Arquitectura y diseño de un modelo de red OBS para simulación

Felix Espina, Javier Armendariz, Mikel Izal, Daniel Morató, Eduardo Magaña
Universidad Pública de Navarra, Campus de Arrosadía s/n, E-31006 Pamplona
e-mail: {felix.espina, javier.armendariz, mikel.izal, daniel.morato, eduardo.magana}@unavarra.es

Resumen

Optical Burst Switching (OBS) es una nueva tecnología de conmutación óptica capaz de soportar una gran demanda de ancho de banda en backbones ópticos con Wavelength Division Multiplexing (WDM). Muchos investigadores están interesados en el estudio de esta propuesta emergente y la búsqueda de sus parámetros y entornos de funcionamiento óptimos. Sin embargo se encuentran con el gran handicap de que no existen muchas testbed para su estudio físico, ni tampoco herramientas de software óptimas para su estudio mediante simulaciones. En este trabajo se presenta un modelo de simulación de OBS para el simulador de eventos discretos OMNeT++. Este modelo permite estudiar tanto los nodos frontera, como los nodos del core, así como enlazar la red OBS con otras redes de datos soportadas en OMNeT++, principalmente IP. Además, el diseño presenta una gran modularidad lo que permite modificar fácilmente el modelo OBS para incluir futuras propuestas que se hagan sobre esta tecnología.

Palabras Clave

Optical Burst Switching, OBS, OMNeT++, simulación, arquitectura, diseño

I. Introducción

Se ha propuesto Optical Burst Switching [1] (OBS) como un sistema de conmutación óptica con ventajas sobre las otras propuestas de conmutación óptica, como Optical Circuit Switching (OCS) y Optical Packet Switching (OPS). OBS tiene un gran interés entre los investigadores como la arquitectura óptica capaz de soportar una gran demanda de ancho de banda en backbones ópticos con Wavelength Division Multiplexing (WDM). OBS es una solución intermedia entre OCS y OPS. OBS tiene una gran utilización del ancho de banda, una baja latencia de establecimiento, sólo requiere de conmutación óptica de moderada velocidad, complejidad de procesamiento media y se adapta muy bien al tráfico intermitente.

En OBS la unidad de transporte básica es la ráfaga, que contiene un número de paquetes IP. Normalmente el agrupamiento de paquetes IP suele ser por destino, pero se puede hacer por cualquier criterio o combinación de criterios: dirección origen o destino, puertos de origen o destino, protocolos sobre IP, tipo de datos, etc. También se han propuesto diversos criterios para decidir cuándo enviar la ráfaga. Por ejemplo, se puede acumular todos los paquetes que lleguen durante un tiempo (con lo que su tamaño es variable) o se puede acumular exactamente una cantidad dada de bytes [2] (con lo que el tiempo para formar una ráfaga es variable). Las ráfagas se ensamblan y desensamblan en los nodos frontera o edge nodes, y cada formador de ráfagas se denomina burstifier. Una ráfaga se puede considerar como un paquete óptico que va desde el nodo entrada a la red OBS hasta el nodo salida de la red OBS sin sufrir ninguna conversión óptica-electrico-óptica (OEO). Para unir todas las parejas origen-destino y crear la red OBS se usan los nodos del core, que en el fondo son conmutadores ópticos o Optical Cross Connectors (OXC) gobernados por una Unidad de Control electrónica (UC).

En OBS se usa señalización fuera de banda con un tiempo de offset de retraso entre la señalización y la ráfaga. Así se obtiene separación espacial y temporal entre la ráfaga y el Burst Control Packet (BCP) [3], también denominado Control Packet Header (CPH). Esto es una característica específica de OBS que le da una gran capacidad de manejo y flexibilidad de red.

Generalmente OBS usa esquemas de señalización unidireccionales e iniciados por el origen, es decir, las ráfagas se envían a la red OBS sin esperar a la confirmación del éxito o del fracaso del intento de reserva del path completo hasta el destino. Así, las ráfagas pueden competir por los mismo recursos. Esta es la principal causa de la pérdida de ráfagas en OBS y sucede cuando el número de intentos simultaneos de reservas de ráfagas en una fibra de salida de un nodo del core es superior al número de longitudes de onda disponibles. También se puede dar que el BCP y su ráfaga asociada se acerquen tanto en tiempo, que el nodo del core no tenga tiempo a programar la conmutación y tenga que descartar la ráfaga. En cualquiera de los dos casos, si no se dispone de nodos del core con técnicas avanzadas como buffer ópticos, esto implica la pérdida de la ráfaga y que el rendimiento de OBS se vea afectado, bien en la utilización del ancho de banda o en la latencia por la retransmisión de la información perdida.

La Fig. 1 presenta un modelo de arquitectura de red OBS.

Este trabajo presenta un modelo de red OBS para el simulador OMNeT++[1] con el que estudiar esta nueva técnología. Actualmente existen propuestas teóricas de las diferentes partes de OBS (burstifiers por timer o tamaño, técnicas de señalización) y muy pocas implementaciones experimentales o testbed [4][5][6][7][8][9]. Por tanto se considera importante como primer paso poder estudiar todas estas propuestas usando herramientas de simulación.

Existen muchos simuladores de red [10]: el open source NS-2, el comercial OPNET, el específico para redes inhalambricas Qualnet, etc. Para este trabajo se ha escogido OMNeT++ [11]. A diferencia del resto su función principal no es la de simulador

[1] http://www.omnetpp.org

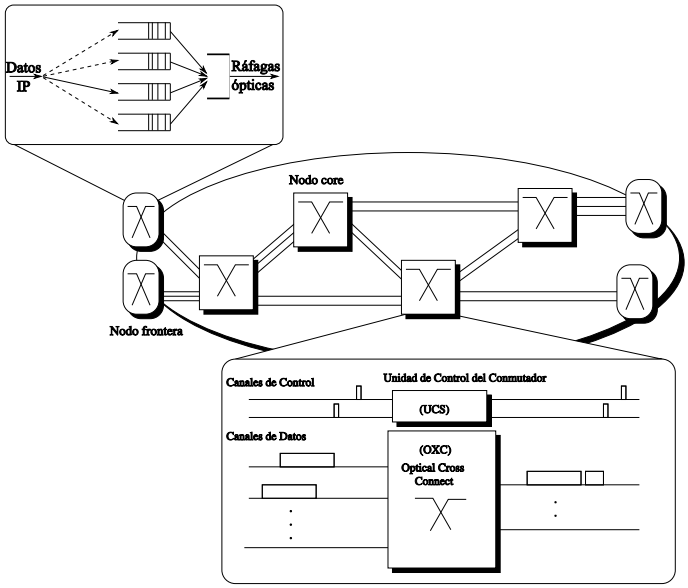

Figura 1. Esquema de arquitectura de red OBS

de redes, sino la de simulador genérico de eventos discretos con el que poder simular desde el funcionamiento de un disco duro hasta el comportamiento de una red Ethernet. Esto le da una gran versatilidad y capacidad de ser explotado en diversos ámbitos. OMNeT++ es de código abierto y tiene una *Academic Public License* que lo hace gratuito para usos no-comerciales. Está disponible para todas las plataformas comunes incluyendo Windows, Mac OS/X y Linux. Todo el código fuente está en C++ y se puede compilar con gcc o con el compilador Microsoft Visual C++.

Existen otros dos simuladores que también podrían ser los escogidos para este trabajo: OPNET[2] y NS-2[3].

OPNET tiene una licencia anual bastante cara, incluso para investigación científica, mientras que como se ha comentado OMNeT++ es gratuito para este fin. Además, pagando la licencia sólo se tiene acceso al código fuente de los modelos, pero no del kernel del simulador. Una diferencia significativa es que los modelos de OPNET son siempre de topología fija, mientras que en OMNeT++ es fácil tener topologías parametrizadas. En OPNET la manera habitual y preferida de definir la topología de red es mediante su editor gráfico, que guarda la red en un formato binario propietario que dificulta la posibilidad de generar estas topologías mediante programación (hay que usar una API específica para C de OPNET). En cambio en OMNeT++ las topologías se guardan en ficheros de textos planos que son fáciles de manipular. La principal ventaja de OPNET sobre OMNeT++ es su gran librería de modelos de protocolos (incluido uno para OBS), mientras que su naturaleza cerrada hace que la programación y la solución de problemas sea difícil.

NS-2 es el simulador de red más usado en el ámbito académico, pero no tiene la separación entre kernel de simulación y modelos que tiene OMNeT++: la distribución NS-2 contiene los modelos junto a la infraestructura del simulador como una única entidad inseparable. El objetivo de NS-2 es crear un simulador de red, mientras que el objetivo de OMNeT++ es ofrecer una plataforma de simulación. A NS-2 le faltan muchas herramientas y componentes de infraestructura que OMNeT++ tiene: soporte para modelos jerárquicos, interfaz gráfica de usuario (GUI) de entorno de simulación, separación entre modelos y experimentos, herramientas gráficas de análisis, algunas funcionalidades de simulación como multiples streams RNG, etc. Todo esto debido a que NS-2 se ha centrado en desarrollar los modelos de simulación, en vez de la infraestructura de simulación. Además, aunque NS-2 es open source y multiplataforma, en Windows pierde ciertas funcionalidades y es necesario compilar y usarlo mediante Cygwin[4], un entorno Linux para Windows.

Hasta ahora, aunque sí se ha propuesto algún modelo de simulación OMNeT++ para OBS [12], no se han hecho públicos. En cambio sí existen modelos OBS públicos para otros simuladores, como NS-2 [13]. Por esto se han realizado los módulos OMNeT++ necesarios (accesibles vía web[5]) para poder estudiar en profundidad las diferentes propuestas de la tecnología OBS. La arquitectura del modelo OBS de este trabajo es diferente al presentado en [12], basado en la propuesta teórica de [14], puesto que asume que existen dos tipos de nodos: los nodos frontera (edge node) y los nodos del core (core node). El primero, el nodo frontera, se encarga de introducir (y extraer) tráfico óptico a la red OBS usando los burstifiers para crear las ráfagas, pero no tiene tráfico óptico en tránsito pasando por él. El segundo, el nodo del core, sólo se encarga de tráfico óptico en tránsito, sin tener capacidad de introducir (o extraer) tráfico óptico de la red.

OMNeT++ provee de la maquinaria y las herramientas básicas para escribir esos componentes y esas simulaciones, en vez de proveer de componentes de simulación para redes de computadores, colas de redes y otros dominios. Es un framework,

[2] http://www.opnet.com/
[3] http://nsnam.isi.edu/nsnam/
[4] http://www.cygwin.com/
[5] **https://www.tlm.unavarra.es/investigacion/proyectos/strrong/soft/**

más que un programa de simulación. Para cada área específica de aplicación se han desarrollado modelos específicos, como el INET Framework para simulación de redes IP. El desarrollo de estos modelos es completamente independiente de OMNeT++, incluyendo sus ciclos de publicación, hasta tal punto que muchos son desarrollados por equipos externos a OMNeT++, aunque se publiquen en la misma página web de OMNeT++.

Un modelo OMNeT++ consiste en módulos que se comunican pasándose mensajes. Estos mensajes se pasan por las conexiones que se crean entre las **puertas**, o interfaces de entrada/salida, de los módulos. Los módulos básicos se denominan **módulos simples**. Los módulos simples se escriben en C++ usando las clases de la librería de simulación. Los módulos simples se pueden agrupar en **módulos compuestos**, Fig. 2, y estos en otros, hasta obtener la complejidad necesaria para describir el componente deseado. Los módulos compuestos se describen mediante el lenguaje NED usando ficheros de texto plano. Por ejemplo, cada interfaz de un router puede ser un módulo simple y el router sería un módulo compuesto. En OMNeT++ las redes son un tipo particular de módulos compuestos sobre las que se pueden realizar simulaciones.

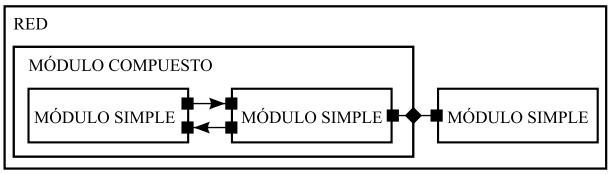

Figura 2. Estructura de módulos en OMNeT++

La información que se pasa entre los módulos y los eventos programados para el simulador se describe mediante **mensajes**. Un mensaje es una estructura C++ específica de OMNeT++ que contiene variables para describir la información deseada. Por ejemplo, mediante un mensaje se puede describir un paquete IP, en el que cada variable de la estructura es un campo de la cabecera IP. Los mensajes se describen mediante un fichero de texto y se pueden ampliar para describir cualquier dato. OMNeT++ preprocesa la estructura de definición del mensaje generando automáticamente el código C++ con el que poder manejar el mensaje mediante una función *setter* y una *getter* para cada variable de la estructura.

A la hora de usar un módulo, el usuario no sabe si es un módulo simple o compuesto, sólo necesita conocer cómo se tienen que conectar sus puertas y qué parámetros necesita. Así, se puede ampliar el módulo subdiviendolo internamente en módulos simples o simplificarlo pasandolo a un único módulo simple, sin que el usuario final y las simulaciones ya preparadas para usar esos módulos se den cuenta del cambio.

El resto del paper se organiza de la siguiente manera. La sección II presenta el modelo OBS que se va a implementar en OMNeT++. En las secciones III, IV y V se explica pormenorizadamente la implementación realizada del modelo OBS, remarcando su versatilidad y modularidad. En la sección VI se presenta un ejemplo de uso de los módulos OBS desarrollados. Por último, la sección VII presenta las conclusiones.

II. El modelo de red OBS

Tanto el nodo frontera como el nodo del core se implementan en módulos compuestos de OMNeT++: ***OBS_edgeNode*** y ***OBS_CoreNode***. En las siguientes secciones se presentarán desde el punto de vista funcional, remarcando su capacidad de configuración y su modularidad.

III. Nodo Frontera - módulo OBS_EdgeNode

A la hora de implementar un nodo frontera, este se puede ver como un router al que por lo menos se le ha añadido una interfaz OBS, o como un router que a su salida se le ha conectado un equipo OBS. Cualquiera de los dos casos son factibles y fáciles de hacer en OMNeT++. En este modelo de simulación se ha implementado el primer caso, el router con interfaz OBS, basándose en los módulos del router básico de OMNeT++.

Cuando actúa como nodo de ingreso, módulo ***assembler***, el nodo frontera es responsable de tomar la decisión de enrutamiento, agregar o ensamblar el tráfico entrante en ráfagas y planificar la transmisión de la ráfaga en el canal de salida.

En OBS el tráfico IP de entrada se agrega a ráfagas en el nodo frontera según el destino óptico. Esta agregación se realiza en los burstifiers y por lo menos tiene que haber un burstifier por destino (o nodo frontera de salida). Se puede tener más de un burstifier por destino para poder diferenciar el tráfico (y las ráfagas que se generan) dependiendo de las direcciones IP de origen o destino, del puerto de origen o destino y del protocolo IP usado. En el modelo propuesto en [12] no se le da importancia a la manera de formar las ráfagas, puesto que sólo se quiere estudiar el comportamiento del backbone OBS. Pero este planteamiento no permite usar el modelo OBS en simulaciones que conecten la red OBS con otras redes de datos. El modelo propuesto en este trabajo sí permite hacer esto. Además, así es posible estudiar el tráfico que componen las ráfagas y cómo cambia esta generación dependiendo del tráfico de entrada y los esquemas de agregación usados en los burstifiers. La implementación actual del nodo frontera soporta los esquemas más extendidos: timer, tamaño [2] y número de paquetes [15]. Y, por supuesto, la mezcla de estos esquemas. Pero añadir un nuevo esquema sólo implica la modificación de un único módulo simple de OMNeT++, el ***packetBurstifier***.

El forwarding óptico se hace mediante un esquema tipo *conmutación óptica de etiquetas*, similar a LOBS [16]. Cada ráfaga lleva una etiqueta. En los nodos del core se usa esta etiqueta, junto con el puerto y la longitud de onda por la que ha llegado,

como parámetros de forwarding. Las etiquetas pueden cambiar en cada salto o nodo del core. La etiqueta inicial la pone el burstifier del nodo frontera que genera la ráfaga. La elección de qué burstifier genera la ráfaga se hace en dos pasos. Primero se decide la interfaz óptica por la que el paquete IP puede llegar a su destino. Para eso se usa la tabla de rutas del nodo frontera. Una vez decidido el interfaz óptico, el ***dispatcher*** de esa interfaz, mediante sus criterios de clasificación, decide en qué burstifier almacenar el paquete IP. La etiqueta que se le da a la ráfaga generada es la etiqueta que se configura en la simulación a ese burstifier.

Una vez generada la ráfaga en el burstifier, esta se pasa al nivel de enlace OBS. Este nivel de enlace se ha implementado como una cola en la que almacenar las ráfagas hasta su transmisión. El tamaño de esta cola, tanto en bits como en número de ráfagas, es configurable para cada simulación y nodo frontera. Cuando una ráfaga no cabe en la cola, simplemente se descarta.

Para planificación actualmente se usa el esquema más básico propuesto, denominado **Horizon o LAUC** [17]. En este esquema, cuando la ráfaga llega a la cola se calcula cuál es el horizonte más cercano en que cualquiera de las longitudes de onda queda libre. Se planifica para ese instante y esa longitud de onda, y se modifica el horizonte de la longitud de onda. Para que los nodos del core tengan tiempo de procesar la información de control antes de que llegue la ráfaga, el BCP asociado a la ráfaga se envía antes. El offset temporal entre el BCP y la ráfaga tiene un máximo y un mínimo. Inicialmente el envío del BCP se planifica para ese offset máximo. Si por congestión en el canal de control, este no puede salir del nodo frontera con el offset mínimo de separación, el BCP y la ráfaga se replanifican.

Cuando actúa como un nodo de salida, módulo ***disassembler***, el nodo frontera realiza la operación inversa, es decir, desagrega las ráfagas en paquetes y los procesa como cualquier otro tráfico.

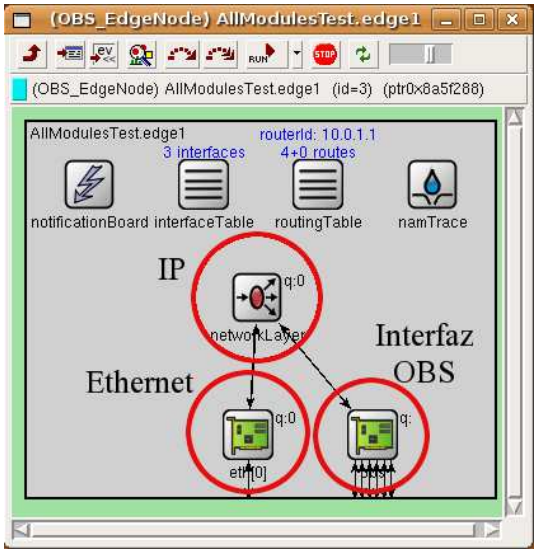

Figura 3. Nodo Frontera

En la Fig. 3 se muestra la estructura interna del nodo frontera. Como ya se ha comentado anteriormente el nodo frontera se basa en el router simple de OMNeT++, añadiéndole una interfaz OBS. Así, igual que en el router simple, tenemos un ***interfaceTable*** con la descripción y parámetros de red de las interfaces de red, y un ***routingTable*** con la tabla de rutas del router. En el ejemplo usado para la figura 3, el nodo frontera tiene una interface Ethernet para que una red de conmutación de paquetes acceda al backbone OBS, y una única interfaz OBS con la que conectarse con un nodo del core. Para cada simulación y nodo frontera, tanto la tabla de rutas como el número de interfaces OBS son facílmente configurables.

La interfaz OBS es la responsable de hacer todo el trabajo relacionado con la parte OBS. Los parámetros interesantes son configurables independientemente para cada interfaz OBS. La red tiene que mantener cierta coherencia, p.e. no se puede conectar un puerto de salida del nodo frontera de 5 longitudes de onda con un puerto de entrada del nodo del core de sólamente 3 longitudes de onda. OMNeT++ considera cada longitud de onda de una fibra óptica como un enlace independiente, por lo que las dibuja por separado. Esto se ve en las figuras 3 y 4, donde sólo hay una fibra óptica de 5 longitudes de onda (4 de datos y una de señalización).

La interfaz OBS se implementa mediante los siguientes 2 submodulos:

III-A. assembler

Es un módulo complejo que realiza el trabajo relacionado con el funcionamiento del nodo frontera como nodo de ingreso. Se encarga de tomar la decisión de enrutamiento, ensamblar el tráfico entrante en ráfagas y planificar la ráfaga. El ***assembler***, como se puede ver en la figura 5, está compuesto por tres módulos simples.

1. ***dispatcher - clasificador***
   En base a las reglas de clasificación, se encarga de enviar a cada burstifier el tráfico entrante que le corresponde. Las

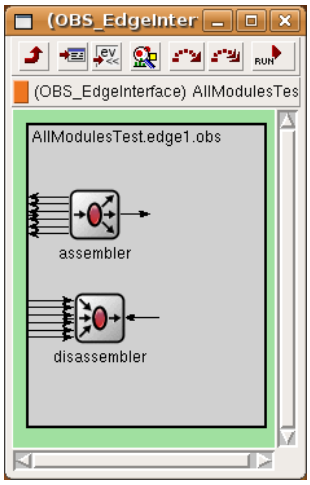

Figura 4. Interfaz OBS del Nodo Frontera

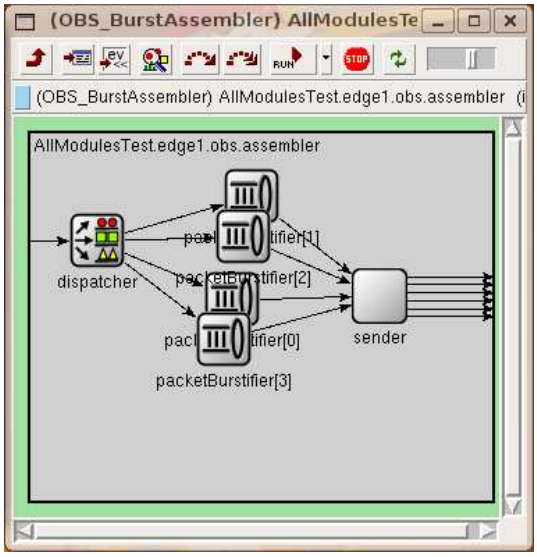

Figura 5. Assembler del Nodo Frontera

reglas de clasificación de cada ***dispatcher*** se configuran en un fichero que se le pasa en la inicialización. En el fichero de reglas de cada nodo frontera tiene que haber una única regla para cada uno de sus burstifier. Las reglas pueden tener tantas parejas *parámetro-valor* como tenga que cumplir el tráfico que va dirigido a ese burstifier. El tráfico entrante que no cumpla ninguna regla se descarta. Por ejemplo, se puede hacer una regla para agrupar en el mismo burstifier todo el tráfico (TCP o UDP) que vaya a la dirección 10.1.1.1 y puerto 80.
Este módulo guarda estadísticas de los tamaños de los paquetes y el número de paquetes descartados.

2. ***packetBurstifier***
   Se encarga de ensamblar el tráfico entrante en una ráfaga usando el esquema de ensamblado y los parámetros que se le han configurado. Tiene parámetros para poder manejar los diferentes esquemas de ensamblado. Así, se puede configurar el tiempo máximo de ensamblado de una ráfaga, *timeout*, o el tamaño máximo de la ráfaga, *maxSize*. Pero también el tamaño de la cabecera, el tamaño mínimo de la ráfaga para un burstifier por *timeout* o si en un burstifier por *maxSize* el paquete que hace superar este límite se incluye en esa ráfaga o en la siguiente. Además, para dar mayor libertad, también se puede configurar el tiempo de offset máximo y mínimo para que sea diferente en cada burstifier.
   Este módulo guarda estadísticas del tiempo medio que pasan los paquetes en el burstifier y el número de paquetes por ráfaga.
   El planteamiento modular que se ha usado permite que en caso de querer incluir un nuevo esquema de ensamblado, el único módulo a modificar sea este.

3. ***sender - nivel de enlace***
   Es el módulo encargado de planificar y transmitir la ráfaga y su BCP asociado. En este módulo se puede configurar

tanto el tamaño de los BCPs como el tamaño de la cola del sistema donde almacenar las ráfagas mientras esperan su instante de transmisión. Es el único módulo del ***assembler*** que necesita saber el número de longitudes de onda de la fibra y su velocidad de transmisión.

III-B. disassembler

Es un módulo simple, que realiza el trabajo relacionado con el funcionamiento del nodo frontera como nodo de salida. Recibe las ráfagas, las desagrega y pasa los paquetes al nivel superior para que se procesen como cualquier otro tráfico.

IV. Nodo del Core - módulo OBS_CoreNode

Los nodos del core son responsables del procesamiento de los BCPs, de la conmutación de las ráfagas de una fibra de entrada a otra de salida sin conversión electro-óptica, y del mecanismo de contienda entre ráfagas. En OBS la señalización típicamente se hace fuera de banda. Normalmente transmitiendo el BCP asociado a una ráfaga en una longitud de onda exclusiva y diferente de las longitudes de onda para las ráfagas. En esta implementación se ha escogido usar para ello la primera longitud de onda de todas las fibras. En caso de querer modificar esto, sólo sería necesario modificar los módulos simples de entrada/salida: el ***Input*** y ***Output*** en el nodo del core y el ***sender*** en el nodo frontera.

Se han propuesto diferentes esquemas de señalización [18] [19], pero los protocolos más populares de señalización distribuida en OBS son Just-in-Time (JIT) [20] y Just-Enough-Time (JET) [21]. Los dos son esquemas de señalización unidireccionales e iniciados por el origen, es decir, las ráfagas se envían a la red OBS sin esperar a la confirmación del éxito o del fracaso del intento de reserva de un path hacia el destino. Se basan en el mismo principio, pero se diferencian en la duración de las reservas. JIT usa reserva inmediata desde que el BCP llega al nodo del core, mientras que JET retrasa la reserva del canal hasta la llegada estimada de la ráfaga. Como la información de señalización necesaria es diferente, en JET el BCP tiene que indicar cuándo se espera que llegue la ráfaga, el tipo de BCP usado en JET y JIT es diferente.

El retraso de la reserva hace que JET sea más eficiente que JIT, obteniendo mejores ratios de bloqueo y retrasos extremo-a-extremo más pequeños [14]. Por esta razón la implementación actual de este modelo de simulación OBS usa un esquema tipo JET, aunque se podría cambiar fácilmente a un esquema tipo JIT. Sólo sería necesario cambiar una pequeña parte del nodo del core, el módulo simple ***ControlUnit***. No sería necesario modificar el mensaje BCP porque el actual ya tiene todos los campos necesarios para señalizacion JIT.

Según el puerto, longitud de onda y etiqueta de entrada de la ráfaga, esta tiene asociadas en el ***ControlUnit*** unas longitudes de ondas que puede usar o *válidas* para cada puerto de salida. A la llegada del BCP, se escoge entre las longitudes de onda de salida *válidas* para su ráfaga asociada la que tenga el horizonte más cercano y menor del instante previsto de llegada de la ráfaga. Se planifica el ***Optical Cross-Connect (OXC)*** para que en ese instante de llegada conmute la longitud de onda de entrada con la longitud de onda de salida escogida y deshaga la conmutación una vez pasada la ráfaga. Actualmente el esquema asume que siempre hay un conversor de longitud de onda disponible para hacer esta conmutación entre longitudes de onda. Si no hay ninguna longitud de onda libre para ese instante de llegada, entonces simplemente el BCP se descarta y cuando la ráfaga llega, como su entrada no está conectada a ninguna salida, la ráfaga se pierde.

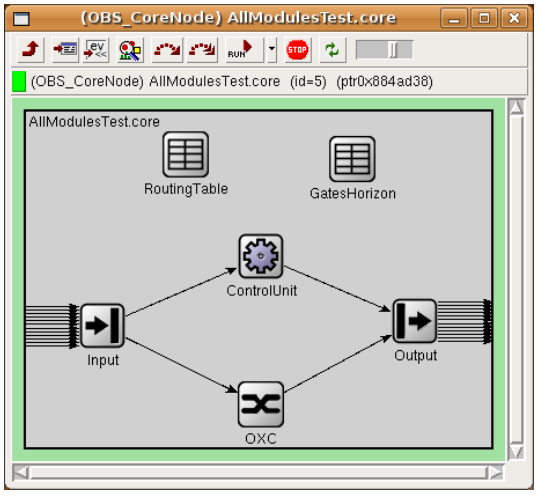

Figura 6. Nodo del Core

En la Fig. 6 se muestra la estructura interna del nodo del core. Se basa en realizar forwarding de las ráfagas de una fibra a otra mediante una matriz de conmutación dinámica configurable. Esta matriz de conmutación sigue las órdenes de la unidad de control. La unidad de control toma sus decisiones usando la *tabla de forwarding*, que indica el puerto y longitud de onda de salida según la etiqueta de la ráfaga. Actualmente esta tabla de forwarding se genera en la inicialización a partir de un fichero, y se mantiene inalterable durante la simulación. Se podría modificar la implementación del nodo del core para incluir una tabla de forwarding dinámica que cambiase su estado p.e. mediante un esquema de enrutamiento centralizado.

El nodo del core se compone de 4 submódulos: 3 módulos simples y uno compuesto.

IV-A. Input

Al haber escogido que la señalización (los BCPs) vayan en una longitud de onda exclusiva, es necesario separar de todas las fibras de entrada las longitudes de onda de señalización de las de transmisión. De esto se encarga el módulo ***Input***. Enlaza las longitudes de onda de señalización con la unidad de control (***ControlUnit***) y las de tráfico con la matriz de conmutación o ***Optical Cross-Connect (OXC)***.

IV-B. Output

Este módulo hace el trabajo inverso del módulo ***Input***: inyecta cada longitud de onda de señalización en su fibra correspondiente junto con el resto de longitudes de onda de tráfico.

Si en algún momento se desea modificar la longitud de onda que se usa para senalización OBS o usar señalización por medios alternativos como ADSL o enlaces de microondas, los módulos a modificar en el nodo del core serían estos dos. Tanto la matriz de conmutación como la unidad de control no se verían afectados y no sabrían que la señalización ha cambiado.

IV-C. OXC - Optical Cross-Connect

Es la matriz de conmutación óptica. Según le va indicando la unidad de control, conecta y desconecta las longitudes de onda de las fibras de entrada con las longitudes de onda de las fibras de la salida. El ***OXC*** no necesita saber qué tráfico pasa, cómo es la señalización OBS y ni siquiera si tiene completa capacidad de conversión de longitud de onda. Todo eso es responsabilidad y tarea de la unidad de control, por lo que este módulo es muy simple pero muy fácil de mantener ante cualquier cambio del nodo del core.

Se puede configurar el tiempo que necesita para hacer una conmutación. La unidad de control tendrá en cuenta este tiempo a la hora de planificar las conmutaciones, ya que para cuando llegue la ráfaga la conmutación tiene que estar completamente acabada o la ráfaga se corromperá.

IV-D. ControlUnit

La unidad de control es la encargada de realizar todo el procesamiento de la señalización y el forwarding y planificación de las ráfagas. Para eso se compone de tres módulos simples:

1. ***OE***
   El procesamiento de la señalización se realiza en el dominio electrónico, por lo que primeramente, como se ve en la Fig. 7 hace una conversión opto-eléctrica de los BCPs ópticos. En simulación esto sirve para emular el tiempo físico de procesamiento opto-electrónico, y poder añadir al mensaje BCP información adicional (puerto de entrada) que necesita la unidad de control.

2. ***ControlUnit***
   Con la información de los BCPs electrónicos y la tabla de forwarding busca entre las longitudes de onda *válidas* la que tenga el horizonte más cercano y menor al tiempo estimado de llegada de la ráfaga. Para el cálculo de este tiempo de llegada estimado, la unidad de control tiene en cuenta tanto el tiempo de conmutación mencionado anteriormente, como un tiempo de guarda configurable para cada nodo del core. Con este tiempo de guarda se puede configurar la separación mínima que tiene que haber entre dos ráfagas a conmutar hacia la misma longitud de onda de salida. El tiempo de procesamiento de la información de control es un parámetro configurable y constante por BCP.
   Las reglas de forwarding de cada nodo del core se cargan en la inicialización de un fichero. En este fichero se indica cual es la combinación de puerto, longitud de onda y etiqueta de salida para cada combinación de puerto, longitud de onda y etiqueta de entrada. Tanto para los puertos como para las longitudes de onda se puede usar el símbolo "*"para indicar cualquier fibra y cualquier longitud de onda. Con este sencillo fichero de configuración se puede implementar un forwarding tipo *conmutación óptica de etiquetas*.

3. ***EO***
   Realiza el proceso inverso del módulo ***OE***, la conversión electro-óptica de los BCPs electrónicos. Para ello, igual que el ***sender*** del nodo frontera, necesita conocer la velocidad de transmisión de las longitudes de onda.

Por cada BCP óptico la unidad de control tiene que hacer una conversión opto-eléctrica, procesar la información del BCP y una conversión electro-óptica. El BCP óptico tarda más tiempo en atravesar el nodo del core que la ráfaga óptica, por lo que el offset temporal entre ellos se acorta. En redes OBS con muchos saltos, el offset se puede hacer tan pequeño que sea imposible procesar el BCP antes de la llegada de la ráfaga y esta se pierde. Para evitar esta situación se pueden emplear esquemas como ODD [22], que usan líneas de retardo (FDL) en los nodos del core para mantener el offset constante. El simulador actualmente utiliza un modelo clásico sin FDLs, pero para soportar esto simplemente se deberían de incluir las FDLs en el ***OXC*** y que la unidad de control, al conmutar el ***OXC***, establezca también el tiempo a retrasar la ráfaga.

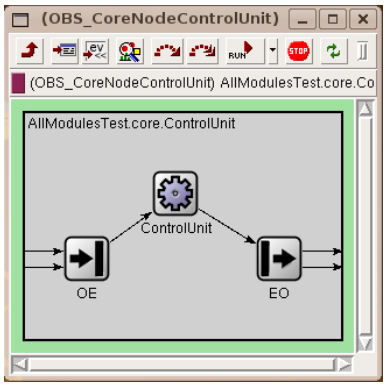

Figura 7. Unidad de control del Nodo del Core

V. Otros módulos

Por último es necesario mencionar que aunque con los módulos ***OBS_edgeNode*** y ***OBS_coreNode*** se puede realizar cualquier topología OBS, se han implementado otros dos componentes OBS para su uso en simulaciones: un sniffer óptico y una módulo de pérdidas.

1. ***OpticalMonitor***
   Es la implementación en OMNeT++ de un sniffer óptico. Su función es recoger estadísticas detalladas de las ráfagas que pasan por él, sin alterar de ninguna manera las ráfagas ni el funcionamiento de la red OBS.
2. ***DropBurst***
   Se usa para simular el efecto de que un segmento de red OBS pierde parte de las ráfagas que pasan por él. Actualmente sólo simula pérdidas independientes identicamente distribuidas, pero se puede modificar para que por ejemplo simule pérdidas que dependan del tamaño de las ráfagas.

VI. Simulaciones

Para demostrar el funcionamiento del simulador se ha realizado una red con la topología que se ve en la Fig. 8. Une cuatro nodos frontera usando dos nodos del core y cinco enlaces bidireccionales. Los enlaces son fibras ópticas con ocho longitudes de onda de datos por enlace y con 1Gbps de capacidad de transmisión por longitud de onda. Los nodos frontera están configurados para acumular tráfico durante 5ms (burstifier por timer) y tienen un buffer electrónico infinito.

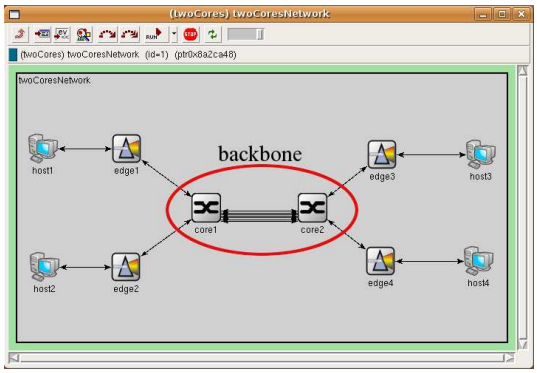

Figura 8. Red usada en las simulaciones

Hay dos flujos de tráfico UDP: tráfico de interés del *host1* al *host3* y tráfico interferente del *host2* al *host4*. En los dos casos el tiempo entre llegadas sigue una distribución exponencial y los tamaños de los paquetes son constantes. El tráfico de interés tiene un bitrate medio de 1.6Gbps, o lo que es lo mismo, el 20 % de la capacidad del enlace entre los dos nodos del core. Se ha hecho un barrido en el bitrate medio del tráfico interferente para que ocupe desde 2.5 % al 90 % de la capacidad del enlace entre los dos nodos del core. Como los nodos frontera tienen buffer electrónico infinito, las ráfagas sólo se pueden perder por contienda en el nodo del core *core1*.

En la figura 9 se puede ver la probabilidad de pérdida de ráfagas en la red OBS. Como era de esperar, se ve que no hay pérdidas por contienda en el nodo del core *core1* hasta valores medios de ρ del tráfico interferente. A partir de ese punto las pérdidas siguen aumentando.

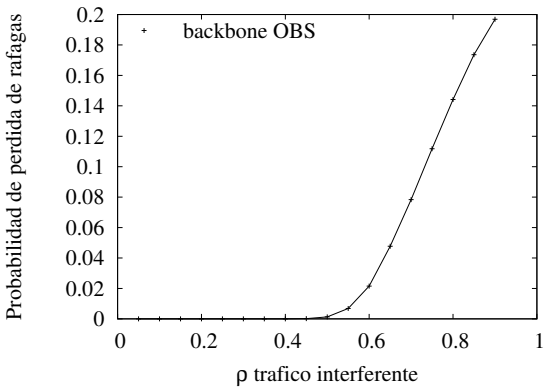

Figura 9. Probabilidad de pérdida de ráfagas

VII. Conclusiones

En este trabajo se ha presentado un modelo de simulación de OBS para el simulador de eventos discretos OMNeT++. Este modelo incluye tanto la implementación del nodo frontera como la del nodo del core teniendo siempre en mente la modularidad que permita incluir futuras propuestas que se hagan sobre esta tecnología.

Se ha visto que es fáctible la realización de un modelo de simulación de OBS para su uso en investigación. Aunque sea el primer desarrollo de este modelo, simula correctamente el funcionamiento básico de OBS.

En un futuro se prevee incluir mejoras. En los nodos frontera se pueden incluir nuevos esquemas de agregración y esquemas de planificación más avanzados que el **Horizon o LAUC**. En el nodo del core se puede plantear usar una *tabla de forwarding* dinámica o la inclusión de líneas de retardo (o FDLs) para darle cierta capacidad de buffering.

Agradecimientos

Este trabajo ha sido financiado por el MEC (proyecto STRRONG TEC2007-62192/TCM). Los autores quieren agradecer a la red temática IPoTN (TEC2008-02552-E/TEC).

Referencias

[1] C. Qiao and M. Yoo, "Optical Burst Switching (OBS) - A New Paradigm for an Optical Internet," *Journal of High-Speed Networks*, vol. 8, no. 1, 1999.
[2] J. Choi, J. Choi, and M. Kang, "Dimensioning Burst Assembly Process in Optical Burst Switching Networks," *IEICE Transactions on Communications*, vol. E88-B, pp. 3855–3863, October 2005.
[3] T. Battestilli and H. Perros, "An introduction to optical burst switching," *Communications Magazine, IEEE*, vol. 41, pp. S10–S15, Aug. 2003.
[4] A. Campi, W. Cerroni, F. Callegati, G. Zervas, R. Nejabati, and D. Simeonidou, "SIP Based OBS networks for Grid Computing," in *LNCS 4534, Proceedings of ONDM*, (Athens, Greece), May 2007.
[5] W. Zhang, J. Wu, J. Li, W. Minxue, and S. Jindan, "TCP Performance Experiment on LOBS Network Testbed," in *LNCS 4534, Proceedings of the 11th International IFIP TC6 Conference, ONDM* (I. Tomkos, F. Neri, J. Sole, X. Masip, and S. Sanchez, eds.), (Athens, Greece), pp. 186–193, May 2007.
[6] J. Kim, J. Cho, M. Jain, D. Gutierrez, L. Kazocsky, C. Su, R. Rabbat, and T. Hamada, "Demonstration of a 2.5 Gbps Optical Burst Switched WDM Ring Network," in *Proceedings of OFC/NFOEC*, March 2006.
[7] Y. Sun, T. Hashiguchi, V. Minh, X. Wang, H. Morikawa, and T. Aoyama, "A Burst-Switched Photonic Network Testbed: Its Architecture, Protocols and Experiments," *IEICE Transactions on Communicactions*, vol. E88-B, pp. 3864–3873, October 2005.
[8] Z. Gao, Y. Qiao, Y. Ji, and T. Saito, "The Optical Burst Switching Ring Network Using AOTF to Drop Data Burst," *Journal of Optical Communications*, vol. 26, no. 6, pp. 255–259, 2005.
[9] Y. Sun, T. Hashiguchi, V. Minh, X. Wang, H. Morikawa, and T. Aoyama, "Design and Implementation of an Optical Burst-Switched Network Testbed," *IEEE Communications Magazine*, vol. 43, pp. S48–S55, November 2005.
[10] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, (ICST, Brussels, Belgium, Belgium), pp. 1–10, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.
[11] A. Varga, "The OMNET++ discrete event simulation system," in *Proceedings of the European Simulation Multiconference*, (Prague, Czech Republic), pp. 319–324, SCS – European Publishing House, June 2001.
[12] A. L. Barradas and M. C. R. Medeiros, "An OMNeT++ model for the evaluation of OBS routing strategies," in *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, (ICST, Brussels, Belgium, Belgium), pp. 1–7, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.
[13] Optical Internet Research Center, "OIRC OBS-ns simulator," April 2008. http://wine.icu.ac.kr/~obsns/.
[14] J. P. Jue and V. M. Vokkarane, *Optical Burst Switched Networks*. Springer, 2005.
[15] X. Yu, Y. Chen, and C. Qiao, "A Study of Traffic Statistics of Assembled Burst Traffic in Optical Burst Switched Networks," in *Proceedings of SPIE Opticomm*, (Boston), pp. 149–159, SPIE, July 2002.
[16] C. Qiao, "Labeled optical burst switching for IP-over-WDM integration," *Communications Magazine, IEEE*, vol. 38, pp. 104–114, Sep 2000.
[17] J. S. Turner, "Terabit burst switching," *J. High Speed Netw.*, vol. 8, no. 1, pp. 3–16, 1999.
[18] J. Teng and G. N. Rouskas, "A comparison of the JIT, JET, and Horizon wavelength reservation schemes on a single OBS node," in *In Proc. of the First International Workshop on Optical Burst Switching*, p. 2003, 2003.
[19] A. Zalesky, E. Wong, M. Zukerman, H. L. Vu, and R. Tucker, "Performance analysis of an OBS edge router," *Photonics Technology Letters, IEEE*, vol. 16, pp. 695–697, Feb. 2004.
[20] J. Wei and J. McFarland, R.I., "Just-in-time signaling for WDM optical burst switching networks," *Lightwave Technology, Journal of*, vol. 18, pp. 2019–2037, Dec 2000.
[21] Y. Chen, C. Qiao, and X. Yu, "Optical burst switching: a new area in optical networking research," *Network, IEEE*, vol. 18, pp. 16–23, May-June 2004.

[22] L. Xu, H. G. Perros, and G. N. Rouskas, “A simulation study of optical burst switching and access protocols for wdm ring networks,” *Comput. Netw.*, vol. 41, no. 2, pp. 143–160, 2003.