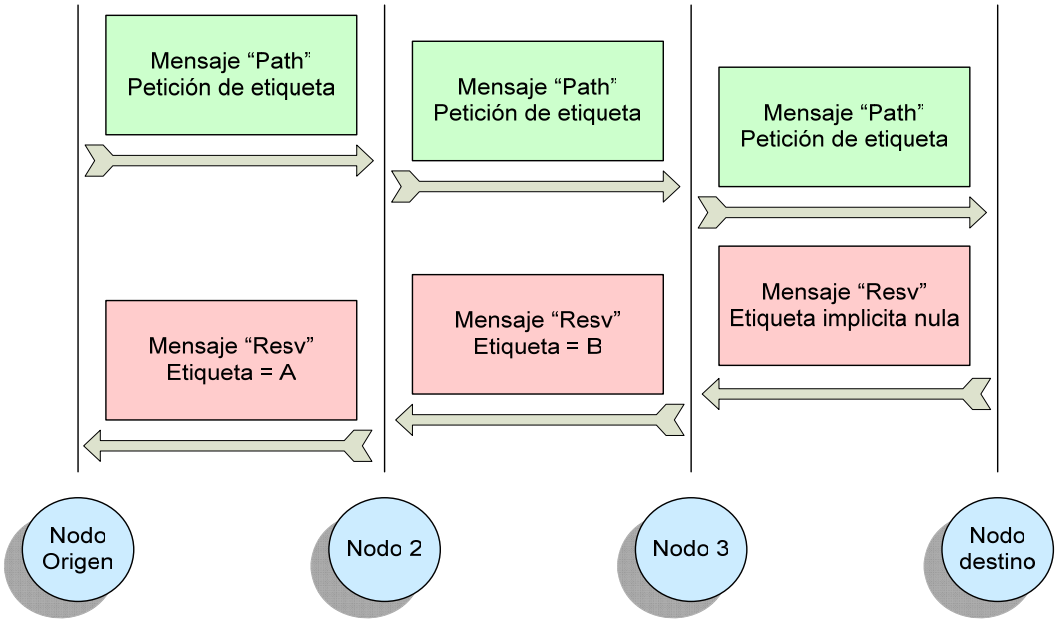


Fig. 2.7 Modelo de funcionamiento de RSVP

El problema fundamental de RSVP es el refresco de la información de la topología. Este refresco se realiza en intervalos normalmente de 30 segundos. Realizar el refresco en intervalos pequeños supone el envío de gran cantidad de información periódicamente. Esto hace que aparezcan retardos en el procesado de los mensajes y puede llegar a hacer al sistema no escalable.

2.5 WDM/DWDM

Hemos llegado al concepto clave a nivel físico (Capa 1 OSI) y más importante para el desarrollo de redes ópticas. WDM/DWDM es la idea con la que de desarrollan los OXC y sobre la que se aplicarán los protocolos explicados anteriormente para su control y gestión. Podríamos decir que es el concepto principal de las OTN's.

En una primera generación las redes ópticas eran enlaces punto a punto de gran capacidad, para cubrir distancias largas, por cada fibra viajaban *lamdas* multiplexadas. En cada extremo de la fibra se demultiplexaban y se vuelven a multiplexar para poder procesar los datos que contiene.

El concepto fundamental es que cada señal digital es transportada por una portadora óptica independiente en una misma fibra. Permite aumentar de una forma económica la capacidad de transporte de las redes existentes, por medio de multiplexores, DWDM combina multitud de canales ópticos sobre una misma fibra, de tal modo que pueden ser amplificados y transmitidos simultáneamente. Cada uno de estos canales, a distinta longitud de onda, puede transmitir señales de diferentes velocidades y formatos: SDH/SONET, IP, ATM, etc. Es decir, DWDM puede multiplexar varias señales TDM sobre la misma fibra. Una de las principales ventajas de los sistemas DWDM es su modularidad, la cual permite crear una infraestructura conocida como "grow as you go", que se basa en añadir nuevos canales ópticos de forma flexible en función de las demandas de los usuarios. Así, los proveedores de servicio pueden reducir los costes iniciales significativamente, al tiempo que desarrollan progresivamente la infraestructura de red que les servirá en el futuro.

La revolución de los sistemas DWDM no hubiese sido posible sin las características claves de la tecnología óptica, más abajo nombraremos las más destacadas:

- La capacidad que poseen los diodos láser de emitir luz a una longitud de onda estable y precisa con un ancho de línea espectral muy estrecho.
- El formidable ancho de banda de la fibra óptica (varios THz), el cual no ha sido aprovechado completamente durante tiempo.
- La transparencia de los amplificadores ópticos de fibra (EDFA) a las señales de modulación y su habilidad para amplificar de forma uniforme varios canales simultáneamente.

La primera generación de redes WDM surgió para aliviar el problema del agotamiento de capacidad de las redes SDH/SONET, y tal y como se ha comentado, consistía simplemente en combinar múltiples longitudes de onda en una misma fibra. El número de canales era pequeño (del orden de 16) y la protección se realizaba en las capas 2 ó 3.

La segunda generación de redes metropolitanas DWDM dobla el número de canales e introduce protección de anillo y OADMs estáticos, permitiendo que los proveedores de servicio proporcionen servicios basados en longitud de onda. Eso quiere decir que ahora en los anillos ópticos, los proveedores de servicios pueden reconfigurar los equipos para reencaminar *lamdas* a petición de los usuarios, los OADMs son los responsables, la **figura 2.8** es un esquema del comportamiento de un OADM, se puede ver como extrae y rellena *lamdas* con tráfico permitiendo "enlutar" la información. La conmutación entre múltiples anillos metropolitanos se realiza de forma centralizada y las longitudes de onda se demultiplexan antes de ser encaminadas de forma individual.

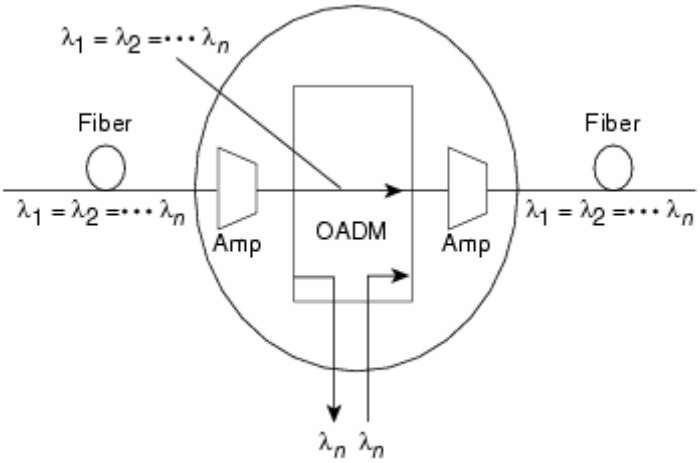


Fig. 2.8: Esquema del funcionamiento de un Add-Drop Multiplexer

Las redes ópticas de tercera generación se caracterizan por ofrecer gestión dinámica de las longitudes de onda directamente en el dominio óptico, proporcionando ventajas significativas con respecto a la segunda generación de redes. Asimismo, el número de canales es mayor y existe una monitorización de prestaciones más sofisticada que se realiza sobre cada canal óptico.

La conversión de longitud de onda es una funcionalidad clave en las redes ópticas WDM por diversas razones. Por un lado, una red que emplea convertidores de longitud de onda resulta más fácil de gestionar puesto que la asignación de longitudes de onda puede determinarse de forma local. Por otro lado, el bloqueo de longitud de onda en los nodos ópticos puede reducirse cuando se conmuta en el dominio de la frecuencia. Por ello la red resulta más

flexible y fácil de configurar. Y en general, los recursos de la red se utilizan de forma más eficiente bajo patrones de tráfico dinámicos.

Tal y como van creciendo las necesidades de los servicios de voz y datos, éstas se convierten en un punto crítico para las empresas, los proveedores de servicios que requieren de redes de más capacidad y tolerante a fallos.

A diferencia de TDM en que una portadora lleva la información multiplexada temporalmente, en DWDM la información es repartida, multiplexada por las múltiples portadoras que existen en la fibra.

Esto significa que cada canal tiene su propio ancho de banda dedicado, a diferencias de sistemas TDM que todas las señales son transportadas en ranuras de temporales.

El protocolo DWDM es fundamental destacar que el ancho espectral de cada una de las longitudes de onda son mucho más pequeñas que en el protocolo WDM, por lo tanto tendrá mucha más capacidad. Los límites de la reducción de este ancho espectral no son conocidos y quizá difícilmente alcanzable, así los límites a mediados del año 2000 la capacidad por fibra rondaba las 128 *lamdas* por fibra.

WDM y DWDM utilizan fibras monomodo para transportar varias *lamdas*. Sin que eso confunda con las representaciones de muchas publicaciones en que aparecen las fibras multimodo representadas por señales multicamino. En las fibras multimodo solo se transmite una *lamda*.

2.5.1 Valor de DWDM en las redes MAN y WAN

DWDM es el gran ganador como arquitectura de transporte en las redes troncales. Mientras DWDM resuelve el problema del agotamiento de las fibras en redes pasivas en un área metropolitana, su valor en este mercado va aun más allá. DWDM aporta la posibilidad de incrementar la capacidad de una fibra simplemente sintonizando una nueva *lamda* en la fibra existente, a diferencia de la arquitectura SONET (TDM) por ejemplo en que habría que incrementar el cableado.

Desde el punto de vista económico y técnico, la habilidad de esta tecnología de proveer una capacidad de transmisión creciente, y con previsión de ser casi ilimitada, es la ventaja mas obvia de DWDM. Tal y como la demanda crece, la capacidad puede aumentar con la actualización de los equipos terminales. Los costes de "upgrade" son relativamente económicos comparados con una nueva inversión en tirar nuevas fibras.

2.5.2 Switching óptico

La alta disponibilidad que nos ofrece DWDM no sería útil sin un hardware que hiciera rendir dicha arquitectura.

Mucho de los equipos de red, hoy en día, está basado en señales eléctricas, por tanto las señales ópticas tienen que ser convertidas al dominio eléctrico para ser amplificadas, regeneradas o conmutadas para luego reconvertidas al dominio óptico.

La información que viaja a través de una red óptica es tratada en los nodos, en ese punto la tramas son reenviadas por el puerto que traza el mejor camino hacia el destino según factores de gestión. La finalidad del Switching óptico es reemplazar el paso de las tramas al dominio eléctrico. Claramente las ventajas son evidentes ahorrándose el paso óptico-eléctrico-óptico, que conlleva una evidente latencia a la red. Hay diversas propuestas para la conmutación óptica tales como: amplificadores semiconductores, cristal líquido, cristales holográficos y espejos. Una de las técnicas más comunes es la denominada MEMS (micro-electro-mechanical systems):

En los puntos siguientes trataremos de describir brevemente las técnicas de switching.

2.5.2.1 Técnicas de switching óptico

2.5.2.1.1 Introducción a MEMs

MEMs consiste en espejos con un diámetro no mayor al diámetro de un pelo humano que son alineados en unos pivotes especiales que permiten su movimiento en los tres ejes espaciales. El rayo de un puerto de entrada después de incidir en el espejo será redirigido al puerto de salida, según el protocolo de enrutamiento.

Los sistemas convencionales MEMs tienen partes móviles y la rapidez con la que se mueve el espejo es limitada. Aplicando más corriente el espejo ira más rápido, pero hay un límite físico que podemos hacer circular por el *array* de espejos sin quemar los circuitos. Cambiando el diseño de los espejos es posible conseguir movimientos más rápidos es la técnica conocida como “fase MEMs”.

En comparación, MEMs es una tecnología de cambio rápido. Parece tener el monopolio que las arquitecturas de los equipos de conmutación ópticos.

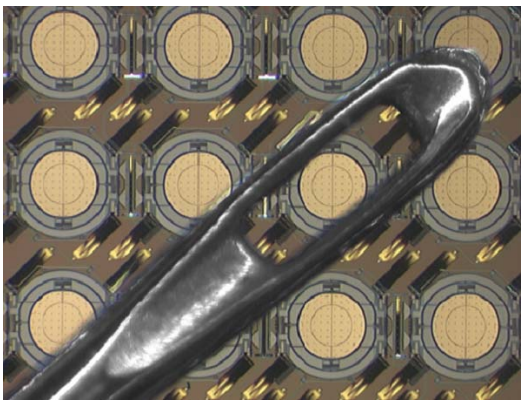


Fig. 2.9 Comparativo de la tecnología MEMS

Una matriz de dos dimensiones con una medida de 32x32 está lista para ser usada en un sistema básico de 256 puertos.

2.5.2.1.2 Conmutadores de Cristal-Líquido

El "cristal-líquido" es un estado que se puede encontrar en muchos materiales orgánicos a ciertos rangos de temperatura. En el estado "cristal-líquido" las moléculas toman cierta orientación dependiendo del campo eléctrico al que están expuestas. Eso es posible, aplicando un voltaje a través de una celda llena de material cristal-líquido, que actuará sobre la orientación de las moléculas. Los cambios de orientación provocan cambios en el comportamiento del cristal-líquido con la luz.

El cristal-líquido actúa después que un rayo de luz llegue a un *splitter* que lo divide en dos señales polarizadas, éstas son entonces dirigidas a 2 celdas llenas de cristal líquido. Dependiendo del voltaje aplicado a la celda cambiará el estado de polarización de los rayos incidentes provocando que éste pase alterado o no. La diferencia entre uno y otro provocará que los rayos salgan por un puerto o por otro.

Estos conmutadores son selectivos con la longitud de onda que incide. Esta característica permite procesar una longitud de onda diferente a deseo del operador.

2.5.2.1.3 Conmutadores Holográficos

La electro-holografía es un método de deflexión de rayos de luz basado en el control de la reconstrucción del proceso del volumen holográfico por la intervención de una campo eléctrico. La aplicación de un voltaje de control es usado para activar hologramas pre-guardados que desvían los rayos de luz apropiadamente.

En uno de los estados del conmutador, el rayo de salida son rayos refractados. Si no hay voltaje aplicado, el cristal es transparente para señales ópticas que pasan rectas, mientras que en el caso contrario el rayo óptico de cruza el rayo es desviado. Es posible almacenar múltiples hologramas en el mismo cristal, esta característica permite poder procesar las diferentes longitudes de ondas que puede traer un rayo láser.

Comparativo:

	Lamda nm	Perdida de inserción	Cross-talk	Perdidas de polarización	Tiempo de conmutado
MEMS	1290-1625	1dB	>50dB	<0,2dB	<1ms
Cristal líquido	1525-1575	<1dB	>40dB	0,2dB	4ms
Holográfico	1310-1550		>30db		<30ns

Tabla 2.1: Compartivo de las diferentes tecnologías de hardware de conmutación

2.6 Hardware óptico

2.6.1 OXC's

Hemos nombrado en reiteradas ocasiones los equipos que hacen posible dar flexibilidad a las redes ópticas.

En la actualidad el desarrollo y estandarización de las redes ópticas están siendo ampliamente promovidas. Se espera que sean funcionales en los aspectos de transporte, multiplexación, enrutamiento, supervisión y tolerante a fallos y que todos estos servicios estén lo más integrados en el dominio óptico.

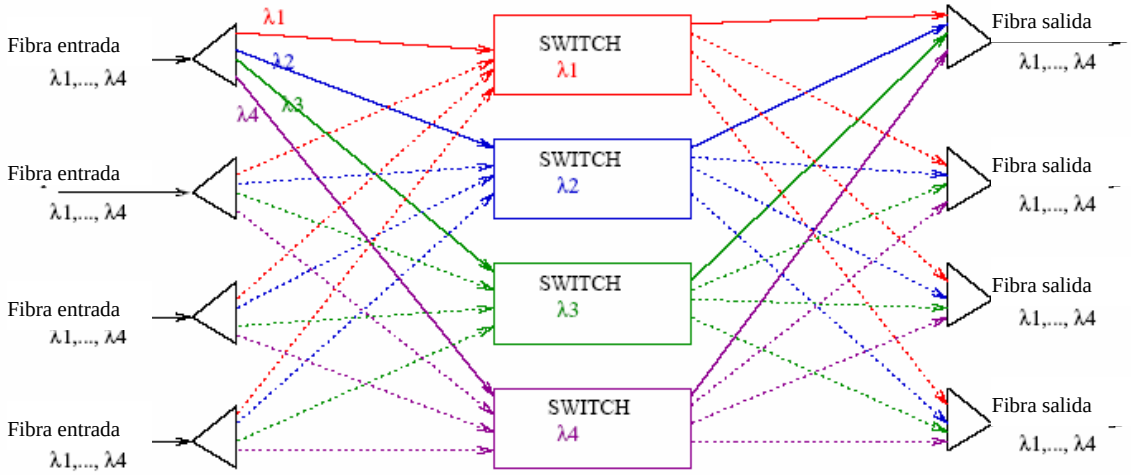


Fig 2.10 Diagrama esquematizado de un *Optical Cross Connect*

Los OXC (Optical Cross-Connect) serán los dispositivos que aprovisionen las redes ópticas de las capacidades de tolerancia a fallos, conmutación y enrutado de *lamdas*.

Como sucede a menudo en tecnología, los OXC no surgen espontáneamente de una necesidad si no que son la evolución de la tecnología actual ampliamente instaurada. Dicha tecnología son las redes SONET/SDH que utilizan la tecnología TDM para multiplexar los canales de comunicación. Los equipos que se encargan de manejar este tipo de redes son los DXC (Digital Cross-Connect) y los ADM (Add-Drop Multiplexer). Juntado elemento de ambos obtendremos los OXC.

Hay dos tipos, ambos clasificados por el modelo de capas, desde la capa encargada del multiplexado óptico (OMS) a la capa encargada de interconectar y conmutar los diferentes canales ópticos (OCH). La función del OMS-OXC es multiplexar los canales ópticos. Básicamente consiste en juntar el tráfico proveniente de un grupo elevado de canales en otro grupo mucho más reducido. Por otro lado los OCH-OXC tienen la misión de interconectar y conmutar cada uno de los canales ópticos que circulan por las fibras conectadas al sistema. El

enrutado de las *lamdas* hace posible de pensar que cada una de estas longitudes de ondas puede ser un camino apto para enviar información a un destino, pudiendo restaurar estos caminos para otras comunicaciones con suma facilidad. Por tanto la gestión de ancho de banda, aquí, se realiza a través de longitudes de onda.

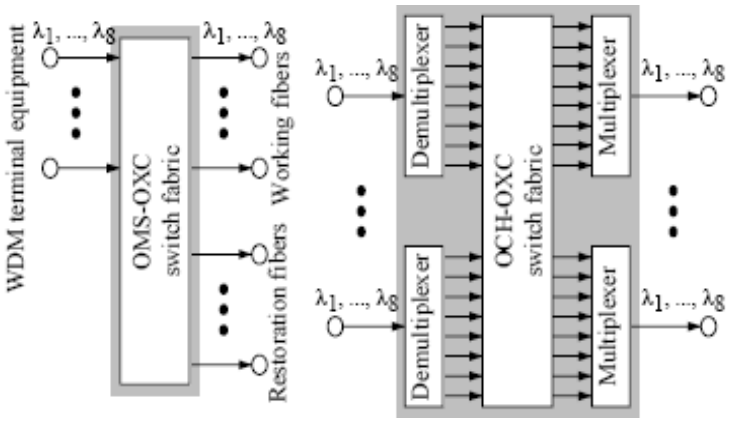


Fig.11 Diagrama de un OMS-OXC y OCH-OXC

2.6.2 Pasado, presente y futuro de los OXC.

Los sistemas OMS-OXC has sido los primeros desarrollos. Sus características básicas están descritas en la tabla 2.2.

Términos	Características
Topología	Malla/anillo
Tolerancia a fallo	Fibra cortada
Funcionamiento	Restauración por conmutación de fibra
Número de nodos	5
Fibras/nodos	16 máx.
Tiempo de restauración	<150ms.

Tabla 2.2. Características técnicas de un OXS

Una de las características de estos sistemas es que la restauración de los enlaces entre nodos es a base de la conmutación de fibras, eso quiere decir que, durante un funcionamiento normal, tendremos fibras sin usarse de soporte a un fallo por corte. Por otro lado las topologías de este tipo de redes son anillos, actualmente sabemos que las redes malladas son las más eficiente.

Una de las características más interesantes de los OXC son las bajas pérdidas de inserción. Esta arquitectura reduce el número de elementos de conmutación, quiere decir que la cantidad de partes del dispositivo que manejan el rayo entrante en el OXC es pequeña dispersando poca energía.

Otra de las características interesantes de los nuevos equipos es la gran velocidad de restauración en caso de fallo. Estamos hablando de quizá 150ms pasarán desde que el corte es realizado a que la red ha vuelto a converger. La restauración se consigue mediante la detección de falta de portadora en uno o varios puertos, después el equipo que lo detecte enviará un comando a los demás OXC's, que empezarán a moverse para poner en servicio las fibras restantes.

Los trabajos presentes llevan a los desarrolladores a buscar poder llevar un haz de luz de un puerto de entrada a otro cualquiera de salida del OXC, pero también la capacidad de poder extraer una longitud de onda de un puerto e insertarla en otro puerto. Esto antes ya se podía hacer rudamente con los OADM's pero buscamos poder incorporar esta característica en una matriz de conmutación automatizada.

En definitiva el futuro de los OXC está en conseguir que una red óptica sea totalmente transparente y ello lo conseguiremos gracias a los PXC (photonic cross-connect). En esta arquitectura de conmutación los datos de señalización estarán en el dominio óptico.

2.7 RWA

Hemos hablado de protocolos que control, de gestión de búsqueda de rutas. También hemos hablado que los dispositivos que trabajan en las redes ópticas semitransparentes son los OXC y estos conmutan *lamdas*.

Pero que protocolo contempla la elección de la *lamda* de salida. Si decidimos que un rayo entrante debe salir por un puerto, ¿Cuál de las *lamdas* de ese puerto utilizaremos? ¿Quién gestiona el conocimiento de cual de ellas está libre u ocupada? ¿Cuál es la mejor *lamda* de salida? Las respuestas se encuentran en la política de asignación de *lamdas*.

2.7.1 Introducción a la reserva de *lamdas*

Establecer los caminos ópticos incluye seleccionar los nodos por los que se va a pasar la portadora de información, más una longitud de onda que hará de portadora.

Como nunca dispondremos de los recursos necesarios para poder dar servicio al 100%, la consecuencia es que hay que compartir los enlaces.

Hay dos caminos fundamentales para la reserva de *lamdas*. El primero es que la información de todas las *lamdas* disponibles sea propagada por la red hasta el destino, el cual seleccionará una de las *lamdas* de acuerdo con la política de asignación de *lamdas* (tales como la primera

libre, aleatorio,…), luego un mensaje publicando la *lamda* elegida es transmitido por la red, los nodos implicados en la reserva liberan las otras *lamdas* excepto la elegida. La reserva se ejecuta de destino a origen, este método se llama Forward Reseration Protocol (FRP).
Una segunda manera consiste en que el nodo origen recoge la información de los mensajes que se difunden por la red acerca de la ocupación de las *lamdas*. Elige todas las *lamdas* disponibles y envía ese mensaje al destino. En este caso no se hace ninguna reserva con lo cual tampoco habrá una futura liberación. Cuando el mensaje llega a su destino, éste elige una de ellas y reenvía un mensaje de retorno con la *lamda* seleccionada al origen, por el camino cada uno de los nodos implicados reservará la mencionada *lamda*.

2.7.2 El problema de RWA

Establecer los caminos ópticos implica seleccionar un camino y luego una *lamda* a través de él. Si el nodo es incapaz de conmutar o convertir *lamdas*, solo es posible establecer un enlace si está libre la misma *lamda* en todos los enlaces del camino.
Esta es la esencia del problema que tratamos de investigar. Conseguir optimizar los recursos de la red. Es necesario seleccionar la ruta óptima y una *lamda* que consiga minimizar la probabilidad de bloqueos, o que un usuario no pueda cursar su llamada en un momento dado. Este es el problema llamado RWA (Routing and Wavelength Assigment).
El problema de RWA es una de las partes principales del diseño de redes ópticas con conmutación de *lamdas*. La pregunta que viene es ¿Cuál es la mejor manera para conseguir optimizar los recursos?
A través de una herramienta de simulación podemos virtualizar una red y sacar conclusiones de las decisiones y elecciones de diseño.

2.8 Introducción a la ingeniería de tráfico.

Antes de entrar de pleno en el análisis del simulador haremos una introducción sobre los recursos matemáticos que se utilizan para cuantificar el número de llamadas que nuestra red debe gestionar. No son funciones matemáticas específicas para estudiar redes de comunicaciones, pero son muy útiles en nuestro campo.

2.8.1 .Erlangs y distribuciones de Poisson: modelo descriptivo de las necesidades reales.

La persona que primero estudio como se comportaba el usuario a la hora de utilizar un servicio fue A.K.Erlang. Él estaba interesado en calcular cuantas líneas telefónicas eran necesarias desde su pueblo hasta el resto del mundo para permitir a los habitantes llamar sin recibir muchas señales de línea ocupada.
Erlang estudio tres posibles situaciones:

- Algunos usuarios encuentran la línea ocupada y cuelgan.
- Algunos usuarios encuentran la línea ocupada, cuelga e inmediatamente prueba a llamar otra vez.
- Algunos usuarios reciben un mensaje de "por favor, permanezca a la espera" y esperan a que los atiendan.

El caso más simple es el primero y generalmente se denomina Erlang B.

En cambio el caso tercero, en que nos permiten permanecer en espera, se denomina Erlang C y tiene el caso adicional de calcular cuanto tiempo tendremos a los usuarios a la espera hasta que sean atendidos.

Los casos Erlangs propuestos anteriormente son extrapolables a muchas otras situaciones como una cola de personas en una cajera de supermercado, el número de troncales que tienen que unir dos ciudades o la cantidad de líneas que tiene que instalar una empresa para su departamento de ventas o de atención al cliente.

La cantidad de tráfico en un sistema está medido en unas unidades llamadas Erlangs. Un Erlang es la ocupación que se les da a los canales de comunicación contratados durante una hora. Por ejemplo, un canal de comunicaciones utilizado durante 30 minutos cada hora significa 0,5 Erlangs.

Dos canales ambos dos utilizados durante 30 minutos cada hora, es un total de carga de 1 Erlang.

Por otro lado dos canales de comunicación ocupados uno de ellos durante 20 minutos y otro durante 30 minutos hacen un total de 0,83 Erlangs.

Los ejemplos anteriores cuantifican de manera numérica el uso que hace un usuario de los recursos de una red, pero lo que sería de más utilidad a los que ofrecen servicios o necesitan contratarlos es poder cuantificar sus recursos según sus necesidades observadas. Las necesidades son resumidas actualmente con las palabras tráfico ofrecido, en otras palabras, es el tráfico que los usuarios aportan al sistema. Lo podemos calcular como:

$$erlangs = \frac{duracion\,de\,la\,llamada}{media\,entre\,llamdas} \qquad \textbf{(2.1)}$$

En un caso en que la media de la duración de una llamada es 1 minuto y la media de llegada de llamadas es también 1 minuto el tráfico ofrecido es 1 Erlang. Si pensamos otra vez en el ejemplo y pudiéramos asegurar que las llamadas llegan cada minuto y duran exactamente un minuto, eso quiere decir que el tráfico es 1 Erlang, con un canal habría suficiente para gestionar las llamadas durante 1 hora.

Pero el problema es que nadie puede asegurar esa clase de tráfico en la vida real. En el tráfico real de una red muchas veces cuando una llamada está en curso, otra trata entrar en el sistema y se encuentra un bloqueo de tránsito y no puede ser cursada, la idea más intuitiva que se nos puede ocurrir es aumentar el número de canales para no permitir bloqueos en nuestra red, pero nos damos cuenta que a veces los dos canales no están ocupados con lo cual estamos perdiendo dinero porque el mantenimiento de un canal tiene coste o el contrato de más líneas a un operador aumenta la factura.

Por tanto el gran conflicto de la ingeniería de tráfico: compartir el medio, teniendo el máximo rendimiento con el mínimo bloqueo.

Aquí entra en juego los estudios que hizo el sr. Erlang. Resolver la pregunta: ¿Cual es la probabilidad de que una llamada sea bloqueada? ¿En que circunstancias?

Los cálculos sobre tráfico son a menudo basados en Busy Hour Traffic (BHT). De todas formas hay otra unidad para cuantificar el tráfico es un CCS, significa 100 minutos de llamada, así 3600 segundos de llamada son 36 CCS. A veces los CCS son utilizados en vez de los Erlangs. Estos datos son importantes porque muchos calculadores o funciones para calcular tráfico se expresan en estas unidades.

Por otro lado otras cabezas pensantes hicieron estudios matemáticos sobre probabilidad, que se conjuntaron con las ideas de A.K.Erlang para desarrollar las ecuaciones sobre la ingeniería de tráfico.

Las distribuciones de Poisson son expresiones matemáticas de probabilidad que se utilizan a menudo como modelo de llamadas entrantes.

Tratemos de razonar que modelan. Imaginemos que tenemos llamadas que son realizadas más o menos cada 6 segundos. Como hemos dicho antes la probabilidad real de que las llamadas consecutivas tengan un espaciado constante es rarísima. Si así fuera con una multiplexación TDM tendríamos arreglado nuestro problema de ocupar un canal eficientemente.

Por tanto la pregunta es:

1.-Qué probabilidad de que se repita una segunda llamada, solo un segundo después de la primera. Si tenemos 6 ranuras temporales existe 1/6 de probabilidades = 0,166667.

2.-Seguimos, ahora. Cuál es probabilidad de que haya una llamada al segundo 2 de la ranura. (Matemáticamente la probabilidad de que ocurran dos eventos independientes, es la probabilidad de que ocurran los dos a la vez y es el producto de las dos probabilidades por separado.)

En nuestro caso, la probabilidad de que no haya una llamada en el segundo 1 y si en el segundo 2 es:

$$\frac{5}{6}\times\frac{1}{6}=0{,}1389 \qquad \textbf{(2.2)}$$

Siguiendo esta rutina:

3.-La probabilidad que no ocurra el evento 1 (segundo 1º de la ranura de 6) y 2 (segundo 2º de la ranura de 6) y sí que la llamada entre el tercer segundo de la ranura temporal. Matemáticamente:

$$\frac{5}{6}\times\frac{5}{6}\times\frac{1}{6}=0{,}1157 \qquad \textbf{(2.3)}$$

Siguiendo este algoritmo tendríamos una tabla 2.3 como la de abajo, está tabla representa una Distribución de Probabilidad de Poisson. Y ella nos dice que probabilidad hay de que una segunda llamada entre en un sistema antes de que la primera acabe.

Tiempo desde la llamada anterior	Probabilidad de llamada
1 segundo	0,1667
2 segundo	0,1389
3 segundo	0,1157
4 segundo	0,0965
5 segundo	0,0804
6 segundo	0,0670
7 segundo	0,0558
8 segundo	0,0465
9 segundo	0,0388

Tabla 2.3

En otras palabras, siempre que la duración de las llamadas sea más larga que el espacio temporal en que las llamadas van llegando al sistema necesitaremos más de un canal para que no haya bloqueo. Así pues según la formula:

$$erlangs=\frac{duracion\,de\,la\,llamada}{media\,entre\,llamdas} \qquad \textbf{(2.1)}$$

Necesitaremos que el denominador sea mayor que el numerador, (Erlangs <1), para que en con 1 solo canal no haya bloqueos nunca.

Otro dato a tener en cuenta en que las llamadas nunca tienen la misma duración, aunque Erlang encontró que la duración de las llamadas que entran en una red o sistema siguiendo una distribución de Poisson no da resultados de ocupación y/o bloqueo apreciablemente diferentes a un modelo de llamadas de duración constante.

Como colofón a los descubrimientos hechos en ingeniería de tráfico, se escribieron unas tablas de referencia para cuantificar las necesidades de los usuarios. En las tablas 2.3 y 2.4, siguiendo el modelo Erlang B que describimos al principio, tenemos en las filas cantidades que representan el número de canales útiles, T, y en las columnas el tanto por ciento de bloqueo. La interjección de ambos es el número de Erlangs que cumplen.
Por ejemplo:

Nº dispositivos	0,01% Bloqueo
1	0,1111
2	0,5954
3	1,2708
4	2,0454
5	2,8801
6	3,7584
7	4,6662
8	5,5971
9	6,5464
10	7,5106

Tabla 2.4

La tabla 2.4 anterior significa que si tenemos 2 canales para tener una probabilidad de bloqueo del 0,01% o sea perder 1 llamada de cada 100 necesitamos 0.5954 Erlangs.
O leído de otra manera si el tráfico de llamadas en nuestra empresa por la fórmula 2.1 es de 3 Erlangs, si queremos asegurar un bloqueo del 0,01% necesitaremos por lo menos 6 canales.
Para acabar nuestra introducción a la ingeniería de tráfico explicaremos un último parámetro importante para valorar la eficiencia de nuestra red. Es el factor de utilización.

2.8.2 Factor de utilización

Como su nombre casi describe, este factor es como de llenos están los canales, en otras palabras, la eficiencia de utilización o rendimiento.

$$F.U = \frac{erlangs \times ratio\ de\ no\ bloqueo}{numero\ de\ canales} \qquad \textbf{(2.5)}$$

En un ejemplo:
Tenemos una instalación con 30 canales de comunicación que pueden gestionar 20,34 Erlangs con un 1% de bloqueo, por tanto el factor de utilización es:

$$\frac{20{,}34 \times 0.99}{30} \times 100 = 67\% \qquad \textbf{(2.6)}$$

En este caso los 30 operadores estarán ocupados el 67% del tiempo.

Capítulo 3

Estudio del Simulador

3.1 Análisis previo: Importancia de la simulación

En el momento actual de las comunicaciones la evolución constante es un orden que no se puede cortar ni detener. El mercado es exigente y compulsivo, las necesidades de hace un tiempo dejaron de crecer linealmente para hacerlo exponencialmente, aun se está exprimiendo todo el jugo que se puede sacar de una pastilla de silicio y eso engrandece enormemente el marco de productos que ofrecer.

El ser humano es comunicativo, tiene la necesidad y el deseo de entrar en contacto con el entorno. Pero hoy en día ese efecto ha ido mucho más allá. Los principales países desarrolladores de tecnología, se encuentran en el primer mundo y pertenecen a la sociedad de consumo. La sociedad de consumo es el gran monstruo desbocado que ofrece, ofrece y ofrece, y el ser humano es un animal con una capacidad de absorción descomunal, que se crea a si mismo necesidades.

En consecuencia, las empresas que han dedicado sus esfuerzos a explotar el mundo de las comunicaciones, empezaron a prestar servicios de todo tipo a través de las redes de información para satisfacer parte de esas necesidades humanas.

Por otro lado, la información es poder, con lo que gobiernos, estados, bancos, centros de información, no paran de invertir en recursos para mejorar y afianzar las redes de información.

Existen grandes intereses montados alrededor de las telecomunicaciones, el cliente y usuario final cada vez es más exigente y entendido. Exige un servicio más amplio, específico y de mayor calidad. Por lo tanto los operadores de servicios deberás satisfacer todos estos deseos.

No es fácil poder realizar cambios en las tecnologías de la comunicación, primero de todo porque el usuario nos pedirá no perder ninguna de las cualidades ya conseguidas, por otro lado es difícil interrumpir un servicio para llevar a cabo un *upgrade* (actualización), con lo cual nos exigirán tenerlo que hacer en caliente. Y por último y más importante, que sucedería si después de planificar un cambio, desembolsar una cantidad espectacular de capital, realizar

la formación de personal y el montaje de equipos, arrancamos nuestro sistema nuevo, para ampliar las expectativas de explotación y resulta que el sistema no va, se cae o no cumple con el *planning* establecido.

¿Resultado? Catastrófico. Tendremos clientes enfadados, usuarios reclamando, podemos hacernos la idea de lo que significaría tener un ministerio de hacienda o de defensa, totalmente fuera de servicio. Seguramente ningún gerente de ninguna empresa quisiera responder a las llamadas de quejas y denuncias. Y si solo fuera una queja de 4 gritos, pero seguramente la cosa no quedaría ahí.

Por tanto para evitar incidencias las empresas encargadas de desarrollar las tecnologías de la comunicación y ponerlas al servicio del usuario, cada vez más optan por simular y testear sus hipótesis de desarrollo para poder llevar a cabo una planificación que no desencadene sorpresas.

Pudiera ser alguien dedicado a investigar sobre nuevas tecnologías quien estuviera interesado en simular determinadas ideas, para ver el resultado de ciertas hipótesis y llegar a conclusiones, cierto es que a veces las sorpresas son notables, sobre los resultados esperados a los conseguidos.

Puede ser perfectamente un operador de datos, acordando datos como cualquier transmisión digital, ya sea de voz, video, Internet…el que esté muy interesado en saber como evolucionar dentro de su campo.

Necesitarán de herramientas que tratan de acercar la realidad al laboratorio. Sin tener que hacer un desembolso excesivo. Podemos asemejar un comportamiento y gracias sobretodo a la informática podemos programar ordenadores para realizar tareas que simulen una realidad.

Este es el destino de este simulador, ayudar a evaluar el comportamiento de una red semitransparente.

3.2 Análisis de requisitos

Para poder simular una red semitransparente, primero de todo tendremos que aprender a matizar los puntos de los que se compone una red semitransparente, después tener en cuenta estos parámetros y valorarlos, finalmente descomponerlos en rutinas que se puedan introducir en un entorno de programación.

Anteriormente hemos estudiado los elementos que componen una red semitransparente.

Tendremos una topología de red, como en todas, habrá unos nodos interconectados por algún tipo de medio, y todo ello controlado por algún sistema inteligente.

Hemos recurrido a la informática para poder definir todos los parámetros de simulación. Necesitamos una máquina potente para poder devolver resultados en un espacio razonable de tiempo, es preciso que nuestras simulaciones no se alarguen excesivamente.

El simulador ha sido acotado a redes ópticas con nodos regeneradores (conmutadores), eso ha significado modelar los cables de interconexión de las redes ópticas, que son ni más ni menos que cables de fibra óptica. De ellos, diferentes parámetros se han analizado para ver que interés pueden tener para nuestro simulador, quiero decir, que parámetros como la dispersión y la atenuación en un simulador que resuelve el problemas del bloqueo de tráfico, quizá no sea un parámetro relevante a tener en cuenta y lo hemos descartado. Los cables de fibra van conectados a unos nodos que después de un estudio dentro del mercado de desarrollo actual de componentes para redes ópticas, nos decidimos por conmutadores ópticos de *lamdas*.

No es un capricho de nadie haber elegido este tipo de nodo para el simulador, hay buenas razones. La actual evolución de las redes ópticas, como podemos ver en la **figura 3.1,** es hacia equipos automáticos de conmutación de *lamdas*. Podrían, ser evaluados otros elementos ópticos como los multiplexores o los ADM, incluso los ROADM, los más próximos a los OXC, pero después del análisis del comportamiento de esos equipos, me di cuenta de que un OXC, es la consecuencia final de la evolución de los otros, por consiguiente si cualquier persona quisiera simular redes con otros equipos simplemente tendría que acotar las características del equipo de referencia del simulador.

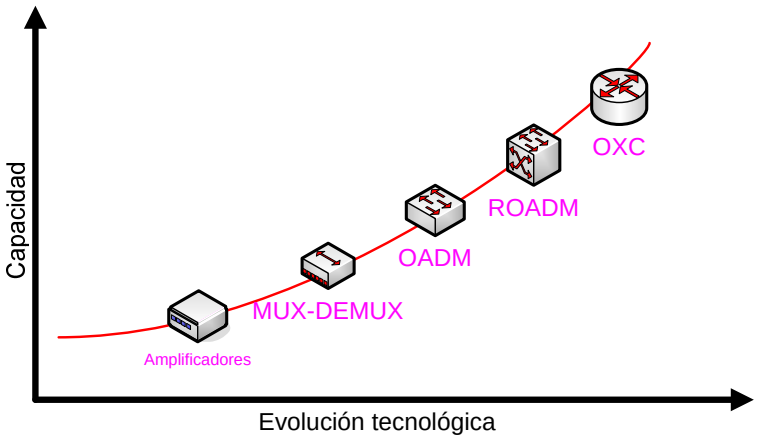


Fig.3.1 Representación de la evolución de los dispositivos ópticos

Esta idea se ha mantenido en todo el simulador, tratar la información analizable lo más genéricamente posible para que el simulador se pudiera amoldar a las múltiples configuraciones que se pueden plantear.

Mi idea era realizar una herramienta potente que tuviera muchas posibilidades y que no redujera las expectativas del usuario que lo utilizara, por culpa de tener funciones limitadas.

Se ha tratado de dejar las funciones que componen el simulador muy abiertas. Esta capacidad permite que el simulador pueda tratar con cualquier topología de red. Esta fue la primera premisa sobre la que trabajé.

Otro de los apartados destacados es la interficie de entrada de datos. Los datos se recogen de ficheros con formato. La ventaja de trabajar con ficheros es que sólo nos hace falta invertir una vez tiempo, en pasar los datos a nivel informático para poder introducirlos en el simulador. Tanto la topología de red como las listas de llamadas son ficheros que se pueden reutilizar o cambiar sin necesidad de crearlos de nuevo, simplemente tocando los valores que nos interesan. Por otro lado también se incorporó algo de formato en los ficheros para facilitar a los responsables de introducir los datos a que la tarea sea más sencilla e intuitiva.

3.3 Primeros pasos:

A través del siguiente grafico trataremos de hacer una descripción visual de los procesos de mi simulador.

Una vez iniciado el simulador, podemos configurar los parámetros deseados en el menú de configuración, para después pasar al módulo de ejecución.

Dentro del módulo de ejecución el programa entra en una rutina que será interrumpida para introducir los datos de las llamadas. Su origen, su destino y el fichero con la lista de llamadas. Este proceso creará las variables para empezar la simulación y el programa empezará a leer las llamadas de los ficheros introducidos a través de una rutina *round-robin* sin *quantum*. Cada lectura es una llamada, el programa buscará un camino y una *lamda* y apuntará el resultado de la búsqueda en la variable resultados. En la variable “L_Fich” almacenamos los datos para establecer las llamadas y en “Core” los datos de la ocupación de los enlaces.

El proceso de simulación puede ser parado en cualquier momento e introducir nuevas listas de llamadas con origen-destino. El simulador seguirá la simulación incorporando las nuevas listas.

El simulador ejecuta el proceso que se llevaría a cabo en una red real, con "timing" que representa los momentos en que la red no converge por la latencia de los procesos de actualización de las bases de datos entre los nodos.

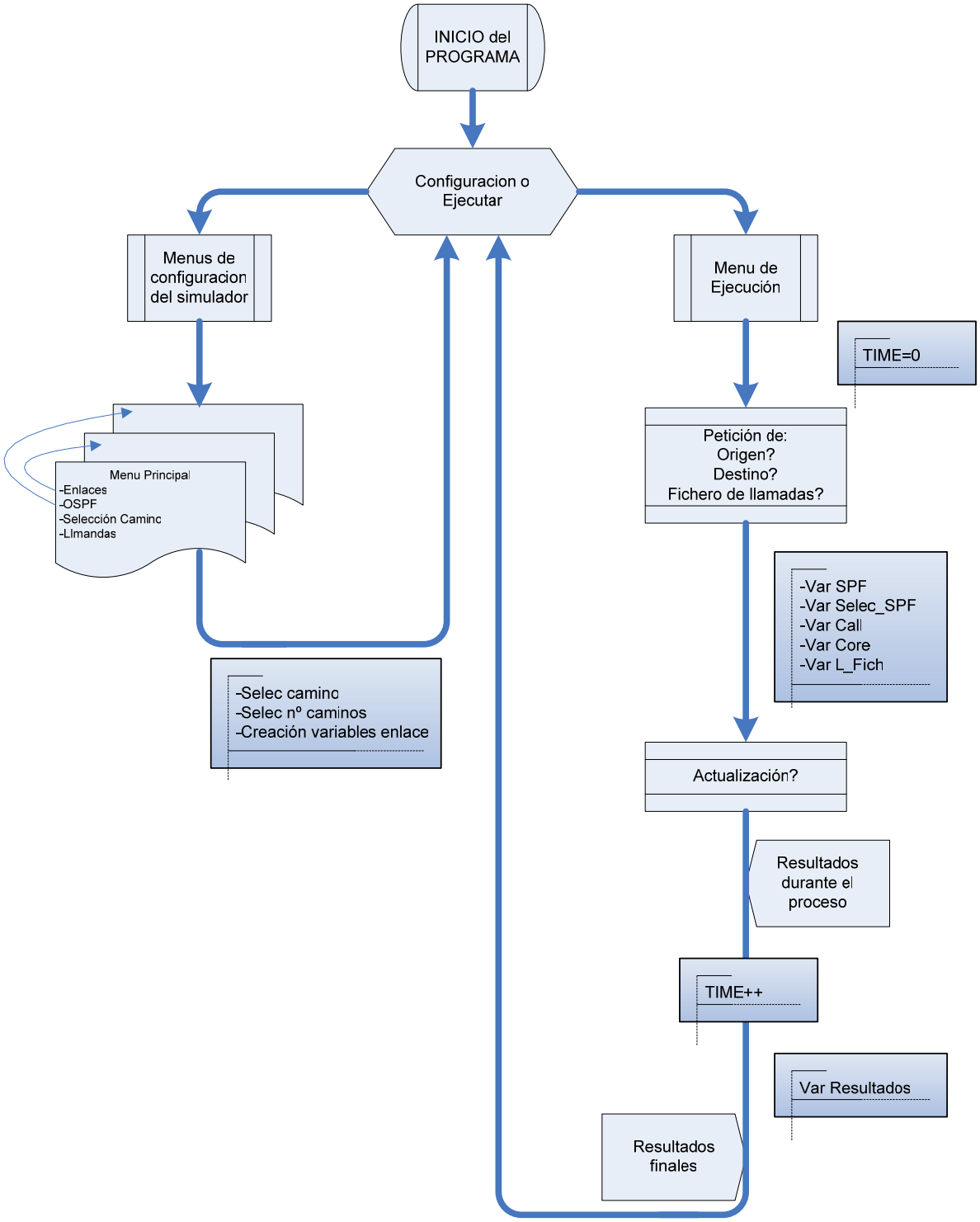


Fig. 3.2 Procesos del simulador

3.4 Definición de parámetros

A continuación hablaremos sobre la serie de parámetros que se han tenido en cuenta a la hora de confeccionar el simulador. Desde parámetros externos configurables por el usuario hasta los internos propios del programa utilizados para la realización de la simulación pero que

resultan transparentes en modo de ejecución. En el redactado se marcarán ambos, externos e internos, para su mejor identificación.

La descripción de a continuación resultará útil, complementado con un anexo: **manual del usuario.doc**, para saber como actúan cada uno de los parámetros configurables del programa.

Vamos a ir paso a paso para hacer una descripción más clara y ordenada de las partes, trataremos de seguir las capas lógicas de una pila de protocolos de comunicación. Aunque en algún momento quizá será imposible seguir la jerarquía y se tenga que saltar de una capa a otra.

Empezando por la parte más baja tenemos la capa de transporte, que esta compuesta por los enlaces de la red. Los enlaces son ópticos, por tanto hemos tenido que modelar los parámetros relevantes de una fibra óptica para nuestro simulador.

a. Primero de todo un inciso para explicar las llamadas (Calls):

Las listas de llamadas representan una línea temporal de requerimiento de comunicaciones por un usuario. De otro modo, cada vez que un usuario quiere comunicarse con otro. Por tanto las llamadas quedan definidas por el momento temporal cuando se inician (Interarrival time) y el tiempo que la llamada está activa (Hold time).

De manera gráfica:

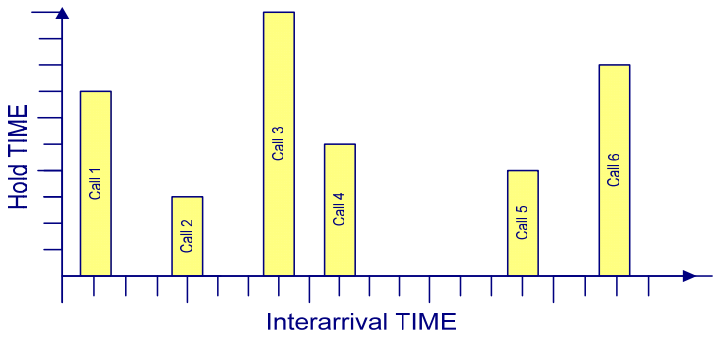


Fig. 3.3 Descripción de las listas de calls del simulador]

Es un parámetro externo y en mi proyecto hemos utilizado el modelo descrito anteriormente para escribir las listas de llamadas.

Antes hablamos de Erlangs y distribuciones de Poisson como modelos matemáticos para simular el tráfico en una red. Los Erlangs cuantifican la ocupación de nuestra red, por tanto serán el valor que utilizamos para evaluar la carga de nuestra red. Se compone de dos variables: longitud media de la llamada, que nombro como *Hold Time* y media entre llamadas que yo nombro *Interarrival Time*. Por consiguiente las listas de llamadas tienen una columna con cada una de estas dos variables.

Por otro lado elegimos la distribución de Poisson como modelo para establecer la entrada de llamadas. Los *calls* que irán entrando en los nodos siguen la fórmula de una distribución de Poisson.

b. El modelo de fibras.

Parámetro externo, que se compone de dos variables fundamentales y son el **número de *lamdas* por fibra** así como el **número de fibras** que componen cada enlace. En mi caso hemos dejado totalmente configurable este parámetro para poder dimensionar los enlaces con el tamaño deseable tanto en número de fibras como en número de *lamdas*. Tomé la decisión de que todos los enlaces tuvieran el mismo tamaño, por agilizar la simulación y simplificarla aunque puede ser un valor a tener en cuenta en un futuro ya que en las redes actuales, los cables troncales que enlazan nodos de *core* no siempre tienen el mismo tamaño, en otras palabras los enlaces no siempre están hechos con cables con el mismo número de fibras, y los equipos que forman esos nodos no siempre tendrán las mismas capacidades, quiero decir, a medida que se renueven los equipos por fallo, planificación o desarrollo, la capacidad de ingresar *lamdas* en una fibra crecerá y se dará la circunstancia que en algunos momentos la red funcione con equipos que pueden multiplexar más *lamdas* que otros.
Aunque la heterogeneidad de la red es un valor importante no es tan determinante, ya que por software podemos controlar la multiplexación de *lamdas* y físicamente desconectar fibras, para conseguir hacer la red con todos los enlaces iguales y esperar a haber acabado de completar el replanteo de toda nuestra red de fibra, para reconfigurar y reconectar los equipos y dejar la red idéntica a la que hemos podido simular.
Vamos introducido otro parámetro, este es interno.

c. Los enlaces.

Cada enlace se compone de un número de fibras con un número de *lamdas*, origen y un destino, y si el enlace permite regeneración o no. Tanto el origen y destino se podrán definir a través de un fichero que introduciremos en el simulador como topología de red. Este fichero contiene el número de nodos, como están interconectados y si cada una de las conexiones permite regeneración de *lamdas* y la cantidad de estas.
Siguiendo por el árbol de procesos, nos encontramos con los parámetros de los protocolos para simular las llamadas, en el apartado de ejecución.
Se generan dos variables básicas internas:

d. Core y L_fich,

La variable "Core" es utilizada para almacenar los valores de los enlaces de toda la red, necesarios para cada una de las llamadas introducidas, en "L_fich" almacenamos las llamadas que se van leyendo del fichero externo así como los caminos y *lamdas* que el programa ocupa para cada una. La finalidad de L_fich es representar los datos que queremos simular.
La relación "Core" → "L_fich" es 1 a muchos, ya que damos la posibilidad de tener ejecutándose varias lista de llamadas con el mismo origen y destino. La finalidad de esta decisión, dejar el programa abierto a exigencias.
Cada vez que el simulador procese una llamada y ocupe una *lamda* del camino elegido (origen→ destino) accederá a "Core" y modificará el valor de las *lamdas* de cada uno de los enlaces que compone el camino. Por otro lado la variable "L_fich" marcará la llamada como procesada y pasará a la siguiente, y así hasta que todas las llamadas leídas hasta el momento con un parámetro interarrival sea menor al TIME actual.

e. TIME.

Otro de los parámetros internos importantes es **TIME**, el cual acabamos de nombrar, este valor es nuestra referencia temporal. Como hemos descrito al principio del punto las listas de llamadas están en base a una línea temporal y nuestro simulador necesita tener una referencia, un *clock* interno, para poder ir sincronizado con las líneas temporales de cada una de las listas de llamadas.

f. Nodo.

La parte más importante del simulador es el modelo de **nodo** diseñado. Como hemos hablado al principio de esta memoria, el futuro está en las redes ópticas con conmutación de *lamdas*, por tanto nuestra referencia para el simulador es un nodo con comportamiento de cross-connect. El modelo de nodo implementa la idea de realizar una red independiente y escalable, permitiendo cualquier topología. Los nodos tendrán una copia del estado de la red cada cierto tiempo. En una red real el tiempo de convergencia es un parámetro crítico.
Será en el nodo desde donde se harán las solicitudes de las llamadas, las listas de llamadas se aplican aquí, los usuarios se conectan a la red a través de los nodos, después se elegirá un camino y una *lamda*. Más tarde toda esa información será procesada por el simulador y se anotarán los resultados. Cada nodo permite ejecutar múltiples listas con destinos diferentes.

3.5 Definición de topologías

Si queremos intercambiar información entre dos nodos, podemos emplear diferentes modos de conexión.

- · Enlace punto a punto: consiste en unir mediante un medio físico los dos ordenadores que queremos conectar. Es una estructura que complica el compartir información cuando se amplia una red de este tipo.
- · Interconexión total: consiste en unir todos con todos, cuando el número de nodos es alto es totalmente impracticable, aunque sería rápido, fiable y seguro.
- · Red de comunicación: se trata del conjunto de medios necesarios para asegurar la transferencia de información entre dos nodos.

Para interconectar varios dispositivos se necesita disponer de un medio físico que permita establecer dicha conexión: este medio físico, en la mayoría de los casos es un cable, que puede ser de diferentes naturalezas, aunque también es posible utilizar como medio físico la radiodifusión.

Existen muchas formas de catalogar una red, de acuerdo al espacio físico que ocupa, a la topología que tiene implantada, la finalidad… La clasificación más difundida quizá sea la que se basa en catalogar a las redes teniendo en cuenta el espacio físico por las que están distribuidas. Como ejemplo: LAN; MAN y WAN.

La topología de una red define la distribución de los medios físicos de interconexión y los dispositivos que interconectan, es decir la topología de red es el mapa de distribución de sus componentes.

Es importante no confundir el término topología con el de arquitectura. La arquitectura de una red engloba: la topología, el método de acceso al medio y los protocolos de comunicaciones.

Es necesario establecer la diferencia entre una topología física y una lógica. La física es de la que hemos hablado hasta ahora. La topología lógica es la forma como trabaja una topología física.

- Tipologías físicas: Bus, Anillo, Estrella, Mallada,Arbol
- Tipologías lógicas: Anillo-estrella, Bus-estrella, Mallada.

3.6 Elección del entorno de programación

Para poder diseñar el simulador he considerado que la herramienta más efectiva será un entorno de programación. Es necesario construir un programa para procesar, un ordenador y los datos para la simulación.

He estado barajando varias opciones como entornos de programación, mi decisión final fue utilizar lenguaje C.

Utilicé un entorno de programación que permite programar en C, como es Visual Studio de Mircrosoft.

El lenguaje C es un lenguaje de alto nivel. Muy utilizado hasta el día de hoy por su gran versatilidad hasta la llegada de los lenguajes orientados a objetos.

C es apreciado por la eficiencia del código que genera y es el lenguaje de programación más popular para sistemas. También se puede utiliza para crear aplicaciones. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos. Proporciona facilidades para realizar programas modulares y/o utilizar código o bibliotecas existentes.

De la misma forma, es muy usado en aplicaciones, industriales (predominan también todo lo que se refiere a simulación de máquinas de manufactura).

El mayor problema que presenta el desarrollo con el lenguaje C frente a los lenguajes de visuales es su diferencia de velocidad de desarrollo y los resultados visuales, siendo menos atractivo al usuario el desarrollado en C. En cambio, los programas terminados presentan una mejor utilización de los recursos hardware.

La mayor parte de los problemas en C son de seguridad, deriva del hecho de que han sido realizados en C: al no disponer de sistemas de control y depender casi exclusivamente de la experiencia del programador, para el control de errores.

Muchas veces quedan algunas entradas de información al programa sin comprobar, provocando la sobre escritura de código en la pila del programa o referencias a variables no inicializadas, incluso referencias zonas de memoria prohibidas. Será la precaución y la buena planificación la que permitirá controlar los agujeros de seguridad, y sabiéndolo hemos tratado tenerlo en cuenta.

La documentación difundida en cualquier tipo de soporte sobre C, es enorme, es por todo los detalles expuestos que me decidí a utilizar este lenguaje y no otro.

Me plantee utilizar el entorno C++, orientado a objetos y sobre todo a aplicaciones visuales de gestión, pero la parte importante de este proyecto es el desarrollo del núcleo del simulador, dejaré para más adelante decorarlo con un entorno gráfico agradable y más familiar como son las ventanas.

3.6.1 Algunos conceptos.

3.6.1.1 Descripción programación modular

La lectura del código fuente de un programa implica realizar el seguimiento del flujo de ejecución de sus instrucciones. Evidentemente, una ejecución en el orden secuencial de las instrucciones no precisa de mucha atención. Pero los programas contienen también instrucciones condicionales o alternativas e iterativas. Con todo, el seguimiento del flujo de control puede resultar complejo si el código fuente ocupa más de lo que se puede observar (por ejemplo, más de una veintena de líneas).

Por ello, resulta conveniente agrupar aquellas partes del código que realizan una función muy concreta en un subprograma identificado de forma individual. Es más, esto resulta incluso provechoso cuando se trata de funciones que se utiliza en diversos momentos de la ejecución de un programa.

Cada uno de estos bloques de código se denomina módulo. A menudo estos módulos se pueden agrupar dentro de ficheros.

3.6.1.2 Variables

Entendemos por ámbito de una variable el lugar, dentro de un programa, en el que esta variable tiene significado. Hasta el momento todas nuestras variables han tenido como ámbito todo el programa, y quizá ahora no es sencillo hacerse una idea intuitiva de este concepto; pero realmente, no todas las variables están “en activo” a lo largo de todo el programa.

Reglas de ámbito:

1.- Un identificador se puede utilizar en el programa en el que está declarado y en todos los subprogramas de él.

2.- Si un identificador declarado en un programa se redeclara en algún subprograma interno Q, entonces cualquier referencia a dicho identificador en Q(o algún subprograma de Q) utilizará el identificador declarado en Q y no el declarado en P. (Prevalece el más interno).

Tipos de variables:

- **Variables static:** cuando queremos que una variable sea creada estática, la asociaremos a esa nomenclatura. Esa variable puede ser local, por tanto su ámbito será local, y sólo podrá ser usada cuando se estén ejecutando sentencias de su ámbito; pero su extensión será la misma que la del programa, y siempre que se vuelvan a las sentencias de su ámbito, la variable estará lista para ser usada. Cuando terminen de ejecutarse las sentencias de su ámbito esas posiciones de memoria no serán accesibles, porque estaremos fuera de ámbito, pero tampoco podrá hacerse uso de esa memoria

para otras variables, porque la variable estática seguirá "viva" y en esa posición de memoria sigue almacenando el valor que quedó de la última vez.

- **Variable Local:** variable declarada dentro de un subprograma y, por tanto, sólo disponible durante el funcionamiento del mismo
- **Variable Global:** variable declarada en el programa principal y, por ello, pueden ser utilizadas por el programa principal y por todos sus subprogramas

La comunicación entre un programa y un subprograma debe realizarse a través de parámetros, y no de variables globales.

Paso del valor de las variables en una función:

·**Por valor**: únicamente nos interesa el valor, no las modificaciones que pueda tener dentro del subalgoritmo. Se trabaja con una copia del valor pasado. Son parámetros unidireccionales, que pasan información desde el algoritmo al subalgoritmo. Puede ser cualquier expresión evaluable en ese momento.

·**Por referencia**: se pasa una referencia a la posición de memoria donde se encuentra dicho valor. Se utilizan tanto para recibir como para transmitir información sobre el algoritmo y el subalgoritmo.

3.6.1.3 Memoria dinámica: malloc() i free()

Hay dos métodos a través de los cuales un programa en C puede guardar información en la memoria de la computadora. El primero es del que hemos hablado antes, a través de variables. El segundo método, el programa puede utilizar las funciones de asignación de memoria dinámica "malloc()" para reserva y "free()" para liberar. Con estos métodos, un programa asigna espacio para almacenar información. La computadora toma la memoria para satisfacer una petición de "malloc()" de la zona de memoria libre, empezando justo debajo de las variables globales y creciendo hacia la pila. Como se puede suponer, en casos claramente extremos, es posible que se nos limite el espacio asignable por tanto es muy importante siempre, después de un "malloc()", comprobar que nos ha devuelto un puntero correcto. Si no tenemos la precaución de controlar esto, el puntero puede ser no utilizable y volver el programa inestable.

Una gran funcionalidad, que da mucha potencia a un lenguaje como C, es la capacidad de poder definir nuestras propias variables, acorde con las necesidades del programador y a su total medida. Son los *structs*, se podrían definir como estructuras de datos que contienen variables y/o punteros, los cuales se pueden anidar. En nuestro caso ha sido un punto importante para realizar listas dinámicas de datos anidadas por punteros, las cuales podemos recorrer de principio a fin, incorporando y eliminando elementos de la lista, con lo que solo utilizamos la memoria física del ordenador que necesitamos.

3.6.2 Análisis de la aplicación

El trabajo sobre el simulador ha sido escalado. Se empezó con una idea y se han ido modificando detalles a medida que se observaban resultados. La primera idea que se llevó a cabo fue implementar un protocolo de enrutamiento, para ello se pensó en un protocolo con el que pudiéramos descubrir los caminos posibles hacia un destino. Se tomo como referencia el OSPF (Open Shortest Path First - RFC 1247) pero con algunas modificaciones en base a las premisas de nuestro simulador.

He tenido en cuenta con el mayor de los aspectos que nos brinda una red óptica que es el ancho de banda enorme que nos ofrece. Por tanto he considerado todos los enlaces con una métrica igual, a diferencia de un algoritmo OSPF estándar en que el valor más importante es la *métrica* que evalúa el *estado del enlace*. Por otro lado igualar la métrica de todos los enlaces nos permite apuntar mejor hacia el enlace que se bloquee más, ya que en igualdad de condiciones el enlace más bloqueado es el más cargado. Otra diferencia con OSPF es que mi protocolo da la posibilidad de buscar todos los caminos entre un origen y un destino, ordenados empezando por el más corto, habilitando el balanceo de carga. Punto que OSPF solo nos permitirá si encuentra duplicidad de caminos.

A mi protocolo lo llamaré SPF.

Protocolos de rutina como OSPF permiten la posibilidad de que haya *multipath* entre un origen y un destino, eso significa que si al construir el árbol de adyacencia para construir la ruta al destino, se encuentra más de un camino el protocolo almacene cada uno de estos caminos y balance la carga entre ellos. Este detalle lo hemos tenido en cuenta al realizar el protocolo, pero con la diferencia del OSPF estándar, que se para cuando llega al destino, el protocolo SPF permite especificar cuantos caminos queremos encontrar.

Después de esto vino plantearse como elegir uno de entre todos los caminos que nos entrega la función anterior.

Repasando como elegir un camino se plantean varias opciones la primera es elegir uno de ellos aleatoriamente, otro de ellos seria balanceando la carga por cada uno de ellos.

Más adelante desarrolle un protocolo con un procesado del estado de la red, este protocolo será el que elija el camino menos ocupado y más tarde a raíz de las primeras simulaciones desarrolle un camino que combinaba la idea de camino menos ocupado con el balanceo.

Después de haber elegido un camino quedaba resolver el elegir una *lamda* libre para cada *lightpath*. Para ello me plantee elegir la lamba utilizando las mismas ideas con las que elegí el camino (aleatorio, balanceado, menos ocupada), pero me acabé centrando en desarrollar uno de ellos. Desarrollé un protocolo que buscaba la *lamda* menos ocupada a lo largo de todo

el camino. Dejo para más adelante desarrollar otro protocolo en que elija la *lamda* a través de un balanceado entre las disponibles.
Una vez solucionado el *RWA* del simulador entraré a hablar del punto más importante del proyecto: son los sistemas regeneradores (conmutadores) del *lightpath*.
Como hablamos anteriormente actualmente se siguen desarrollando equipos de conmutación de *lamdas*, los OXC, como nodos para las redes ópticas semitransparentes y más adelante las transparentes. Estos permiten, gracias a su matriz de conmutación, mandar las tramas que transporta la *lamda* y que entran por una fibra, a otra *lamda* de salida, y si es necesario, por otra fibra.
Este efecto de cruce entre *lamdas* y fibras ha sido incluido en el comportamiento del simulador, permitiendo ver los efectos de una red óptica regenerada y otra que no lo es.

3.6.3 Módulos

Siguiendo las directivas redactadas antes, la programación del simulador se ha realizado con módulos, cada parte del programa está contenido en un módulo, así pues tenemos agrupadas las funciones para crear los caminos, como también las funciones de selección de camino. Otra de las utilidades de haber utilizado la programación modular es la posibilidad de extraer nuestro código a otros programas e incorporar más funciones al simulador en un futuro. Todo ello sin tener que modificar prácticamente el flujo del programa principal.
Cada módulo contiene las funciones necesarias, de igual manera las variables necesarias son definidas por defecto en cada módulo, con la posibilidad de ser modificadas externamente.
Más profundamente, el valor *extern* delante una variable nos da la opción a poder dimensionar una variable que se utiliza localmente en las funciones del módulo pero con la opción poder asignarle un valor fuera del módulo. Así pues los módulos no son herméticos sino que son configurables ciertas variables fuera del módulo.
A continuación describiremos los módulos y una breve explicación de sus principales funciones.
Casi todos los módulos trabajan con memoria dinámica con los cual todos estos módulos gozan de una función “insertar”, y otra “eliminar”.
En la función “insertar” se le pasa el puntero de la lista de variables *struct,* que contiene una llamada “malloc()” con comprobación de puntero apto, para agregar memoria a la lista y poder ocuparla con los valores correspondientes. A la vez que se crea una función insertar, se crea otra eliminar, para poder liberar toda la memoria utilizada.
Para la función “insertar” ha habido dos estrategias a utilizar según me convenía:

- 1.- estrategia de inserción de variables tipo LIFO en la que las nuevas variables se incrementaban al final de la lista, con lo que el último elemento incorporado era el

que quedaba en la parte superior de la pila, por tanto, la variable a la que apuntaba el puntero de la lista.

- 2.- estrategia de inserción de variables FIFO en la que cada nuevo elemento de la lista se incorporaba al final.

Después en algunos métodos se han necesitado funciones para buscar elementos de las listas de variables. Estas funciones son sencillas. Las búsquedas son secuenciales desde el primer elemento hasta el último comparando cada uno de ellos con los parámetros de búsqueda. El desarrollo de algoritmos más eficientes de búsqueda lo dejo para una versión futura del simulador ya que no es objetivo de este proyecto.

A continuación introduciremos brevemente cada una de las variables *struct* que hemos utilizado. Para ilustrar las variables que los forman y que son útiles para llevar a cabo las rutinas del simulador. Es importante destacar que el uso de este tipo de definición de variables a las cuales nos da capacidad el lenguaje C, ha sido determinante para poder crear cada uno de los objetos necesarios en nuestro programa. A partir de ahora denominaremos "objetos" a las variables *struct* definidas por nosotros mismos. Debajo haremos una descripción de cómo son los objetos utilizados.

Objetos base

-L_fich:

```
typedef struct lista_ficheros
{       FILE *fich;
        int origen;
        int destino;
        struct call *HAO;
        struct listaSPF *l_SPF;
        struct lista_ficheros *siguiente;
}tipolista_ficheros;
typedef tipolista_ficheros *lista_fich;
```

Descripción:

- FILE *fich: es un puntero al fichero de texto que contienen la lista de llamadas:
- Int origen: variable entero con el valor del nodo origen de las llamadas
- Int destino: variable destino con el valor del nodo destino de las llamadas
- Struct call *HAO: puntero a un objeto call; dicho objeto es una lista de las llamadas que se van leyendo del fichero de texto y que se irán procesando.
- Struct listaSPF *l_SPF: puntero a un objeto listaSPF; que contiene la lista con los N caminos encontrados por el protocolo de enrutamiento.
- Struct lista_ficheros *siguientes: puntero al siguiente valor de la lista; anidamiento

-Core:

```
typedef struct lista_CORE
{       nt ID;
        int time_actual;
        int origen;
        int destino;
        int param_lam;
        struct enlace *l_enlaces;
        struct lista_CORE *siguiente;
}tipolista_CORE;
typedef tipolista_CORE *l_CORE;
```

Descripción:

- Int ID: variable entera; número de identificación de CORE, es secuencial.
- Int time_actual: variable entera que nos indica la ultima vez que se actualizó la lista de enlaces
- Int origen: variable entero con el valor del nodo origen de las llamadas.
- Int destino: variable destino con el valor del nodo destino de las llamadas
- Int param_lam: variable entera para guardar las características de las *lamdas* de ese enlace.
- Struct enlace *l_enlaces: objeto donde guardamos hay una copia de todos los enlaces de la topología de red. En esta lista vamos guardando un registro de los movimientos que tienes las fibras y las *lamdas* durante el proceso del simulador. Cada Core contiene una copia para ir anotando los cambios de los enlaces según el origen y destino de las llamadas.
- Struct lista_CORE *siguiente: puntero al siguiente objeto de la lista; anidamiento.

-Calls:

```
typedef struct call
{       int hold;
        int arrival;
        int lamda;
        int ID_l_l;
        bool exec;
        struct list_lamda *lamdas
        struct tablaSPF *SPF_choose;
        struct call *siguiente;
}tipocall;
typedef tipocall *ptabla_call;
typedef tipocall *pcall;
```

Descripción:

- Int hold: variable entera que nos indica de punto temporal en que llega una llamada
- Int arrival: variable entera que nos indica el momento temporal en que llega una llamada
- Int *lamda*: variable entera donde se registra la *lamda* utilizada en el *lightpath* por donde se cursa una llamada; se utiliza cuando no usamos sistemas regeneradores
- Bool exec: variable boleana que nos indica cuando una llamada ha sido procesada.
- Struct list_lamdas *lamdas: puntero a una lista de objetos list_lamdas donde se guarda la lista de todas las *lamdas* utilizadas en cada uno de los enlaces del camino.
- Struct tablaSPF *SPF_choose: puntero a un objeto tablaSPF que contiene el camino elegido por el pertinente protocolo, el cual utilizaremos para cursar esta llamada.
- Struct call *siguiente:puntero al siguiente objeto de la lista; anidamiento.

Enlace:

```
typedef struct enlace
{       int * carac;
        int ID_origen;
        int ID_destino;
        int PARAM;
        int * regen;
        struct enlace *siguiente;
}tipoenlace;
typedef tipoenlace *plist_enla;
typedef tipoenlace *penlace;
```

Descripción:

- int *carac: puntero a un array de entero en los cuales cada elemento de array representa una *lamda* del enlace y el valor del elemento es una fibra.
- Int ID_origen: nodo origen del enlace.
- Int ID_destino: nodo destino del enlace.
- Int PARAM: variable entera que utilizamos para guardar algún valor característico del enlace.
- Int *regen: puntero a un entero que nos indica si el nodo origen del enlace es un sistema regenerador
- Struct enlace *siguiente: puntero al siguiente elemento de la lista; anidamiento.

-Resultados:

```
typedef struct resul
{       int origen;
        int destino;
        int cantidad;
        struct resul *sigue;
}tiporesul;
```

Descripción:

- Int origen: variable entera con el origen que provoca la alarma.
- Int destino: variable entera con el destino que provoca la alarma.
- Int cantidad: cantidad de veces que se repite esta misma alarma.
- Struct resol *sigue: puntero a la siguiente alarma.

Objetos inherentes

-SPF:

```
typedef struct tablaSPF
{       int IDnodo;
        struct tablaSPF *siguiente;
}tipotablaSPF;
typedef tipotablaSPF *ptablaSPF;

typedef struct listaSPF
{       int PARAM;
        struct tablaSPF *ptr;
        struct listaSPF *siguiente;
}tipolistaSPF;
typedef tipolistaSPF *plistaSPF;
```

Descripción:

- Int IDnodo: variable entero con el nombre de un nodo.
- Struct tablaSPF *siguiente: puntero al siguiente objeto de la lista de SPF.
- Int PARAM: variable entero habilitada para salvar características sobre los caminos.
- Struct tablaSPF *ptr: puntero a la lista de caminos encontrados por el algoritmo de routing.
- Struct listaSPF *siguiente: puntero al siguiente objeto de la lista.

-Least Loaded:

```
typedef struct least_loaded
{       int min_libre;
        int origen;
        int destino;
        long int IDorden;
        struct tablaSPF *filaSPF;
        struct least_loaded *siguiente;
}tipoleast_loaded;
typedef tipoleast_loaded *ptablal_l;
```

Descripción:

- Int min_libre: variable entero donde apuntamos la cantidad *lamdas* libres en nuestro camino

- Int origen: variable entero con el valor del nodo origen del camino.
- Int destino: variable destino con el valor del nodo destino del camino.
- Int IDorden: variable del orden que ocupa el objeto dentro de la lista.
- Struct tablaSPF *filaSPF: puntero a una copia del camino.
- Struct least_loaded *siguiente: puntero al siguiente objeto en la lista.

La programación modular tiene el cometido de simplificar la lectura y la utilización del código. Se crean dos ficheros uno que es *.h y otro *.cpp. El primero de todos incluye las deficiones de las variables, los structs y las funciones, así como los *#include* necesarios para que funcione el módulo. En el fichero *.cpp escribimos todo el código de las funciones definidas en el fichero *.h.

El incluir las funciones en los ficheros *.h permite que cuando incluimos el módulo en cualquier proyecto se pueden utilizar dichas funciones.

Como hemos visto más arriba en la descripción de los objetos, muchos de ellos, que hemos creado para el simulador, contienen punteros a otros objetos. Estos punteros pueden contener la referencia a otro puntero de dos maneras diferentes una de ellas es:

- **Por copia**: esta opción es la más utilizada ya que nos da total independencia de nuestros módulos. Quiere decir que al hacer la copia creamos nuevas variables con una copia de los valores, Por tanto si en cualquier momento se libera la memoria de los objetos copiados, los nuevos objetos no quedarán con punteros apuntando a posiciones vacías, hecho que puede provocar la inestabilidad del sistema si en algun momento queremos utilizarlos.
- **Por referencia:** así poder identificar cuando han sido borrados porque era importante saber si existían aún o ya habían sido liberadas estas variables.

3.6.4 Variables más destacadas.

A continuación hablaremos de las variables más destacadas que hay en el código del simulador, para aclarar y ayudar a la lectura y comprensión del funcionamiento del código.

Algunas de estas variables son de configuración, otras son variables internas que utilizamos para el control de los procesos del simulador.

·**libre**>> nos devuelve el valor true o false de una función que nos indica si una *lamda* está libre en todos los enlaces de un camino.

·**util_regen**>>esta variable con valor true, nos indica que hay que utilizar la función de selección de camino con regenerador.

·**max_coincide**>>variable de configuración, donde almacenamos el valor que nos indica el número de coincidencias que puede llagar ha haber entre los diferentes caminos que vamos encontrando con el protocolo de routing.

·**selec_SPF**>>variable entero de configuración en el que queda registrado el protocolo de selección de camino que utilizaremos: 1=aleatorio; 2=balanceado; 3=least loaded balanceado; 4=least loaded corto.

·**time_actualiza**>>variable de configuración que utilizaremos para establecer el tiempo de actualización. Podemos indicar al programa cada cuanto tiempo estimamos que tardan las tablas en actualizarse. Un valor elevado de esta variable podía provocar grandes bloqueos por no tener la red convergida correctamente y un valor bajo provoca gran ocupación de ancho de banda de la red con señalización, bajando su rendimiento.

·**tmp_F_libres**>>vector de números enteros donde se almacenan las *lamdas* por orden de ocupación. En las posiciones primeras tenemos el número de la *lamda* menos ocupada. Por tanto si empezamos a leer el vector recorriéndolo desde el principio empezaremos a leer primero las *lamdas* más libres.

·**caminos**>>variable de configuración que nos indica el número de caminos que nuestro protocolo de "routing" deberá buscar, teniendo en cuenta una variable que vimos antes, max_coincide.

·**lamdas**>>variable de configuración que establece en número de *lamdas* que aplicamos a las fibras de nuestra topología de red.

·**fibras**>>igual que la anterior, pero en este caso aquí esta variable establece en número de fibras que contienen los enlaces de nuestra red.

·**tmpl_enlaces**>>puntero a lista de enlaces que se genera al leer el fichero de texto donde está la topología de red a simular.

·**l_enla_actual**>>punteros a lista de enlaces en la que tenemos almacenada la visión real en todo momento del estado de la red, es la lista de referencia. De esta variable cogeremos las actualizaciones. La idea de nuestro simulador es que el tiempo de actualización (time_actualiza) es el tiempo que tarda toda nuestra red en converger.

·**enla_busc_lamda, enla_busc_lamda2**: puntero a un enlace donde anotamos los cambios las *lamdas*, tanto si se ocupa restando su vamos y si se libera incrementando su valor. El primer puntero apunta al enlace de la lista de enlaces de referencia, el segundo puntero apunta al enlace de Core que está siendo evaluado en este momento.

·**l_fich**>>puntero al l_fich que se está procesando actualmente.

·**l_fich_backup**>>puntero al principio de la lista de l_fich, para poder reiniciar el bucle de l_fich's, ya que nuestro simulador hace un ciclo con cada uno de los l_fich.

·**tmpSPF**>>puntero donde se creará la lista de caminos, dicha lista será después copiada al l_fich que se está procesando en ese momento.;

·**pbalanced**>>puntero del camino seleccionada mediante el método de balanceado

·**CORE**>>puntero a la lista de Core's que se van creando cada vez que el usuario introduce un origen y destino.

·**l_l**>>puntero a una lista de objetos least_loaded.

·**tmp_call**>>variable de tipo puntero utilizado para la creación de las calls que se van leyendo del fichero de texto, para el proceso más delante de sus valores.

·**l_lamdas**>>puntero a la lista de *lamdas* que se genera cuando se buscan *lamdas* para estableces un *lightpath*. Si se consigue encontrar una *lamda* libre en todo un camino, la guardaremos en el call que se está procesando.

·**error_res**>>variable que apunta a la lista e objetos donde guardamos los resultados de la simulación.

3.6.5 Diseño del programa e implementación

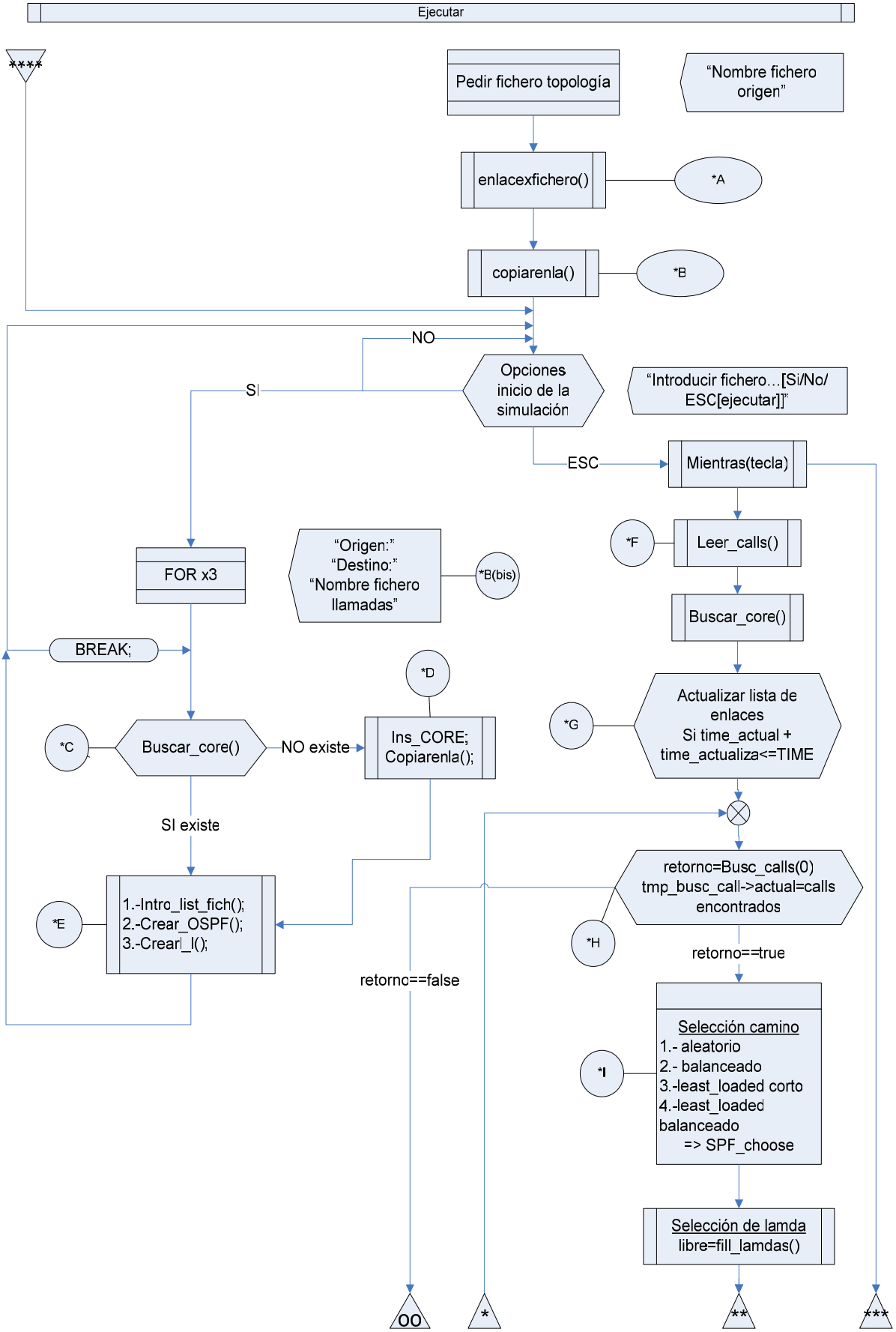
He descrito con anterioridad las partes del simulador, hemos explicado conceptos importantes y los puntos centrales para poder simular una red semitransparente.

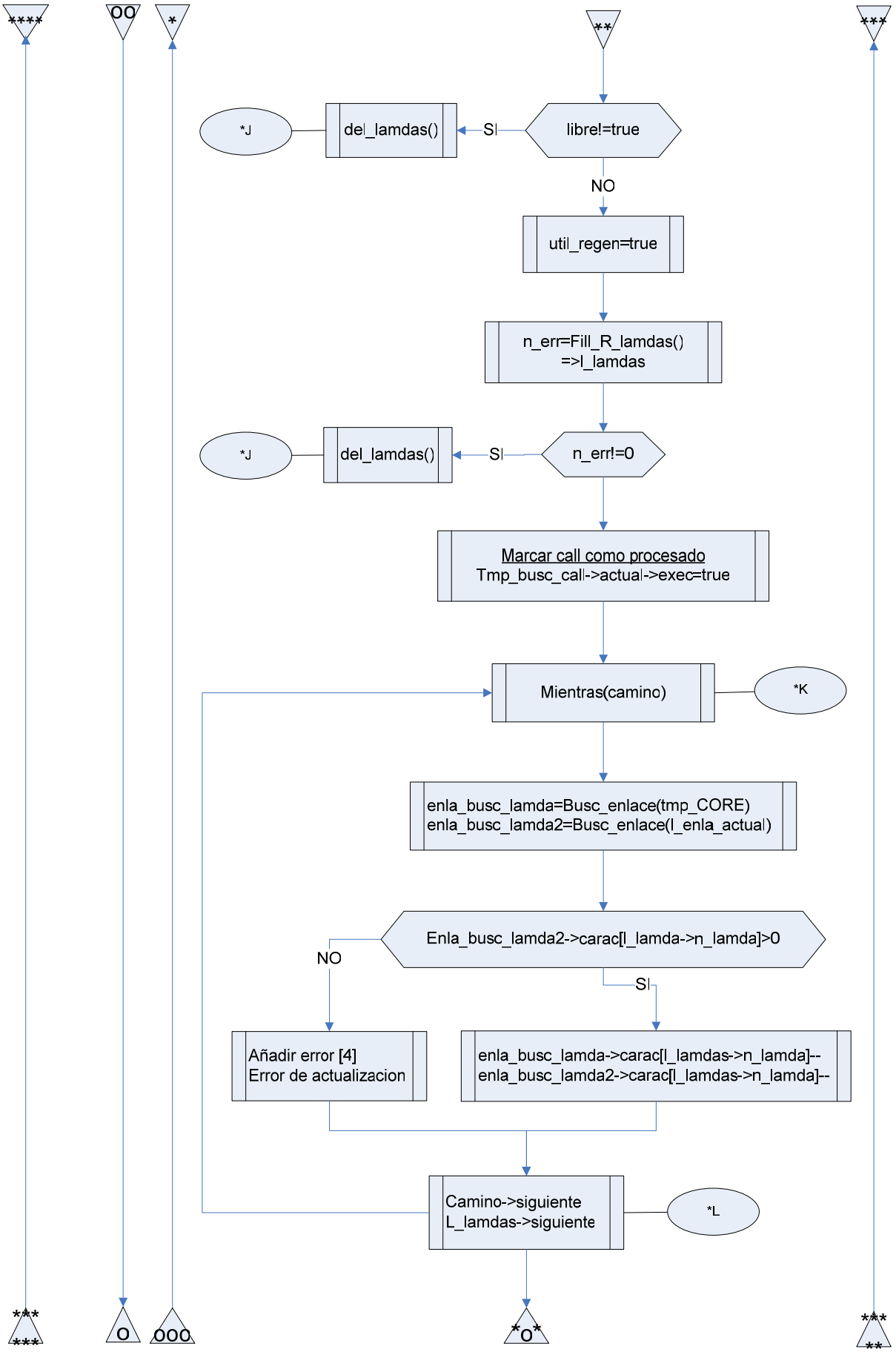
He descrito el concepto de llamadas, el de los enlaces y nodos. El núcleo del simulador se basa en las variables de "Core" y "L_fich". Descubrimos el concepto de "TIME". Y después hemos descrito los conceptos físicos y dispositivos que hemos modelado para el simulador.

Después he introducido la descripción de las variables principales del entorno de programación, es importante saber como son para después descubrir su funcionamiento.

Ahora a continuación trataremos de explicar como trabaja todo este junto.

Módulo principal (main()):




****
OO
*
**
***
*J
del_lamdas()
SI
libre!=true
NO
util_regen=true
n_err=Fill_R_lamdas()
=>l_lamdas
*J
del_lamdas()
SI
n_err!=0
Marcar call como procesado
Tmp_busc_call->actual->exec=true
Mientras(camino)
*K
enla_busc_lamda=Busc_enlace(tmp_CORE)
enla_busc_lamda2=Busc_enlace(l_enla_actual)
Enla_busc_lamda2->carac[l_lamda->n_lamda]>0
NO
SI
Añadir error [4]
Error de actualizacion
enla_busc_lamda->carac[l_lamdas->n_lamda]--
enla_busc_lamda2->carac[l_lamdas->n_lamda]--
Camino->siguiente
L_lamdas->siguiente
*L
***
***
O
OOO
O
***
**

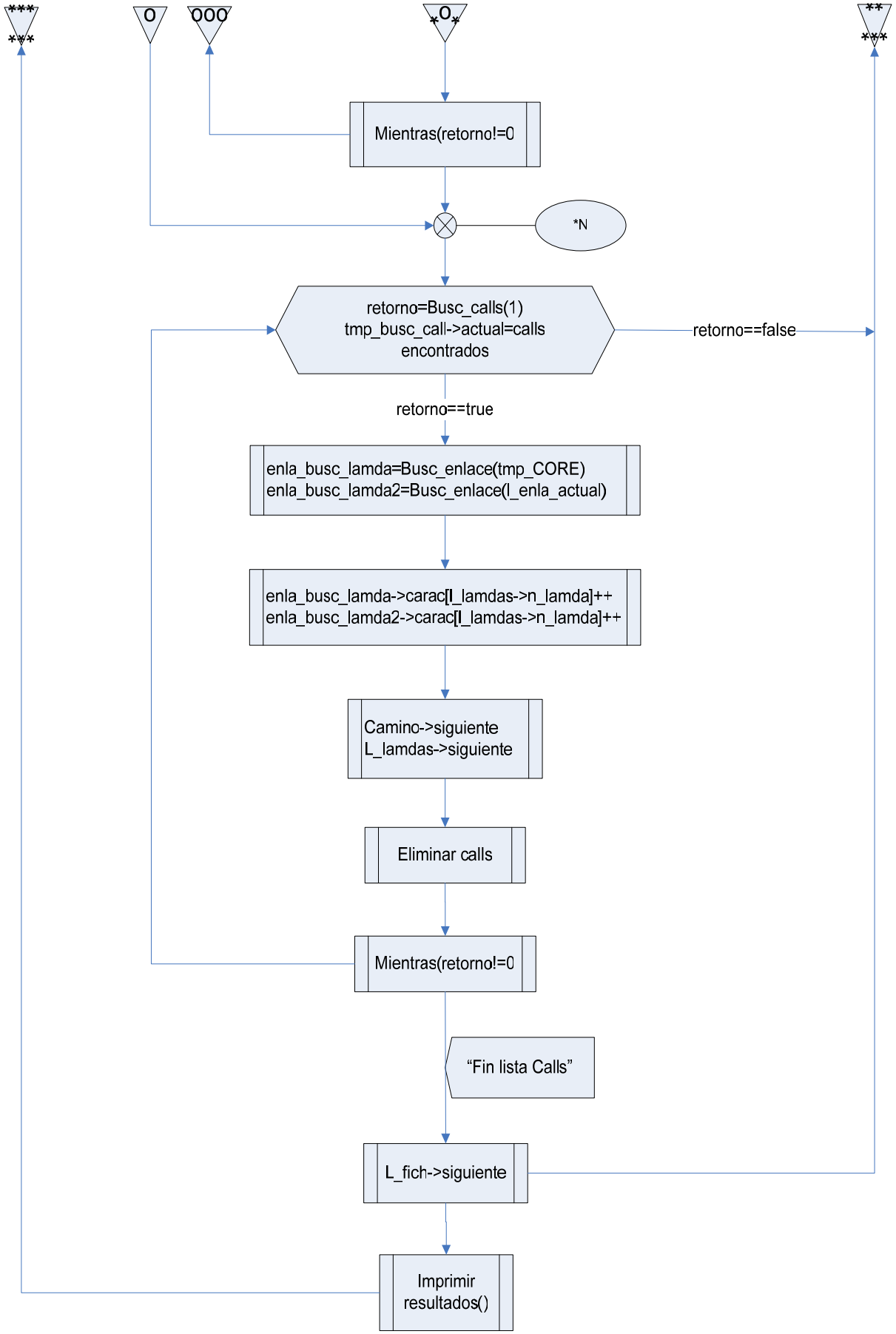


Fig. 3.4 Diagrama de flujo del main()

*A:

Se leen el fichero de topología y se generan la lista con los objetos nodos con cada nodo leído y la lista de enlaces con cada objeto enlace, en el momento que la función lee en el fichero con formato que un nodo esta conectado a otro.

*B:

Copiamos la lista de enlaces (tmpl_enlaces) creada a partir de la lectura del fichero de topología a otra en la que guardaremos los cambios que se produzcan durante la simulación en tiempo real (l_enla_actual).

*C:

Comprobamos a través de una búsqueda si existe el un CORE con Origen y Destino iguales a las variables Origen Destino que se acaban de introducir

*D:

Se crea un nuevo objeto CORE; se copia en el las variables del objeto y una copia de la lista de enlaces como se creo de origen copiarenla (tmpl_enlaces)

*E:

1.-Se crea un nuevo objeto l_fich (intro_list_fich()) dentro de la lista de l_fich.

2.-se crea un camino a través del protocolo creado para el fin (crearOSPF()) y se copia por valor a l_fich->l_SPF; se rellenan las variables de l_fich con los valores de origen y destino introducidos en *B(bis).

3.-Creamos los objetos necesarios para la selección de camino en este caso: l_l (crearl_l) con los valores de origen destino en la lista l_l por si la selección de camino de hace a través least_loaded.

*F:

Se empieza a realizar la lectura de líneas del fichero de texto de llamadas apuntado por el l_fich actual. Se recuperan los valores de HOLD y ARRIVAL time. Se genera un nuevo objeto call en la lista de calls del objeto l_fich actual.

*G:

Después de la búsqueda del CORE con el mismo Origen y Destino que l_fich, se utiliza la variable del Core correspondiente al tiempo de actualización: time_actual para saber si dicho l_fich ha superado el tiempo de actualización con lo cual debería actualizarse ya.

*H:

En este momento se empiezan a inspeccionar los calls leídos del fichero de texto de llamadas. Se leen todo los objetos de la lista l_fich apuntado en el momento. Cuando se finaliza la lectura de toda lista se retorno = y se sale del bucle para seguir con el siguiente objeto l_fich de la lista. Las llamadas cursadas son las que time arrival= o > TIME.

*I:

Es este momento se elegirá el camino (*lightpath*) por el que se cursará la llamada, dependiendo de la elección del protocolo de seleccionado. El resultado del camino seleccionado queda guardado en SPF_choose y este puntero queda registrado en el call->SPF_choose que se cursa en este momento.

*J:

Si el resultado de intentar encontrar una *lamda* libre en todo el camino para ocuparla es negativo el objeto, donde guardamos la lista de *lamdas*, donde se debieran guardar las *lamdas* encontradas, lo eliminamos.

*K:

Empezamos un bucle para ocupar las *lamdas* encontradas. Se ocuparan las *lamdas* en la lista de enlaces que contiene el CORE (CORE->l_enlaces) y l_enla_actual.

*L:

Anotamos las modificaciones en cada enlace; recorremos todos los enlaces que componen el camino seleccionado y modificamos la *lamda* encontrada.

*N:

Iniciamos el bucle para liberar las llamadas que han llegado a su fin o sea Arrival+Hold =TIME

Como podemos observar en el diagrama de flujo y ya hemos comentado con anterioridad el modo en como se van leyendo los L_fich es cíclico.

Estamos trabajando con un entorno de programación monotarea. Se programan instrucciones que se leen y se ejecutan secuencialmente. En el diseño del programa analice que para poder hacer una simulación correcta y poder gestionar el funcionamiento de los procesos del simulador, como si de una red real se tratara, tenia que hacerlo por eventos.

Tal función no es posible en el entorno de trabajo monotarea que utilizo, por tanto tuve que diseñar un método de ejecución del flujo del programa que permitiera gestionar cualquier evento de los nodos en cualquier momento.

La manera de hacerlo fue llevarlo a cabo mediante un ciclo de trabajo en que se inspecciona cada L_fich. Este método fue inspirado en el *Round-Robin* como ejecutan muchos sistemas operativos multitarea, en mi caso sin *quantum*, el sistema se detiene en cada proceso el tiempo necesario para que acabe.

A partir del punto *F se empieza el bucle que no acaba mientras alguno de los L_fich insertados contenga llamadas para procesar. Los eventos a escuchar por una red real son las llamadas que realizan los usuarios, que en nuestro simulador son representadas en las listas de llamadas en los ficheros de texto.

En cada vuelta del bucle (*F), dentro de cada L_fich se hará una lectura del fichero de texto de llamadas, después se comprobarán si, de las llamadas leídas con anterioridad, hay alguna de cumpla con la condición Arrival Time igual a TIME con lo cual habrá que procesar esa llamada.

Así sucesivamente hasta que se cabe el fichero de texto. El proceso se repite con todos los L_fich hasta que todos hayan acabado con el fichero de texto. Cuando sea así la simulación está finalizada y presentaremos los resultados.

Módulo de enrutamiento:

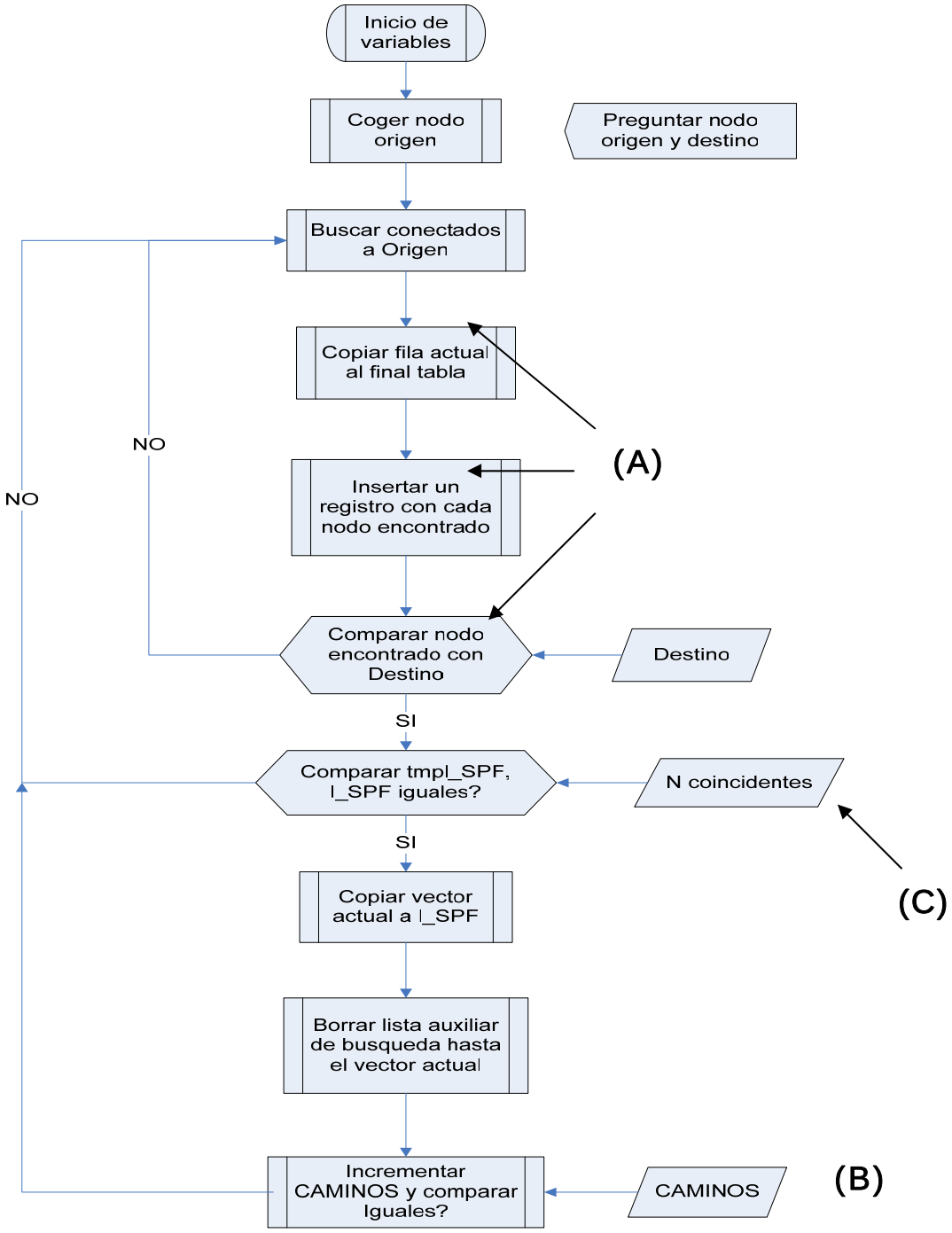


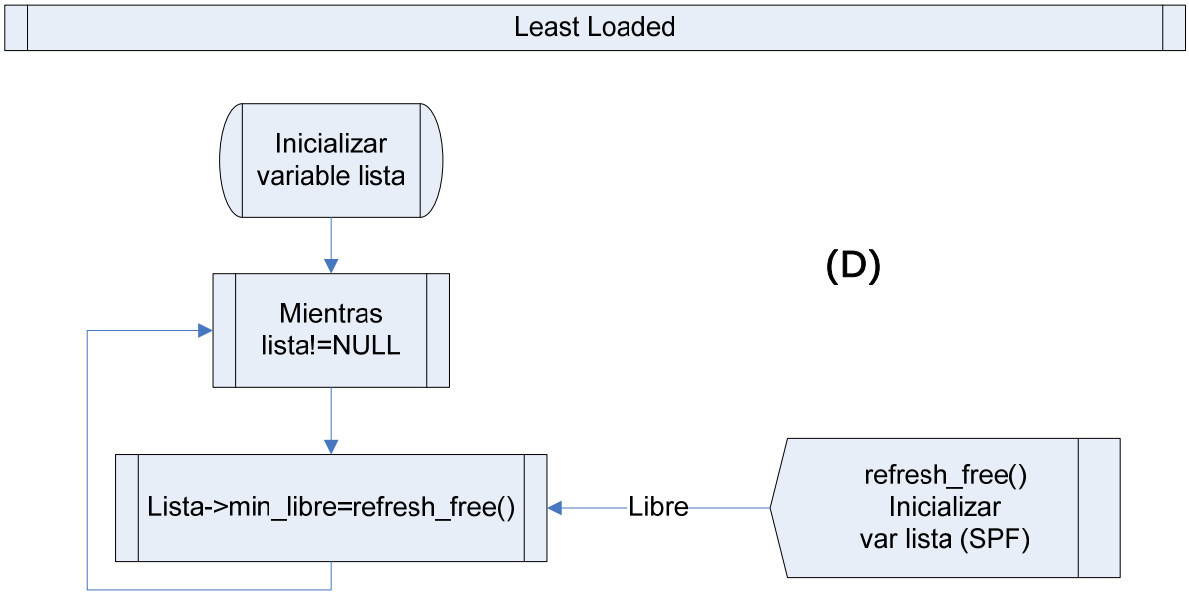
Fig. 3.5 Diagrama de flujo del protocolo SPF

Hubo varias consideraciones a la hora de crear la función de *routing*. Como podemos ver en el diagrama, en el punto **(A)** se realiza el árbol de conexiones, hasta que otras ramificaciones lleguen al punto destino. En el punto **(B)** comparamos con la variable CAMINO hasta conseguir los caminos deseados.

Después de probar el protocolo con varias topologías y diferentes puntos de origen y destino, descubrí que tenía un inconveniente y es que en topologías poco malladas se repetían excesivamente los nodos que intervenían en los diferentes caminos, lo que podía provocar que se cargaran excesivamente algunos nodos y se hicieran impracticables los caminos descubiertos. Fue por ello que añadimos la posibilidad de poder decirle al protocolo la cantidad de nodos repetidos que podía haber entre todos los caminos encontrados. Esta marcado con **(C**) el punto donde se miran las coincidencias entre caminos.

El resultado fue muy satisfactorio. Pero cabe la posibilidad que si exigimos mucho a los parámetros que deben cumplir los caminos, para ser seleccionados, no encontremos ninguno que cumpla las exigencias. Por consiguiente se dejó la posibilidad, que manualmente, se pueda cortar el proceso del protocolo para no entrar dentro de un bucle infinito. Para un futuro se puede programar un temporizador programable que corte el proceso cuando lleve un tiempo trabajando.

Módulos de elección de camino "Least Loaded":



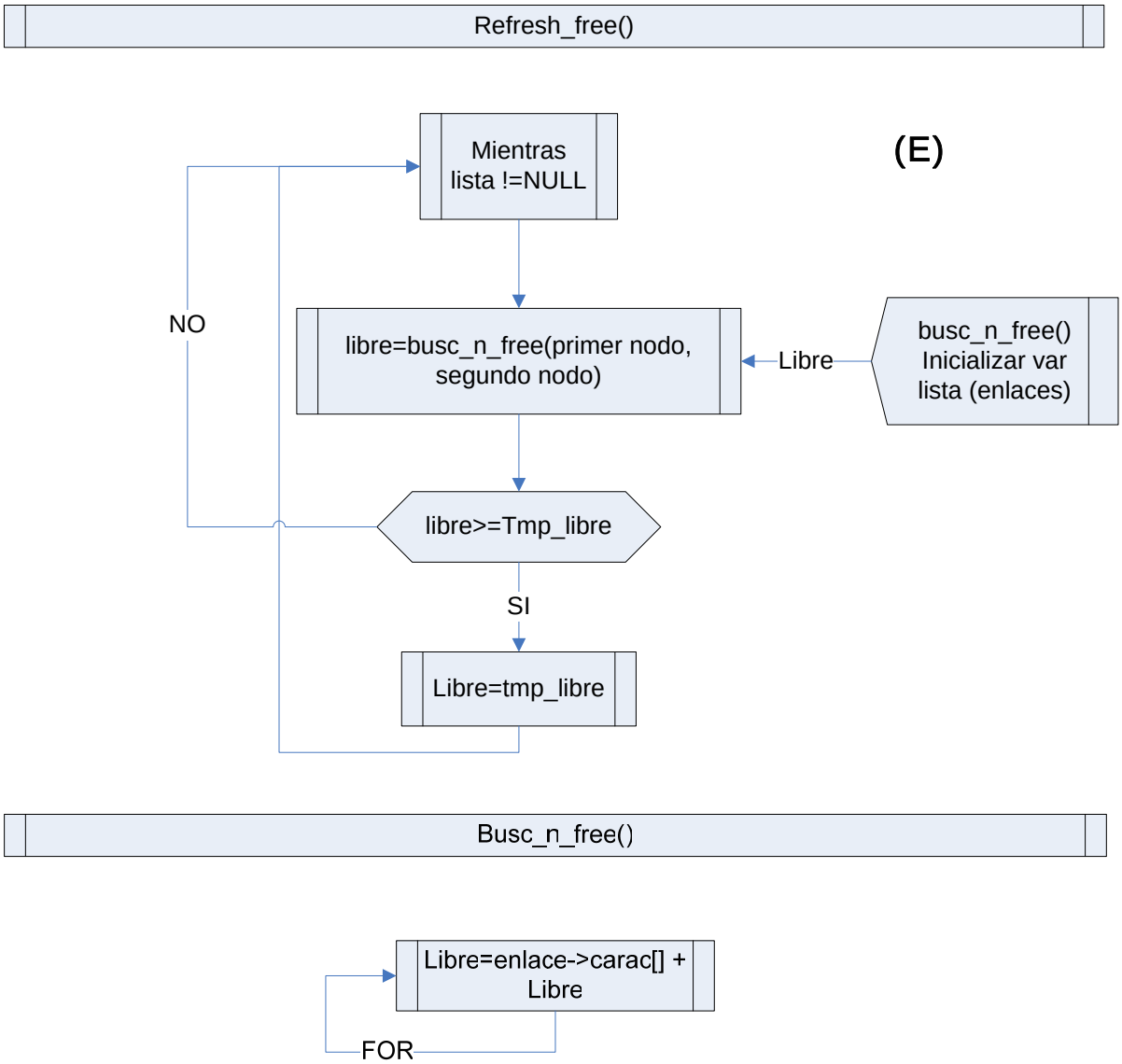
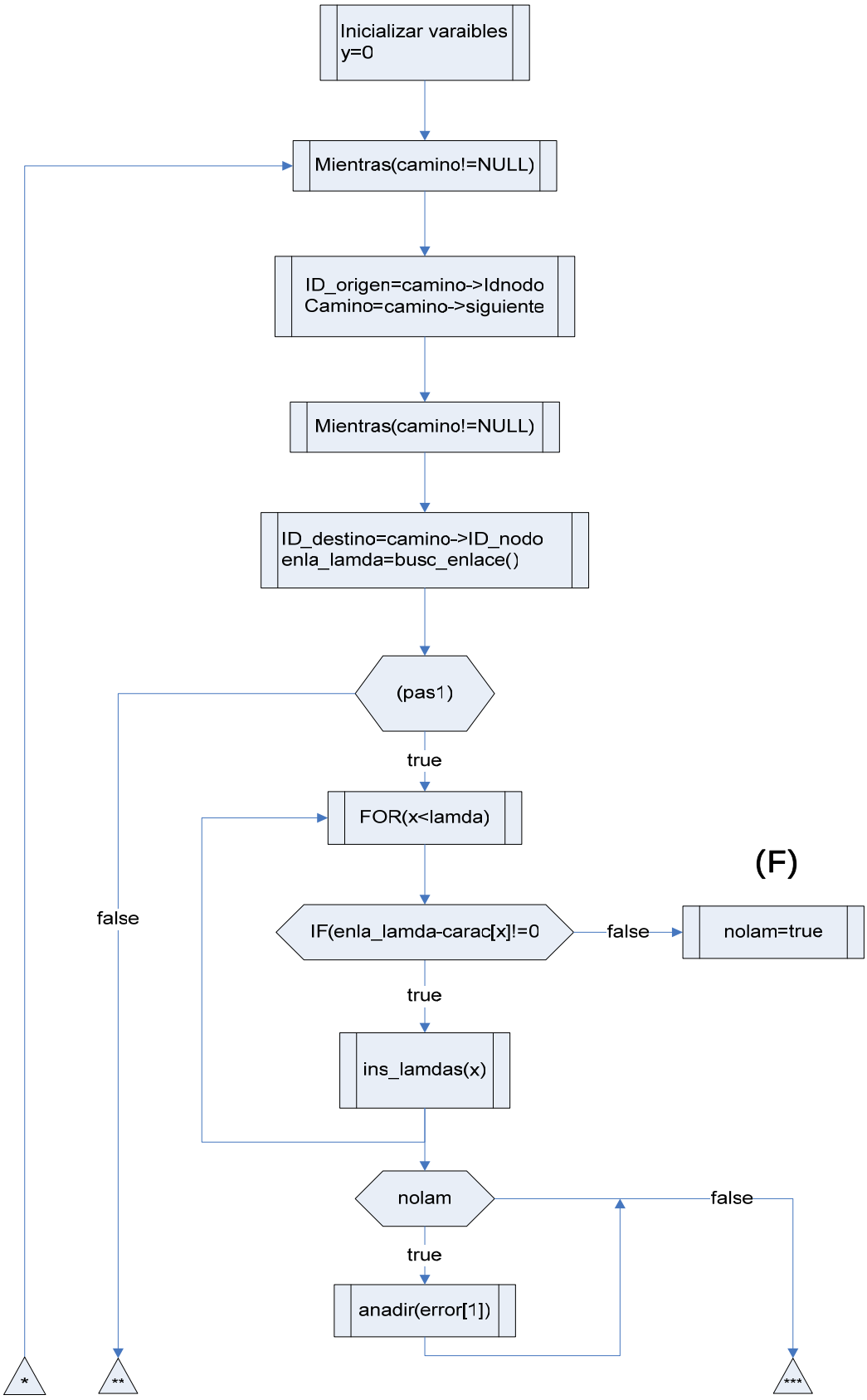


Fig. 3.6 Diagramas de las funciones con que obtener el camino más descargado

Como podemos ver en la figura 3.5 el algoritmo Least Loaded se compone de dos funciones, la primera de ellas es ***crearl_l()***, marcada con el punto (D), la cual crea una lista con los caminos "least loaded" y otra función llamada ***refresh_free(),*** marcada con el punto (E) la cual actualiza la lista de "least loaded" al nuevo estado de la red.

Otras funciones importantes:

-Fill_R_lamdas():



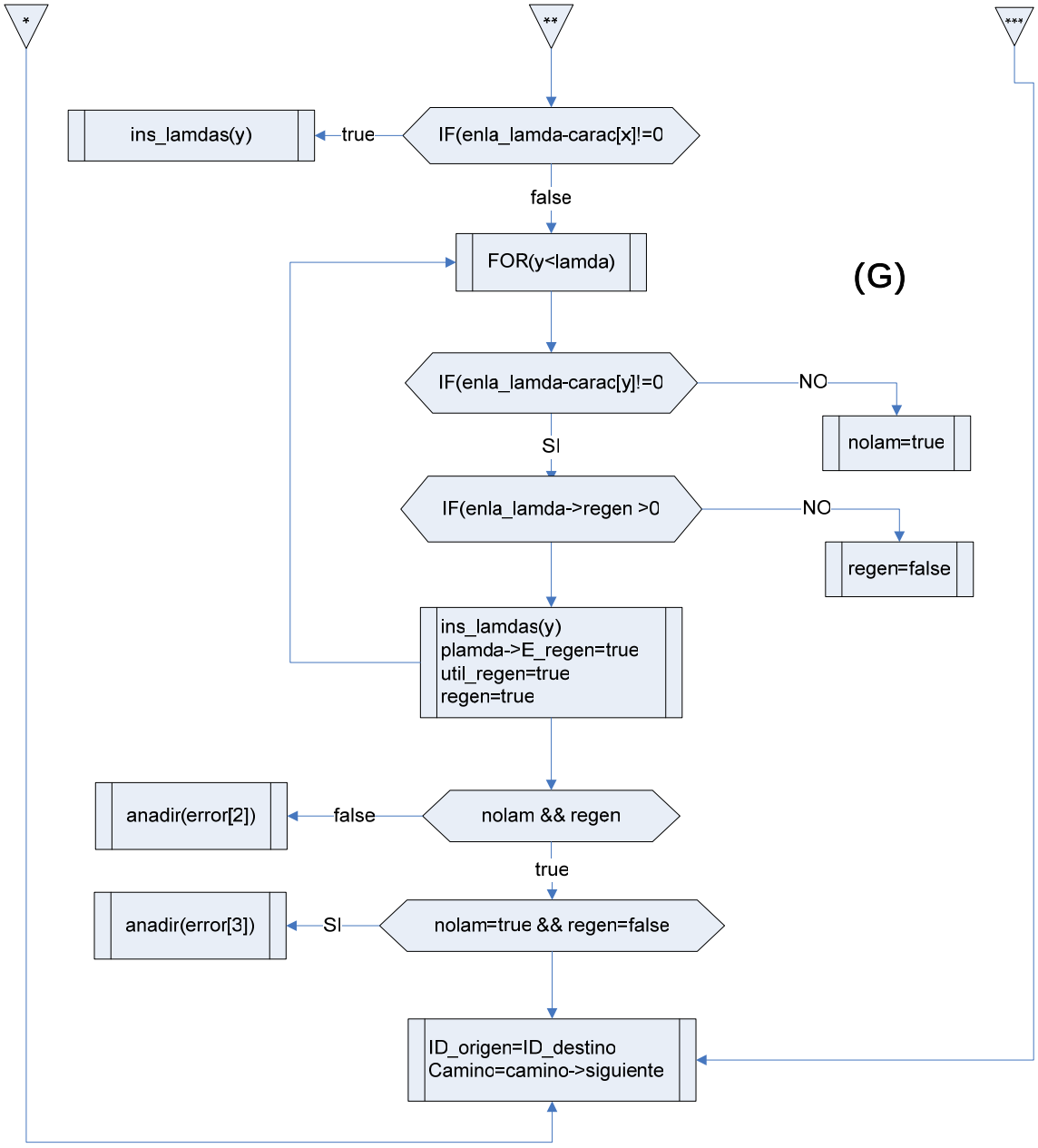


Fig. 3.7 Diagrama de flujo de la función que genera los *lightpath* con regeneradores

Esta función es la diseñada para buscar los caminos teniendo en cuenta los regeneradores de los nodos. En el punto F, empezamos a probar la primera *lamda* más desocupada, en el caso que no sea posible, en alguno de los nodos ocupar dicha *lamda*, pasaremos al punto G, donde se testean los regeneradores libres, si existe alguno, se utiliza quedando ocupado.

3.7 Obtención de resultados

3.7.1 Puntos previos.

- Para las simulaciones utilizaremos dos topologías de red reales, la red española RedIris, que interconecta muchos centros de investigación y universitarios, y la red americana NSF.
- Otro punto, las listas de llamadas contienen el valor "Interarrival Time" y "Hold Time" ambos valores están calculados con una función Poisson.
- Las listas se componen de 5.000 registros cada una, cada registro representa una llamada introducida a la red y contiene su valor "Interarrival Time" y "Hold Time".
- El protocolo de enrutamiento SPF calcula el mismo camino para nodos inversos, si introducimos un nodo origen A y otro destino B y luego un nodo origen B y destino A, el protocolo nos devolverá los mimos nodos intermedios pero a la inversa. Por tanto todos los resultados son aplicables a los caminos inversos.
- Para más detalle, adjunto a la memoria viene un fichero de Excel donde se hallan todos los datos devueltos por el simulador, que corresponden a las gráficas presentadas aquí.

3.7.2 Comparación de topologías

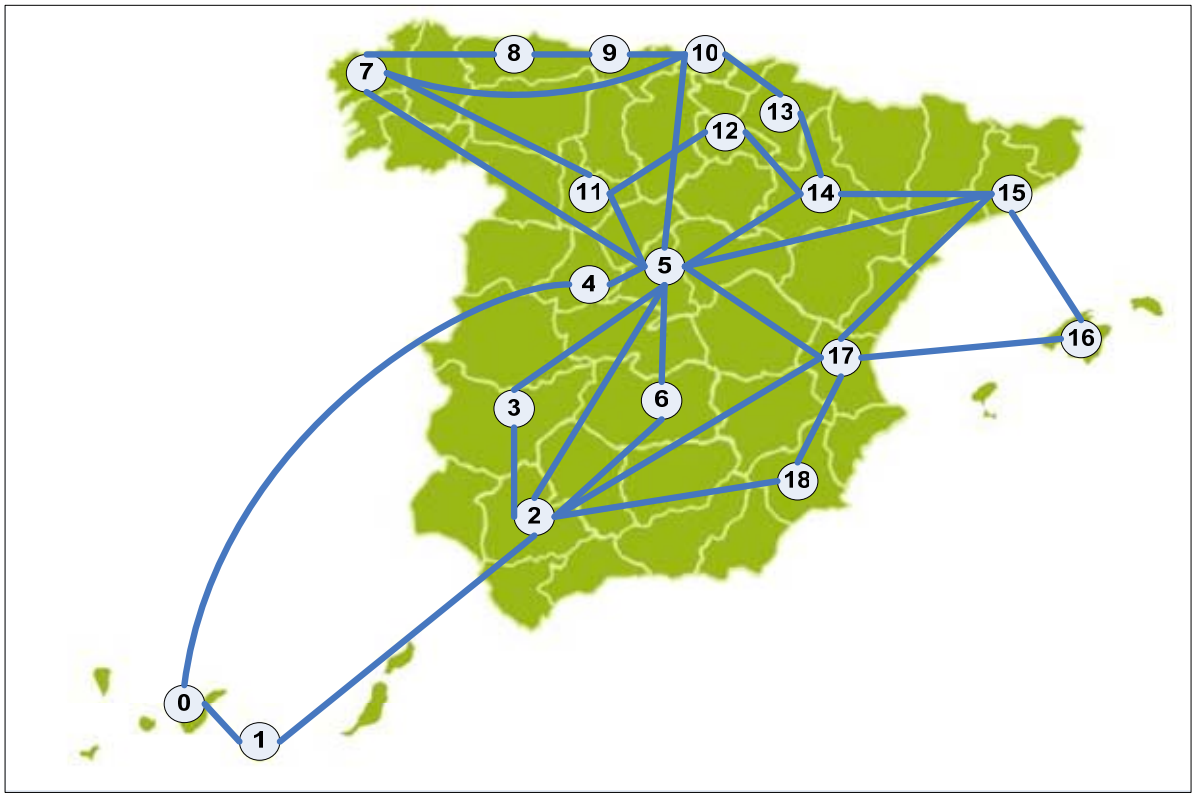


Fig. 3.8: Topología de RedIris en España

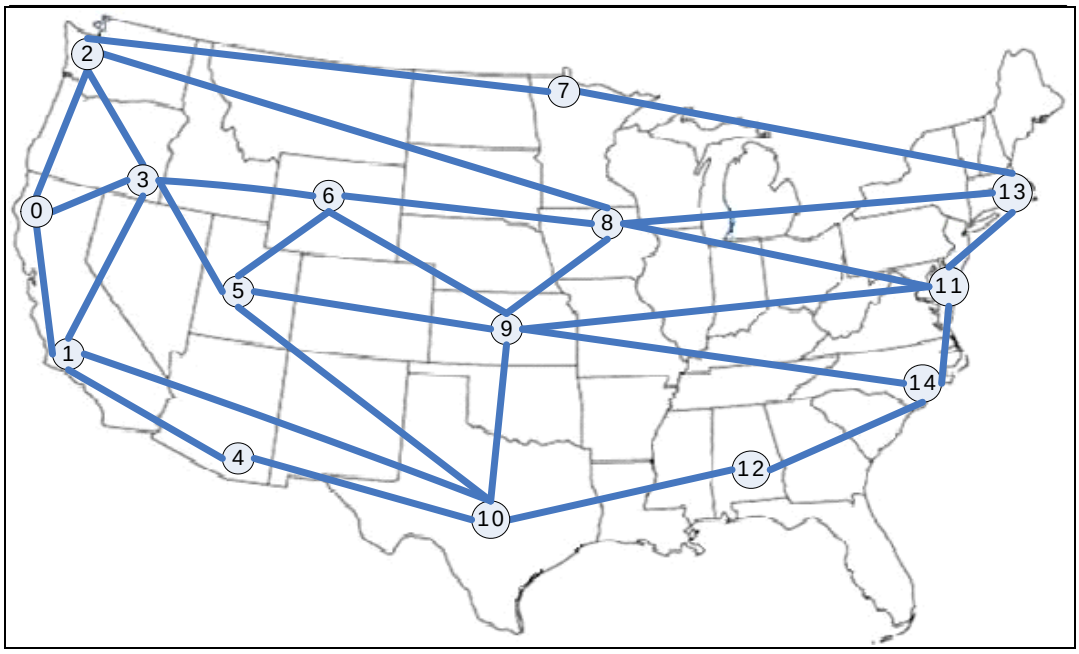


Fig. 3.9: Topología de red NSF en EEUU

Condiciones iniciales:

- Número de nodos coincidentes= 2
- Protocolo= SPF
- Número de caminos a buscar= 4

NSF (USA)

Origen y Destino	Caminos encontrados
Norte-Sur	0 - 2 – 8 – 13
0-13	0 – 2 – 7 – 13
	0 –3 – 6 – 8- 13
	0 – 3 – 2 – 8 – 13
Este-Oeste	7 – 13 – 11 – 9 – 10
7-10	7 – 13 - 8 – 9 – 10
	7 – 2 – 8 – 9 – 10
	7 – 2 – 3 – 5 – 10

Tabla 3.1

RedIris (España)

Origen y Destino	Caminos encontrados
Norte-Sur	1 – 2 –5 – 10
1-10	1 – 2 – 17 – 5 - 10
	1 – 2 – 6 – 5 – 10
	1 – 2 – 5 – 7 - 10
Este-Oeste	18 – 17 – 5 – 7 – 8
18- 8	18 – 2 – 5 – 7 – 8
	18 – 17 – 5 – 10 – 9 – 8
	18 – 17 – 15 – 5 – 11 – 7 - 8

Tabla 3.2

Comentarios:

En la simulación superior se compara el comportamiento de dos redes: una mallada parcialmente con un nodo central muy lleno de enlaces (RedIris), con otra red mallada completamente.

Los resultados son bastante descriptivos de que sucede con las redes malladas. Como comentábamos con anterioridad es la topología de red que se impone.

Como se puede observar en la **tabla 3.1** en la red NSF cuando trazamos caminos desde el Este al Oeste o de Norte a Sur, la cantidad de nodos implicados en el camino se asemeja. Eso quiere decir que el comportamiento de la red en general sea el nodo que sea, será parecido, no encontraremos zonas de la red muy congestionadas y otros escasamente utilizados.

Por otro lado se puede observar que la variedad de nodos que se utilizan en una red mallada es mucho mayor que en una red semimallada con nodo central. Esta característica aportará mayor solvencia para evitar bloqueos y más opciones en caso de fallas de enlaces o nodos.

En la red RedIris, **tabla 3.2**, se observa como el nodo 5, el central, aparece en todos los caminos. Resultado este nodo estará sumamente congestionado ya que todas las llamadas que se cursen, en este caso, pasarán por el nodo 5. Es por ello que tendremos que empezar a pensar en proveer a este nodo de mucha más capacidad (más fibras y/o *lamdas*), comparado con los otros, sin aún haber hecho simulaciones con tráfico.

Siguiendo lo anterior, pensemos que pasaría si el nodo 5 cayera, siendo RedIris una red española, los usuarios del Sur de España no podrían comunicarse con el Norte, así mismo los del Este con los del Oeste.

3.7.3 Comparar: protocolos elección de camino sin regeneradores.

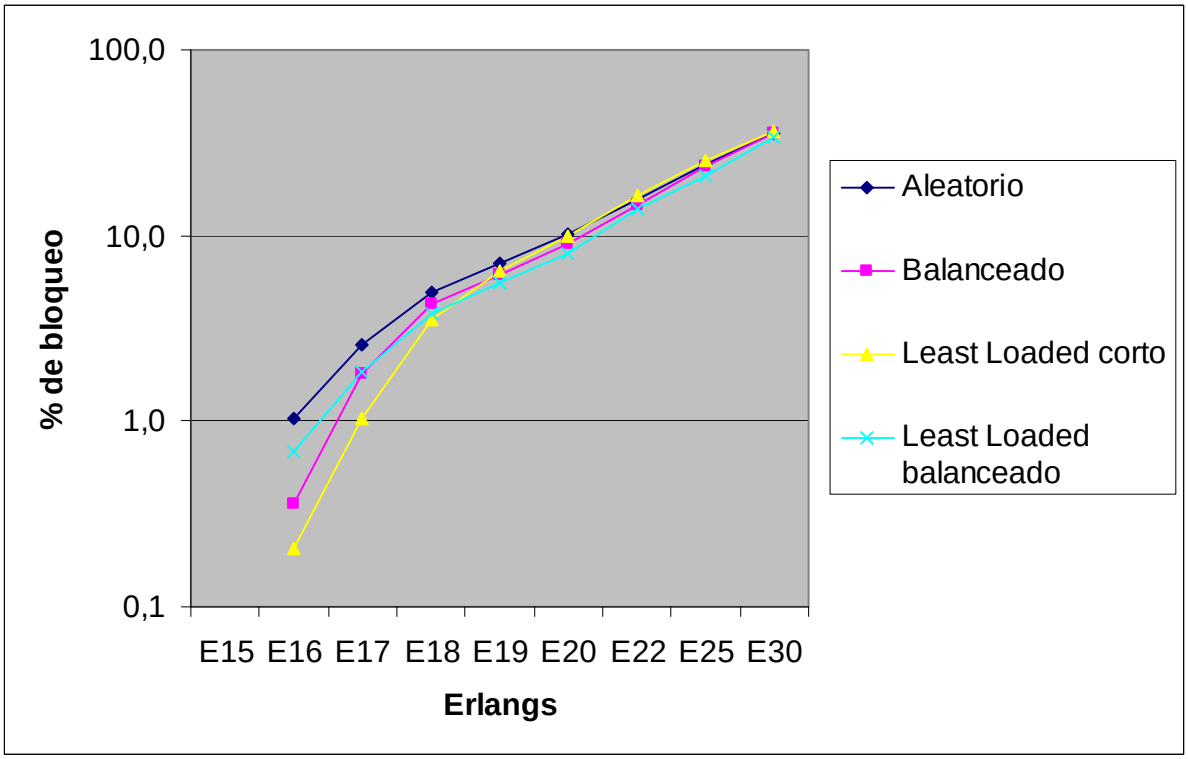
Condiciones iniciales:

- Número de nodos coincidentes=2
- 5000 llamadas por lista origen-destino
- Número de caminos=4
- Número de lamdas=20
- Número de fibras=2
- Sin regeneradores

Lista de llamadas:

Nodo origen	Nodo destino
0	14
0	13
1	8
0	12

Tabla 3.3

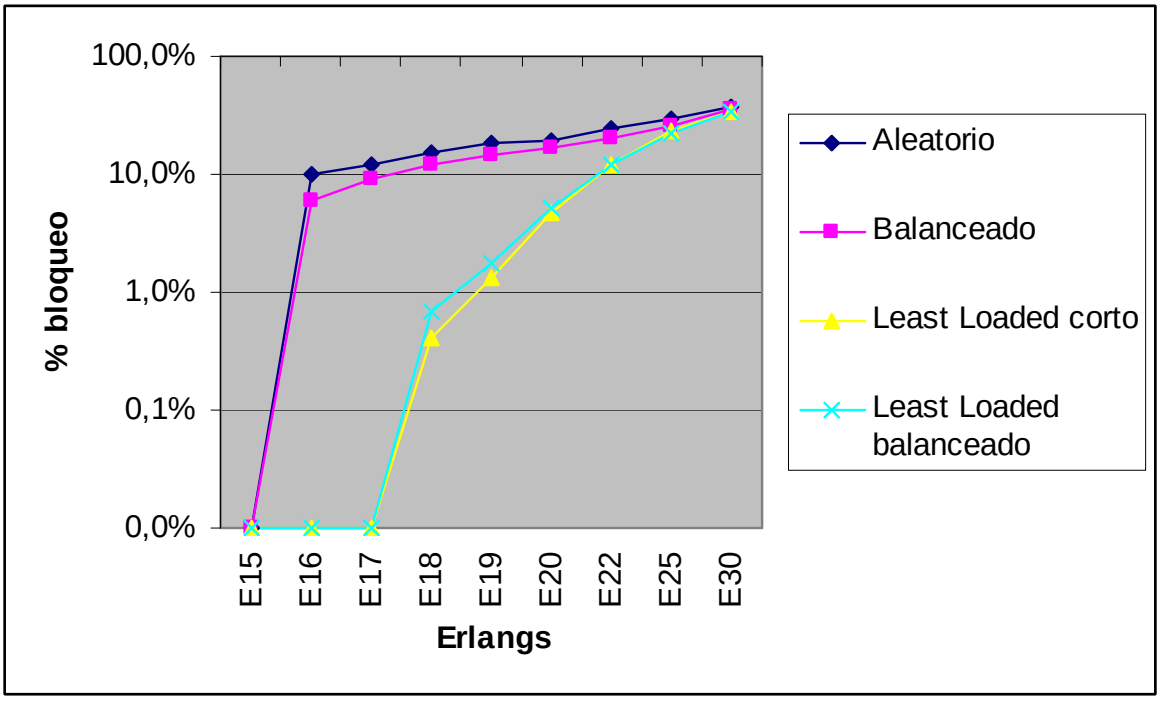


Gráfica 3.1

Lista de llamadas:

Nodo origen	Nodo destino
17	0
11	1
9	1
13	0

Tabla 3.4



Gráfica 3.2

Comentarios:

En este apartado hemos querido comparar los protocolos de elección de camino que tenemos disponibles en nuestra aplicación actualmente.

El proceso de elección de uno de los 4 caminos disponibles, caso anterior, se realiza cada vez que se cursa una llamada. Eso quiere decir que el protocolo con la información del estado de la red, en cada llamada decide cual es el camino mejor para ocupar.

En la **gráfica 3.1**, se han elegido unos nodos origen destino con la peculiaridad que los nodos origen es casi siempre el mismo **(Tabla 3.3)** y los nodos destino están muy próximos, con la finalidad de evaluar que grado de bloqueo obtendremos cuando la red quiera establecer comunicaciones entre nodos vecinos.

Los resultados de la gráfica nos muestra que cualquiera de los protocolos de elección de caminos, a baja carga, el que mejor funciona es el “Least Loaded corto”, que elige el camino más descargado y más corto.

Aunque no hay extremas diferencias entre los cuatro, a demás a medida que la red se va cargando vemos como el comportamiento se equipará, lo que nos hace concluir que llega un momento en que la red se queda sin recursos que ofrecer, empieza a bloquearse sin remedio, sea cual sea la estrategia que realicemos para buscar un camino libre.

Aunque se puede observar que el protocolo “Least Loaded balanceado” salvo en el principio siempre corre por debajo de los demás.

La **gráfica 3.2**, muestra información sobre unos nodos que tienen la peculiaridad, en el caso de los nodos origen, de estar dispersos por los extremos de la red.

En un principio se puede pensar que teniendo destinos más dispersos, los nodos que intervendrán en los caminos pueden ser más variados y devolver menos bloqueo.

El resultado es bastante concluyente, no nos equivocamos en un principio, ya que para cargas bajas vemos que los protocolos que buscan el camino más descargado resultan mucho más eficientes.

Sin embargo a medida que se carga la red la eficiencia de estos dos protocolos baja, otra vez atribuible a la falta de *lamdas* que puedan ser ocupadas. Es concluyente otra vez que si la red no dispone de recursos, será imposible dar servicio a los usuarios.

Para acabar se vuelve apreciar en la gráfica, como el protocolo “Least Loaded balanced” resulta más efectivo a cargas elevadas de tráfico.

3.7.4 Funcionamiento de la red con regeneradores

Una vez hemos visto como se comportan los protocolos de enrutamiento y elección de camino, vamos a incorporar en nuestras simulaciones la capacidad de conmutación de los nodos.

Como hemos hablado en apartados anteriores, la capacidad de conmutación de los nodos da más eficiencia a la red. A continuación veremos si eso es cierto o no.

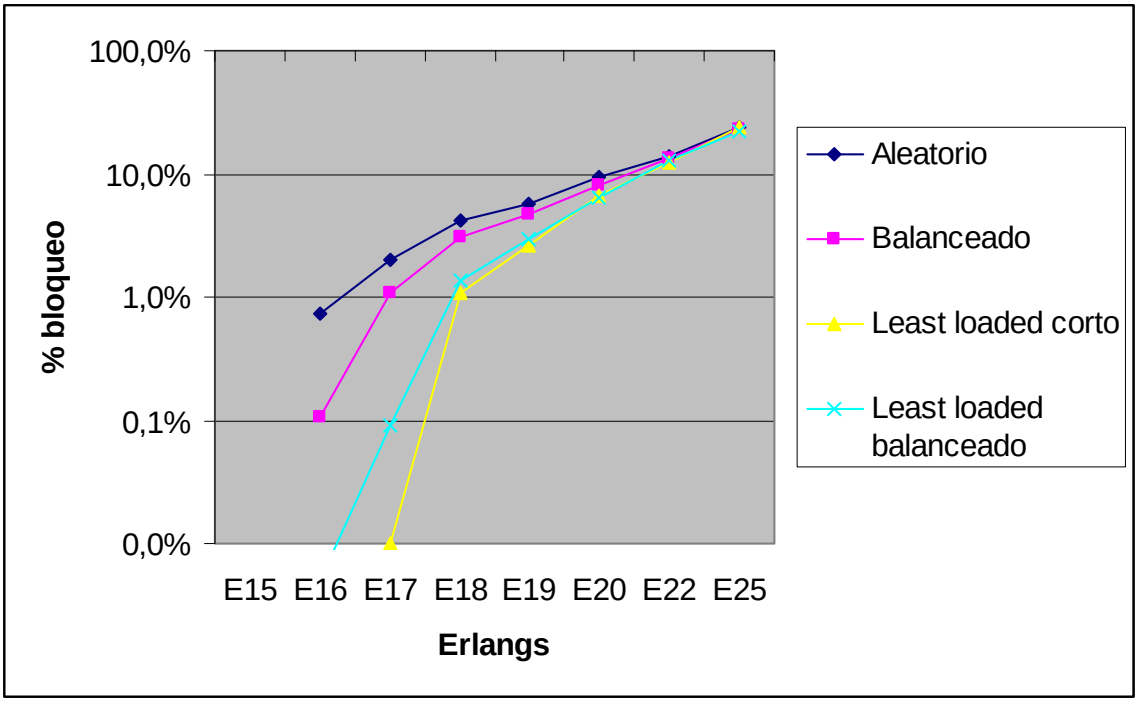
Condiciones iniciales:

- Número de nodos coincidentes=2
- 5000 llamadas por lista origen-destino
- Número de caminos=4
- Número de lamdas=20
- Número de fibras=2
- Con regeneradores:5

Lista de llamadas:

Nodo origen	Nodo destino
0	14
0	13
1	8
0	12

Tabla 3.5



Gráfica 3.3

Comentarios:

Los resultados de la **gráfica 3.3** creo que son bastante aclaradores.

Hemos elegido los mismos nodos que nos dieron los resultados de la **gráfica 3.2** para comparar.

Podemos observar como en un inicio con baja carga, los cuatro protocolos se comportan de una manera mucho más eficiente, en el caso de "Least Loaded corto", sobre el punto de los 17 Erlangs de carga, el protocolo no da ningún bloqueo.

Sobre 18 y 19 Erlangs los resultados son significativamente mejores. Aunque al llegar a los 20 Erlangs las cosas se igualan.

Cabe recordar que de las dos simulaciones probadas con anterioridad, ésta es el peor caso.

Si seguimos inspeccionando la gráfica vemos que los valores para más carga de llamadas a partir del punto E20 la gráfica crece con la misma pendiente que en la simulación sin regeneradores (**gráfica 3.1**). También se observa como los cuatro protocolos crecen de la

misma manera, por consiguiente pienso que otra vez no encontramos con el fenómeno falta de recursos de red.

Se nos agotan las fibras, las *lamdas* y regeneradores disponibles para ser ocupados y cada vez llegan llamadas con una duración más larga, resultado imposible establecer los enlaces y bloqueo de la red.

Pero ahora nos podemos preguntar: ¿Es por culpa de falta de *lamdas* que la red se bloquea? ¿Es culpa de la falta de regeneradores? ¿Es culpa de ambos?

Seguiremos simulando centrándonos comparar *lamdas* y regeneradores, con lo que todas las simulaciones se realizarán con el mismo protocolos de elección de camino.

3.7.5 Rendimiento de la red con regeneradores

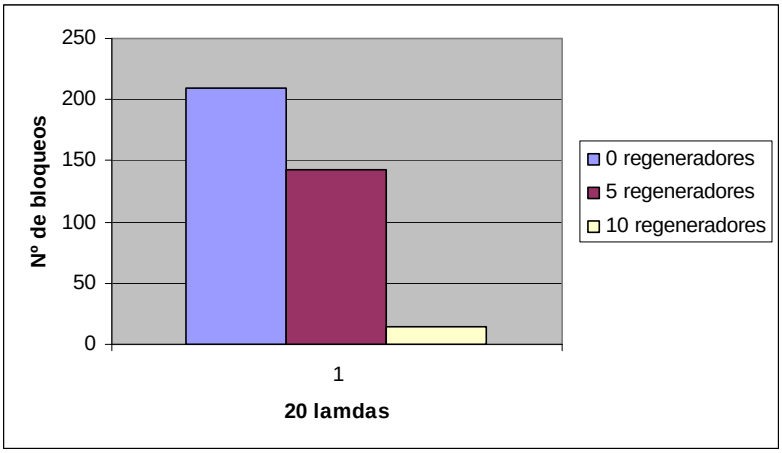
Condiciones iniciales:

- RedIris
- Número de nodos coincidentes=2
- 5000 llamadas por lista origen-destino
- Número de caminos=4
- Número de lamdas=20
- Número de fibras=2
- Protocolo elección= balanceado
- Erlangs= 18

Lista de llamadas:

Nodo origen	Nodo destino
7	16
17	9

Tabla 3.5



Gráfica 3.4

Comentarios:

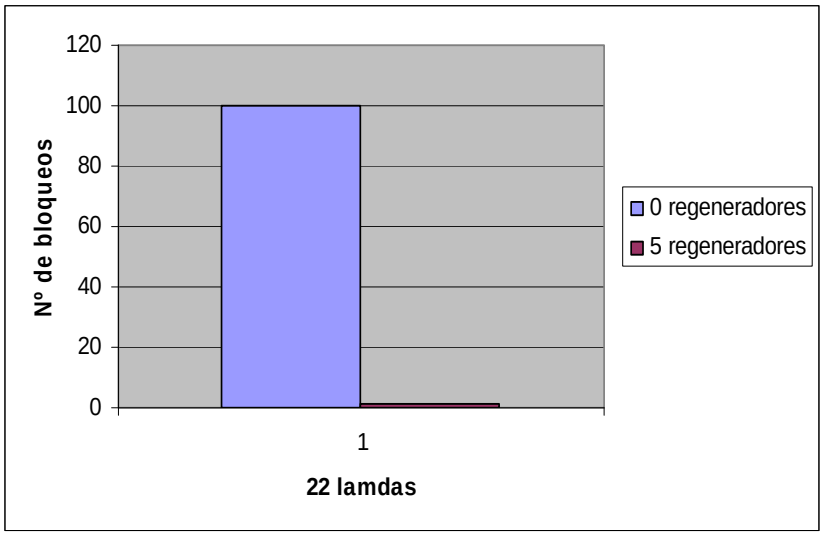
En este caso hemos buscado la cantidad de bloqueos totales que se producían en la red cuando esta trabajaba sin regeneradores, más tarde pusimos 5 regeneradores en cada enlace y después 10.

Vemos que en el tercer caso casi eliminamos el número de bloqueos en la red. Este resultado nos da más esperanza de que la capacidad de conmutación de la red, le aporte a ésta, mucha más eficiencia.

Comparando 0 regeneradores con 10, el cálculo sube a casi 10 veces más eficiente.

Seguidamente simularemos la misma carga de red, sobre los mismos nodos origen y destino, pero esta vez incrementando el número de *lamdas*.

3.7.6 Comparar: Lamdas - regeneradores ante incremento de lamdas



Gráfica 3.5

Comentarios:

A la vista de los resultados de la **gráfica 3.5**, como ya hemos ido concluyendo durante las anteriores simulaciones, incorporar más *lamdas* en nuestros enlaces, aumenta notablemente la eficiencia de la red.

En este caso incrementar el número de *lamdas* provoca que la cantidad de bloqueos existentes cuando la red trabaja sin regeneradores, disminuya a la mitad, y si incorporamos regeneradores, el número de bloqueos prácticamente desaparezca.

Es por ello, que de los resultados obtenidos hasta ahora se observa como cuando una red trabaja a media carga, estado en el que tendremos la red la mayor parte del tiempo, el hecho de poder tener elementos que conmuten *lamdas* aumenta notablemente la eficiencia.

Ya hemos tratado con una de las dos topologías de red disponibles, ahora seguiremos haciendo comparaciones a ver como se comporta una red mallada como es NSF.
Pero ahora nos interesa ser más precisos en los bloqueos, y trataremos de afinar la configuración de los recursos de la red, a ver si es posible aplicar modificaciones sólo en enlaces concretos; consiguiendo aumentar el valor de los datos recogidos del simulador de cara a un usuario que utilice la herramienta de simulación.

3.7.7 Comparar: Lamdas - regeneradores en una red mallada

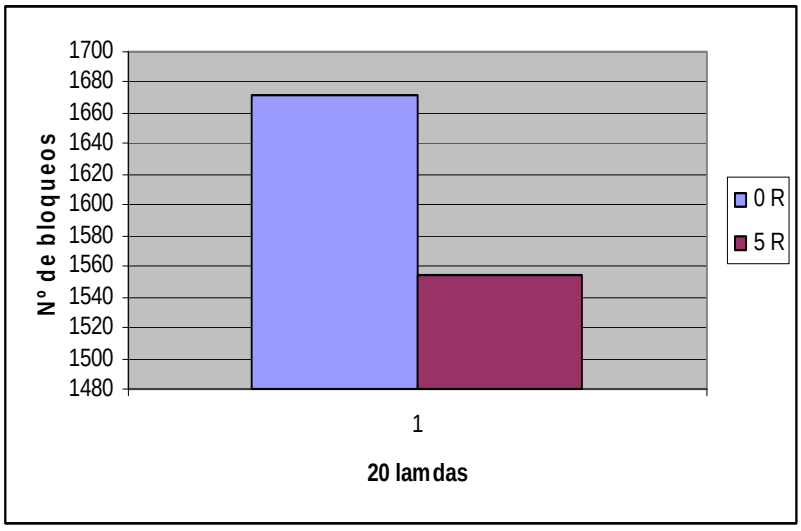
Condiciones iniciales:

- Red: NSFUSA
- Número de nodos coincidentes=2
- 5000 llamadas por lista origen-destino
- Número de caminos=4
- Número de lamdas=20
- Número de fibras=2
- Protocolo elección= balanceado
- Erlangs= 18

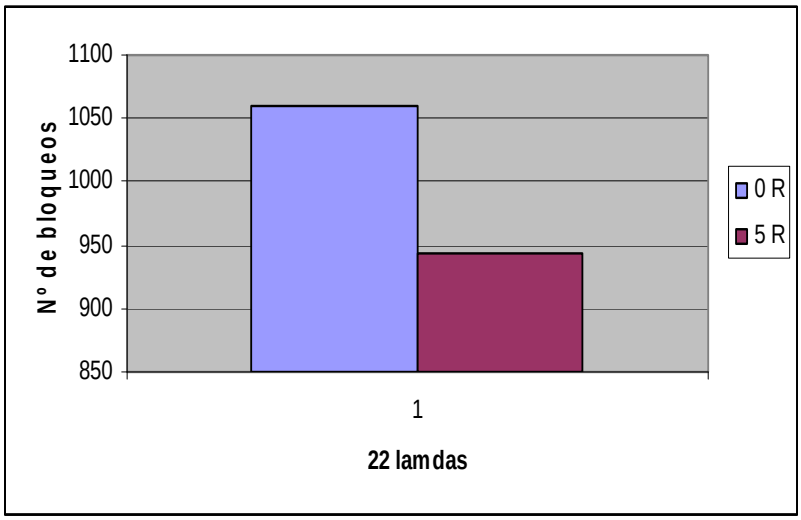
Lista de llamadas:

Nodo origen	Nodo destino
13	0
14	1
4	7
0	12

Tabla 3.6



Gráfica 3.6



Gráfica 3.7

Comentarios:

En la simulación de la gráfica 3.6, utilizamos los resultados del simulador para aplicar una corrección el los enlaces que presentan más bloqueo por culpa de que no hay un regenerador disponible.

La corrección es:

Enlace	Número de regeneradores
1 - 3	+ 5
1 - 4	+5
1 -10	+5
4 – 10	+5
10 - 12	+5

Tabla 3.8

Bien, los resultados nos marcan que tenemos un par de nodos bastante cargados como son el 1 y el 10.

Viendo la **gráfica 3.7** y consultando los valores se obtiene que sin regeneradores hay 1672 bloqueos en total, de los cuales 291 son por falta de regenerador. Después de la corrección, el total de bloqueos son de 1555 con 18 por falta de regenerador.

Por tanto el beneficio, que sacamos de incorporar una pequeña cantidad de regeneradores, es que disminuyen en un 7,5% la cantidad de bloqueos. Un dato que creo es bastante relevante.

Por otro lado observo como el tipo de bloqueo provocado por la falta de *lamdas*, habiendo regeneradores libres, aumenta de 0 a 614. Este dato es significativo si queremos decrementar

aun más la cantidad de bloqueos. Y la solución será indiscutiblemente invertir en más *lamdas* en los enlaces más saturados.

A raíz de esta deducción, realizo una simulación representada en la **gráfica 3.7** donde he aumentado la cantidad de *lamdas* que hay por enlace.

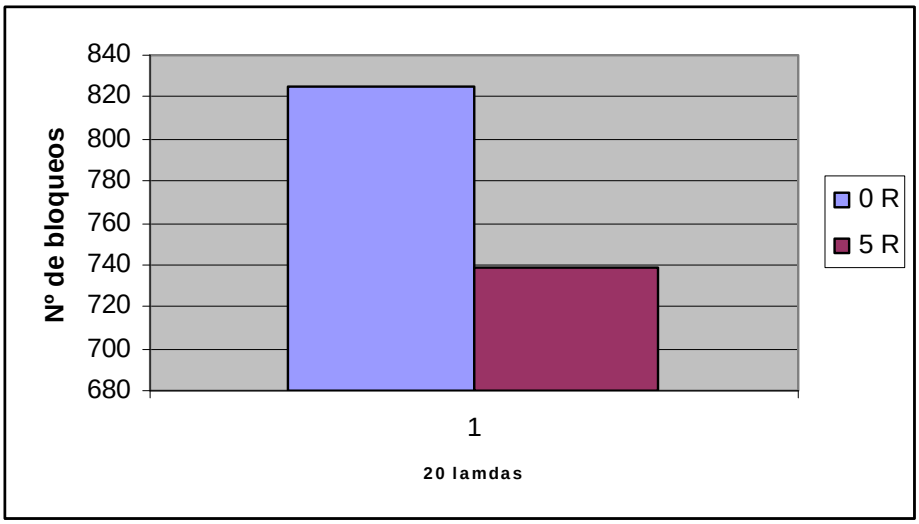
Comentamos los resultados. Teniendo la corrección de regeneradores en los mismos puntos en número de bloqueos que tenemos solo por el hecho de haber aumentado las *lamdas* pasa de 1672 a 1060 un 47% menos.

Por otro lado, en estas nuevas circunstancias con un incremento ligero de las *lamdas* también se observa que el rendimiento de los regeradores también aumenta. Ya que el dato de bloqueo con 5 regeneradores y 22 *lamdas* es 943 bloqueos, lo que es un 12,5% menos que sin regeneradores.

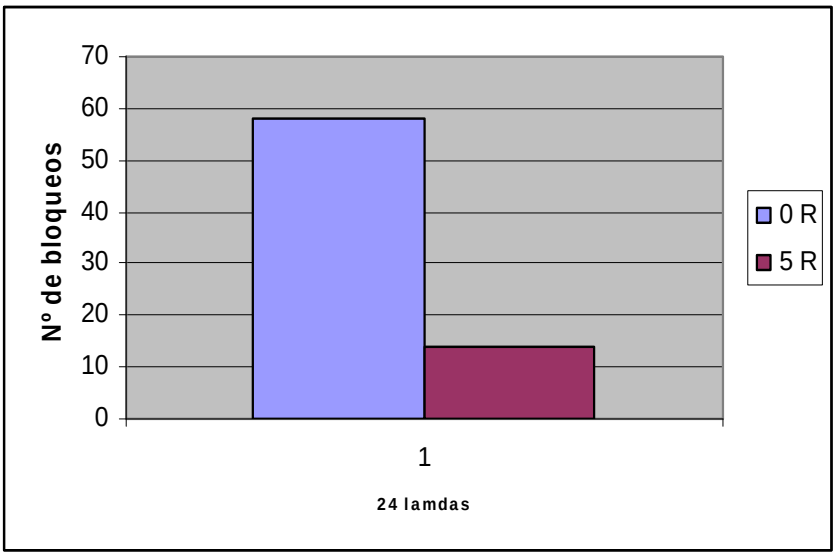
3.7.8 ¿Incrementando lamdas aumenta el rendimiento de los regeneradores?

Nodo origen	Nodo destino
13	0
14	1
4	7
0	12

Tabla 3.9



Gráfica 3.8



Gráfica 3.9

Comentarios:

Después de haber observado como se incrementa la eficiencia de los regeneradores aumentando el número de *lamdas* en la red, hacemos otra simulación para fidelizar este dato.

Y creo que los nuevos datos confirman las sospechas.

Comparando los resultados obtenidos en la **gráfica 3.8**, podemos observar que la red sin regeneradores tiene un bloqueo 58 llamadas, pero al aplicar los 5 regeneradores en los enlaces de la tabla CC, el número total de bloqueos es 14, por tanto más de un 30% de beneficio.

De esto se puede pensar en la idea de aumentar notablemente el rendimiento de los equipos de conmutación, aumentando ligeramente la capacidad de la fibra.

Como en casi todos los proyectos de ingeniería hay un punto de compromiso entre dos o más variables.

En este caso el simulador nos puede ayudar a ajustar este compromiso entre regeneradores y *lamdas*.

Capitulo 4

Conclusiones y perspectivas

Dentro de las necesidades humanas está la de comunicarse los unos con los otros. Con la tecnología actual, dejamos esta necesidad humana a las redes de telecomunicaciones.
Las redes de telecomunicaciones son gestionadas por operadores que se encargan de su puesta en servicio, mantenimientos, control del funcionamiento y dar de alta a los usuarios que requieren de servicios de telecomunicaciones.
Este proyecto va dirigido a la parte de puesta en servicio de una red.
Para llegar a ello, será necesario hacer una planificación y luego confirmar que dicha planificación es correcta.
La primera parte de la planificación será tener conocimiento de las necesidades, luego, según esas necesidades, dimensionar la red. Para tener conocimiento de que el diseño es correcto utilizaremos una herramienta de simulación para comprobar como se comporta nuestra red delante de las necesidades requeridas.
Utilizando el simulador y aplicándole datos reales, topologías de red por ejemplo, hemos podido conocer como responde una red y las como varía su comportamiento cuando aplicamos ciertos cambios o le añadimos ciertas capacidades, por ejemplo la conmutación de *lamdas*.
Hemos dirigido este simulador a las redes ópticas semitransparentes, ya que son los trabajos de planta actuales y hacia donde están dirigidos los trabajos actuales.
Hemos introducido conocimientos de DWDM, conmutación de *lamdas*, y la arquitectura ASON, la cual contiene una serie de protocolos para automatizar la conmutación. Todos estos conocimientos han tenido que ser profundamente analizados para poder extraer modelos. Los modelos han sido programados en un lenguaje que sabe interpretar una computadora, la cual nos ayudará a obtener los resultados que buscábamos.
Los resultados hallados por el simulador son bastante concluyentes ya que, bajo mi punto de vista, deja muy claro el camino a seguir para la planificación de una red óptica.
He diseñado una aplicación en lenguaje C, dirigida a usuarios técnicos, sin complejidad visual, para buscar la efectividad y el rendimiento de sus cálculos.

Después de diseñar el simulador hemos realizado una muestra de su funcionamiento con dos topologías reales y podemos sacar conclusiones firmes sobre la red nacional RedIris y la red americana NSF.

En el apartado 3.7.2 comparamos una topología parcialmente mallada (RedIris) y otra totalmente mallada (NSF). La solución es que una red totalmente mallada resulta mucho más eficiente que una parcialmente mallada, por tanto los esfuerzos de los responsables españoles sería en mallar más la red y no depender tanto de un nodo central.

Después hemos comparado cuatro algoritmos de elección de camino en el apartado 3.7.3, descubriendo que el protocolo más efectivo con bajas cargas en el que elige el camino más corto y con menos carga, pero que pierde rendimiento, convirtiéndose el protocolo "Least loaded balanceado" el que gestiona mejor los caminos a elevada carga.

Después de conocer el rendimiento de la topología y el protocolo de elección de camino, me ha interesado conocer como se comporta la red con conmutadores. Ya que, como hemos descrito en el capítulo 1 de la memoria, la evolución de las redes ópticas es hacia convertirlas en totalmente automatizadas y transparentes.

Con los resultados de la **gráfica 3.3** observamos como aumenta el rendimiento de la red al utilizar conmutadores, aunque también se muestra, que si se lleva a una red óptica a sus límites de capacidad, no basta con la posibilidad de poder conmutar *lamdas* para solventar el problema de congestión sino que tendremos que aumentar el número de *lamdas* y fibras para descongestionar.

Para reafirmar el dato anterior he llevado a cabo las pruebas del apartado 3.7.6

Por último después de observar el comportamiento global del sistema, me hice la pregunta del apartado 3.7.8, y puedo afirmar que un incremento pequeño del número de *lamdas* en una red, incrementa espectacularmente el rendimiento de los regeneradores.

El simulador nos resuelve el problema de bloqueo que podemos tener en una red, nos dice que cantidad de bloqueo y donde se encuentra.

Después se pueden tomar las decisiones oportunas para solventar esos problemas.

El punto más importante de poder utilizar una herramienta de simulación es rendibilizar enormemente los costes de implantación de la tecnología.

En este proyecto se presenta la primera versión del simulador para redes semitransparentes. Pero considero que puede crecer aun más. Como cualquier aplicación informática, el programa se puede ir adecuando a los usuarios, después de recibir sus sugerencias, por ejemplo, y después ampliar sus capacidades con nuevas ideas como las que describo a continuación.

También me han surgido nuevas ideas para poder incorporar a la aplicación que la ayudarían a crecer mucho más. La primera de ellas es dar la opción de poder configurar cada enlace con

una cantidad propia de fibras y *lamdas*. Otra posibilidad, que considero muy interesante, es la de poder desactivar nodos en cualquier momento de la simulación, para poder simular la tolerancia a fallos de una red. Aunque en esta aplicación utilizamos un protocolo de "routing", similar al OSPF, seguiría investigando otros protocolos.

Para acabar considero muy gratificante haber hecho este proyecto ya que me ha hecho descubrir la importancia de un método de trabajo y la planificación de éste, de cara a los trabajos que realizaré en el mundo laboral como ingeniero.

Referencias

[1] ITU-T Recommendation G.8080/Y.134 Amendment 1 (03/2003), *Architecture for the automatically switched optical network (ASON)*.

[2] “Funcionalidad de las redes conmutadas ASON”. Carlos J. Fuentemayor Toro Revista AHCIET: revista de telecomunicaciones, Nº. 94, 2003

[3] “Implementing the ASON: interest and critical issues for the operator” Jacques Robadey, Carmen Mas, Evi Zouganeli and Daniel Rodellar Swisscom, Switzerland; Intracom S.A., Greece; Telenor R&D, Norway

[4] “Programación en turbo C”: Herbert Schildt ed. Mcgraw hill 1988

[5] Appendix E – Queuing Theory and Erlangs by Peter A. Stark

[6] “Sistemas y redes ópticas de comunicaciones” Autor: Martín Pereda, José Antonio Editorial: Pearson Educacion

[7] "Optical fiber communications", Gerd Keiser. 3ª edición. McGraw Hill, 2000.

[8] “Fundamentos de Comunicaciones Ópticas” J. Capmany, F.J. Fraile-Pelaez, J. Martí. Edit. Síntesis, 1999.

[9] "Generalized MultiProtocol Label Switching (GMPLS) Architecture", P. Ashwood-Smith et al., internet-draft November 2001, http://search.ietf.org/internet-drafts/draftietfccamp-gmpls-architecture-01.txt

[10] “OPTICAL WDM NETWORKS” Sivalingam, K. M., and Subramaniam, Principles and Practice. Kluver Academic Publishers, 2000.

[11] http://www.radioptica.com/Fibra/dwdm.asp;

[12] http://www.wonesys.com/esp/aplicaciones.php?id=2;

[13] http://www.cisco.com/univercd/cc/td/doc/product/mels/dwdm/dwdm_fns.htm.

[14] rfc1247.txt

[15] rfc2205.txt

[16] rfc3037.txt

[17] rfc3473.txt

[18] rfc3475.txt

[19] rfc3945.txt

[20] rfc4139.txt

[21] rfc4258.txt

[22] rfc4394.txt