UNIVERSIDAD DE VALLADOLID

Dpto. de Teoría de la Señal y Comunicaciones e Ingeniería Telemática
Escuela Técnica Superior de Ingenieros de Telecomunicación

Práctica 2
Estudio e implementación de una red con encaminamiento por longitud de onda
WRON

Proyecto de Innovación Docente: Aprendizaje Basado en Proyectos mediante un simulador de Redes WRON en Asignaturas de Redes Ópticas
PID 126

Curso 2014/2015

1. Introducción y objetivos

Escenario

Acabáis de ser contratados por Optelca I+D, una empresa (ficticia) dedicada a investigación y desarrollo en el área de telecomunicaciones por cable. Vuestra primera tarea es integraros en el proyecto europeo COMMON (*COntrol and Management of Multiwavelength Optical Networks*). El proyecto, financiado dentro del programa Horizon 2020, va a ser llevado a cabo por un conjunto de universidades, operadores y fabricantes de equipos de telecomunicaciones de distintos países de Europa. El objetivo de dicho proyecto es estudiar distintos sistemas de control y gestión de redes ópticas y a continuación implementar una red de pruebas (*testbed*) utilizando los equipos de dos operadores nórdicos integrados en el proyecto.

El proyecto se ha dividido en varios paquetes de trabajo, y Optelca I+D está encargada de desarrollar los métodos control en un entorno centralizado que determinarán el comportamiento de estas redes para implementarlo posteriormente en el *testbed*. Dentro de este objetivo hay tres tareas a desarrollar:

- Diseñar una topología virtual a partir de una matriz de tráfico utilizando métodos propuestos en la literatura.
- Determinar el algoritmo de encaminamiento y asignación de longitud de onda dinámico que para una misma carga de tráfico y capacidad de la red (longitudes de onda empleadas en cada enlace) ofrece la menor probabilidad de bloqueo.
- Diseñar métodos de supervivencia en estas redes basados en protección de caminos.

Se supone que todos los alumnos de la clase formáis el departamento de ingeniería de Optelca I+D. Por lo tanto, tendréis que trabajar como un equipo de decidir quien realiza cada una de las tareas. El resultado final es que todo tiene que funcionar de forma conjunta.

A lo largo de este enunciado se puede encontrar el planteamiento del problema, una descripción de la arquitectura y el simulador que se va a emplear en la práctica, las clases y métodos necesarios para implementar la simulación y una guía con los pasos que se deben llevar a cabo para completar la práctica. Así, a partir de este enunciado y junto a la ayuda que ofrece la propia herramienta y el material de las clases teóricas, el alumno deberá ser capaz de finalizar la práctica.

Objetivos de aprendizaje

Al finalizar la práctica el alumno deberá:

- Haber reforzado los conceptos de la parte teórica del tema 3 (Redes WRON).
- Ser capaz de implementar algoritmos de encaminamiento y asignación de longitudes de onda estudiados en clase.
- Conocer técnicas básicas de simulación de redes ópticas.
- Ser capaz de realizar una pequeña investigación y escribir un artículo describiendo sus resultados.
- Ser capaz de organizarse para realizar trabajo en un gran grupo.

Temporización de la prácticas

Para el desarrollo de esta práctica dispondréis de 10 horas de laboratorio con la organización mostrada anteriormente y la práctica se realizará en el laboratorio de docencia de la asignatura: 2L009. La duración de la prácticas se dividirá en tres semanas:

- Semana 1 (S1):
  - Sesión 12/05/2015: Clase de simulación, presentación de la práctica e introducción al simulador.
- Semana 2 (S2):
  - Sesión 19/05/2015: Desarrollo del simulador.
  - Sesión 21/05/2015: Desarrollo del simulador.
- Semana 3 (S3):
  - Sesión 26/05/2015: Desarrollo del simulador.
  - Sesión 28/05/2015: Desarrollo del simulador y sesión de evaluación.

División del trabajo

El proyecto tendrá que tener un coordinador elegido democráticamente entre vosotros. El coordinador será el responsable de que todo el trabajo se realice correctamente y de coordinar los informes de seguimiento que se entregarán al organismo subvencionador (profesor en este caso). Además, podéis dividiros en los grupos de trabajo que consideréis necesarios para desarrollar correctamente este proyecto. Cada grupo de trabajo estará dirigido por un líder de grupo que también decidiréis vosotros.

Entregas y evaluación

Al finalizar cada semana, el coordinador del equipo tendrá que enviar al profesor un informe de evolución del proyecto, el grado de consecución de los objetivos del mismo y los problemas acontecidos durante cada periodo. Este informe no debe exceder las 10 páginas y al mismo contribuirán todos los alumnos aunque sea labor del coordinador la coordinación del mismo. En cada informe se indicará quien es(son) el(los) responsable(s) de redacción de cada apartado.

En el entregable de la primera semana tiene que aparecer: los grupos de trabajo que se han formado, la división de tareas, la temporización prevista para cada una de las tareas y un plan de contingencia que indique como actuar en el caso que surjan imprevistos.

Además, al finalizar la prácticas, se realizará una sesión de evaluación con cada uno de los grupos en los que os habéis dividido para evaluar el trabajo realizado. En estas sesión, se realizará una evaluación personal a cada uno de los alumnos. Se valorará que el programa funcione correctamente (sin errores y que los algoritmos se hayan codificado correctamente). También es importante que el código esté suficientemente comentado y se valorará (levemente) el estilo de programación.

Además, el coordinador del proyecto podrá obtener 2 puntos (sobre 10) a mayores del máximo que se puede obtener en la práctica dependiendo de cómo realice sus funciones de coordinación.

Los líderes de los grupos de trabajo podrán obtener 1 punto (sobre 10) a mayores del máximo que se puede obtener en la práctica dependiendo de cómo realice sus funciones.

2. Introducción a OMNeT++

OMNeT++ [1] es un simulador de eventos discretos que permite diseñar, evaluar y optimizar el funcionamiento de todos aquellos sistemas susceptibles de ser implementados como una simulación de eventos discretos. Entre estos sistemas se encuentran los métodos de control de redes ópticas como las estudiadas en esta asignatura, tanto de transporte como de acceso. OMNeT++ tiene la ventaja de tratarse de un software de libre distribución y de código siempre que se utilice por universidades para trabajar en investigación o en docencia.

OMNeT++ es un entorno de simulación de eventos discretos, de código abierto, basado en componentes, modular y de arquitectura abierta que cuenta, además, con un potente soporte para la interfaz gráfica de usuario. OMNeT++ está disponible para distintas plataformas y sistemas operativos. En esta práctica se utilizará la versión desarrollada para LINUX.

El principal ámbito de aplicación de dicho simulador son las redes de comunicaciones pero, debido a su arquitectura genérica y flexible, se ha utilizado con mucho acierto en otras áreas de simulación de sistemas industriales, redes de colas, arquitecturas hardware y procesos de negocio. De hecho, actualmente existe una distribución del mismo en versión comercial denominada OMNEST [2] desarrollada por la empresa Simulcraft, Inc. Además de estos dos simuladores, existen otros que también pueden ser empleados para simular el comportamiento de redes ópticas como son NS2 [3] y OPNET Modeler [4]. El primero es de libre distribución y más orientado al estudio de redes IP, mientras que el otro es comercial y tiene una gran cantidad de módulos actualmente desarrollados.

OMNeT++ proporciona una arquitectura de componentes por modelos. Los componentes (denominados módulos en OMNeT++) son programados en el lenguaje C++. A continuación, son ensamblados en componentes y modelos más grandes utilizando un lenguaje de alto nivel (NED). En realidad, OMNeT++ no es una herramienta de simulación de redes en sí misma, pero está obteniendo cada vez más popularidad como plataforma de simulación de redes.

La plataforma OMNeT++ incluye una biblioteca de simulación del kernel, un compilador para el lenguaje propio de descripción de topología que se denomina NED, dos interfaces de usuario (Tkenv y Cmdenv), herramientas para la documentación del modelo, herramientas para mostrar resultados (Plove para salidas vectoriales y Scalars para salidas escalares), ejemplos y una documentación bastante completa.(Ver referencia [5]).

2.1 Interfaces de simulación

Las interfaces de usuario de OMNeT++ son utilizadas en los procesos de simulación para hacer visible al usuario el interior de la simulación, permitir detener y reanudar las simulaciones, y permitir al usuario intervenir para modificar variables dentro del modelo. Esto es muy importante en la fase de desarrollo y depuración (debugging) del simulador. Las interfaces de usuario son bibliotecas independientes que conectan el kernel de simulación y los modelos a través de interfaces bien definidas. Esto significa que cualquier modelo puede ser ejecutado bajo cualquier interfaz de usuario. Actualmente, existen dos interfaces de usuario para OMNeT++:

Tkenv: interfaz gráfico de usuario de ventanas basado en Tk. Soporta la ejecución interactiva de simulaciones, el seguimiento de trazas y herramientas de depuración. Por tanto, esta interfaz se recomienda para las fases de desarrollo del simulador o para presentaciones, puesto que permite observar una visión general del estado de la simulación en cualquier punto de la ejecución de la misma y permite a su vez seguir qué ocurre en el interior de la red. Cuando se utiliza de forma simultánea con un debugger de C++, proporciona un entorno excelente para el testeo y la depuración. La posibilidad de mostrar los resultados de la simulación durante la ejecución puede acelerar el proceso de verificar el correcto funcionamiento del programa de simulación y proporciona un entorno adecuado para la experimentación sobre el modelo. Este interfaz es compatible con cualquier plataforma que soporte Tcl/Tk.

Cmdenv: interfaz de usuario de línea de comandos para ejecución por lotes. Es pequeño, portable y rápido. Se compila y ejecuta en cualquier plataforma sea Unix o Windows. Cmdenv está diseñado principalmente para la ejecución por lotes. Cmdenv simplemente ejecuta todas las simulaciones que estén descritas en el archivo de configuración correspondiente.

2.2 Estructura de los modelos

En OMNeT++, un modelo consta de módulos que se comunican con mensajes entre sí. Los módulos activos, aquellos que realizan las instrucciones de la simulación, se denominan módulos simples (*simple modules*) y están escritos en C++ utilizando la biblioteca de clases de simulación. Los módulos simples se pueden agrupar en módulos compuestos (*compound modules*) y así sucesivamente; el número de niveles de jerarquía no está limitado. Por otra parte, los mensajes se pueden enviar o bien a través de conexiones que unen los distintos módulos, o bien directamente a los módulos destino. Cuando se envían mensajes del modo habitual (no directamente), éstos se

transmiten a través de puertas. Las puertas son interfaces de entrada y de salida de los módulos. Una puerta de salida y una de entrada se pueden unir por conexiones. Las conexiones se crean dentro de un mismo nivel de la jerarquía y también pueden conectar una puerta de un submódulo y una puerta del módulo compuesto al que pertenece.

Debido a la estructura jerárquica del modelo, los mensajes típicamente viajan a través de cadenas de conexiones, transmitiéndose desde y hasta módulos simples. Por tanto, los módulos compuestos son prácticamente cajas negras para el mundo exterior. A las conexiones se les puede asignar propiedades como el retardo de propagación, la tasa binaria, la tasa de error de bit. Los módulos pueden tener parámetros. Los parámetros se utilizan principalmente para pasar datos de configuración a los módulos más simples, y para ayudar a definir la topología del modelo. En OMNeT++ cada módulo, enlace y puerta tiene un identificador (ID) que es único para toda la simulación.

El usuario define la estructura del modelo (los módulos y sus interconexiones) a través del lenguaje de descripción de la topología NED. Típicamente, la descripción NED consta de declaración de módulos simples, definición de módulos compuestos y definición de la red. Las declaraciones de módulos simples describen la interfaz del módulo: puertas y parámetros. Las definiciones de módulos compuestos constan de la declaración de los interfaces externos de los módulos, y la definición de los submódulos y sus interconexiones. Una definición de la red básicamente define el conjunto de módulos que conforman la red y sus parámetros.

En OMNeT++ existen posibilidades de respuesta a los eventos. Estas dos posibilidades se traducen en dos métodos en los módulos simples sin que puedan emplearse al mismo tiempo. En primer lugar está *activity()* en el que habría que preparar al módulo para estar continuamente escuchando y se realizarían bucles infinitos para ello. Por otro lado está *handlemessage()* que en un método en el que se entra siempre que le llegue cualquier paquete al módulo. De esta forma, se pueden recibir paquetes de otros módulos enviado por alguno de los enlaces (método *send()*) o directamente (método *sendDirect()*), pero también, se pueden programar el envío de paquetes a si mismo en el momento que el usuario lo desee (método *scheduleAt()*). De esta forma, no habría falta tener bucles infinitos. Esta segunda es la forma más eficiente y la que se utilizará en esta práctica.

3. Diseño de una red WRON

Para la realización de la práctica, se van a emplear dos topología física (una topología de seis nodos y NSFNet). La primera de ellas se utilizará para realizar pruebas y testear los métodos mientras que la segunda se empleará para sacar los resultados de los artículos. Se utilizará un esquema de control centralizado para lo que se ha incluido dentro de la topología física un módulo de control que será donde esté almacenado la base de datos con el estado de la red o TED[1] (en este caso simplificada por un conjunto de tablas) el estado de la red y donde se resuelva el problema RWA. Un ejemplo de la arquitectura puede verse en la Figura 3.1. Toda la arquitectura general y de nodo de control va a estar previamente desarrollada para que el alumno no tenga que estudiar el lenguaje propio del simulador, NED. En la práctica, tendréis que programar los módulos simples con las instrucciones que se ofrecen en este enunciado. Como se ha comentado anteriormente estos bloques están programados en C/C++.

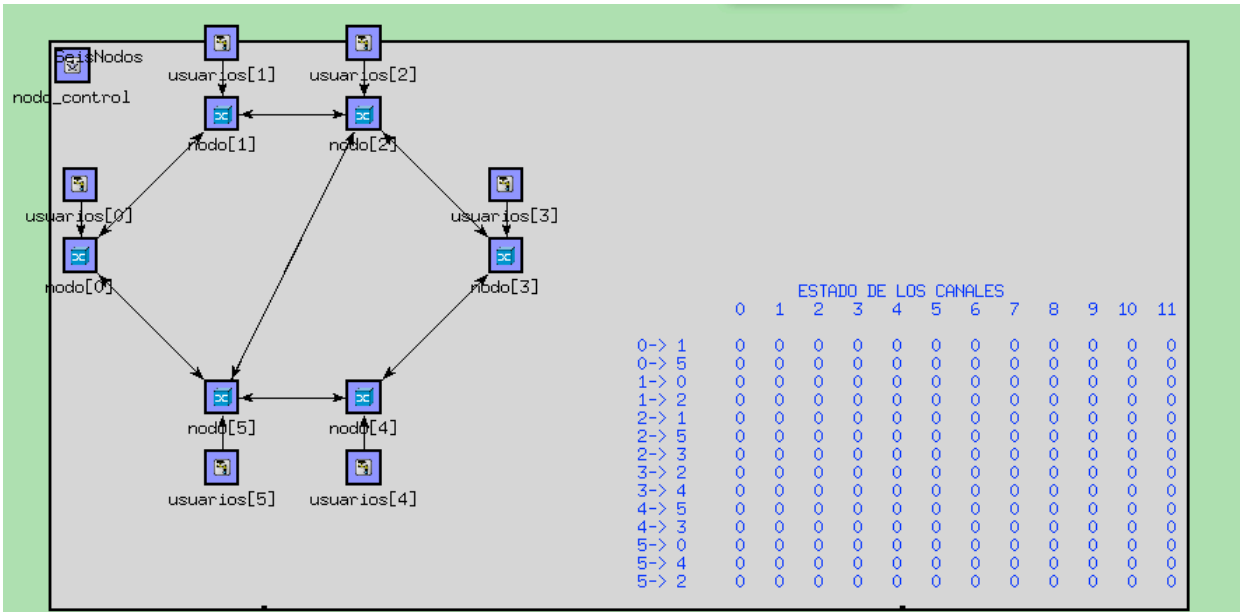

Figura 3.1. Topología física de la red de seis nodos y vista del simulador

Dentro del Nodo de Control podemos encontrar los siguientes componentes:

- **Estado**: Se trata de un módulo simple. Este módulo va a imitar la TED y contendrá la información sobre el estado (utilizado/libre) de los recursos de la red. Por ejemplo: habrá una tabla para indicarnos que longitudes de onda no están siendo utilizadas en una determinada fibra por ningún *lightpath* por lo que puede ser utilizado para establecer uno nuevo. En realidad a este módulo no van a llegar paquetes pero el resto

[1] Traffic Engineering Database

de módulos van a poder acceder a él para saber el estado de los recursos y actualizar los mismos. Además, dentro de este módulo esta implementado el cálculo de rutas entre un origen y un destino (desde los caminos más cortos a los más largos). Para acceder a la información de este módulo se han desarrollado un conjunto de métodos que serán los que tengáis que utilizar en la práctica. En concreto:

- *bool checkTxAvailability(int source)*: Devuelve la disponibilidad de transmisores libres en el nodo origen (*source*).
- *bool checkRxAvailability(int destination)*: Devuelve la disponibilidad de receptores libres en el nodo destino (*destination*).
- *bool checkChannelAvailability(int link, int wavelength)*: Devuelve la disponibilidad de la longitud de onda *wavelength* en la fibra *link*.
- *int numPath(int source, int destination)*: Devuelve el número de caminos desde el nodo origen (*source*) al nodo destino (*destination*).
- *int pathDistance(int source, int destination, int path)*: Devuelve la distancia (en número de saltos) del camino *path* desde el nodo origen (*source*) al nodo destino (*destination*).
- *int linkInPath(int source, int destination, int path, int hop)*: Devuelve la fibra que se emplea en el salto *hop* del camino *path* desde el nodo origen (*source*) al nodo destino (*destination*).

- **Nodo_Control_Dynamic_WRON** (que se encuentra dentro del módulo **Dynamic_WRON)** Nodo compuesto que contendrá todo el funcionamiento de la red WRON dinámica.

  1. Resolverá el problema RWA para cada petición y comprobará que hay al menos un transmisor libre en el nodo origen y un receptor libre en el nodo destino. Este es el trabajo que tenéis que realizar vosotros implementando distintos métodos para resolver el problema RWA.
  2. Si estuviese activada la opción de protección, búsqueda de una ruta y una longitud de onda para reservar los recursos que ofrezcan protección de camino.
  3. Si hubiese recursos suficiente se establecerá la conexión y se actualizará la TED para indicar que conjunto canales utilizado por el *lightpath* está ocupado y no podrá ser utilizado para establecer otra conexión óptica.
  4. Se liberarán los recursos de la conexión una vez que haya vencido el tiempo de

establecimiento de la misma.

- **Nodo_Control_Static_WRON** (que se encuentra dentro del módulo **Static_WRON)** Nodo compuesto que contendrá todo el funcionamiento de la red WRON estática.
  1. Generará las matrices de tráfico entre cada par de nodos simulando el funcionamiento de un sistema monitorización de tráfico.
  2. Diseñará la topología virtual atendiendo a la matriz de tráfico solicitada y al conjunto de recursos de la red.

3.1 WRON dinámica

La red empleará un sistema de control centralizado, en la que el nodo de control recibe peticiones de establecimiento de conexión de *lightpaths* unidireccionales y las cursa teniendo en cuenta la disponibilidad de recursos en la red (transmisores/receptores y longitudes de onda) —si bien no se simulará el envío de los mensajes de control por la red—. Se deberán evaluar los algoritmos determinando en función de su probabilidad de bloqueo para distintas cargas y su comportamiento.

Como se observa en la Figura 3.1 la arquitectura propuesta consta de nodos y usuarios que realizan dinámicamente peticiones para establecer *lightpaths*. Cada usuario generará peticiones conforme a una carga (*load*) determinada por el diseñador (y que en su día será obtenida de los monitores de tráfico de la red). Considerando que *N* es el número de nodos en la red, el tiempo entre peticiones se ha modelado como una distribución exponencial de media:

$$\overline{tiempo\ entre\ peticiones} = \frac{\overline{duración\ de\ un\ lightpath}}{N \cdot (N-1) \cdot load}$$

El nodo origen y el nodo destino de cada *lightpath* se elegirá según una distribución uniforme y para determinar la conexión de un *lightpath* se utilizará una distribución exponencial de media *duración de un lightpath*. Además, se permitirá el establecimiento de más de un *lightpath* entre cada par origen-destino (siempre que haya recursos disponibles).

La generación de números aleatorios es otro de los elementos más importantes para la simulación de eventos. Para está práctica vamos a utilizar el que trae OMNeT++ 4.2.2 por defecto dado que ofrece muy buenas propiedades. En concreto se trata del Mersenne Twister de M. Matsumoto y T. Nishimura que tiene un periodo de 2^{19937}-1.

En el interior del módulo Nodo_Control_Dynamic_WRON se han creado tres bloques:

- Control RWA Dinámico: Módulo simple. Funciones:
  - Recibe las peticiones.
  - Se envía la petición al módulo algoritmo RWA (explicado debajo) para que resuelva el problema RWA.
  - Cuando reciba la respuesta del módulo RWA:
    i. Si no hay recursos suficientes, el lightpath no se establece y, si la opción de protección está activada, reserva los recursos para protección.
    ii. Si los hay:
      1. Actualiza la tabla de estados para ocupar las longitudes de onda oportunas en cada enlace.
      2. Establece un mecanismo para liberar la conexión.
  - Libera las conexiones después del tiempo que se determine.
- Método RWA dinámico: Resuelve el problema de encaminamiento y asignación por longitudes de onda. En esta práctica tendréis que implementar: **Fixed-Routing (Shortest Path)** para el encaminamiento y **First-Fit, Random, Most Used** y **Least Used** para la asignación de longitudes de onda. Supondremos que la red no dispone de convertidores en longitud de onda por lo que hay que asegurar la continuidad de longitud de onda a lo largo del camino.
- Método Protección: Resuelve el problema de búsqueda de rutas y longitud de onda para la reserva de recursos para establecer protección de camino al lightpath solicitado. Tendréis que implementar **protección de camino** a los lightpaths dinámicos que se establezcan **utilizando compartición de recursos y sin utilizar compartición de recursos**.

3.2 WRON estática

El diseño de la topología virtual también se realizará en el nodo de control. Este diseño sólo se realizará una única vez al inicio de la simulación. Para ello, se ha implementado un módulo que enviará la matriz de tráfico al módulo encargado del diseño de dicha topología virtual. El primero de estos módulos simula un sistema de monitorización de tráfico que mide continuamente el tráfico entre los nodos de la red y envía esta estadística al nodo de control para que diseñe y establezca la topología virtual que mejor se adapta a dicha demanda de tráfico. En nuestras

simulaciones, cada valor de la matriz se va a generar de forma aleatoria y supondremos que la capacidad de cada lightpath es de 10 Gbps.

El módulo que tenéis que implementar es aquel que **diseña la topología virtual**. Para ello, vais a utilizar el **algoritmo HLDA** [6, pp. 847] que es uno de los métodos clásicos para la resolución de dicho problema.

4. Agradecimientos

Natalia Fernández Sordo colaboró en la elaboración de esta práctica.

5. Bibliografía

[1] OMNeT++ http://www.omnetpp.org
[2] OMNEST http://www.omnest.org
[3] NS-2 http://www.isi.edu/nsnam/ns/
[4] OPNET Modeler http://www.opnet.com/solutions/network_rd/modeler.html
[5] Manual de OMNeT++
[6] R. Ramaswami, K.N. Sivarajan, "Design of logical topologies for Wavelength-Routed Optical Network," IEEE Journal on Selected Areas in Communications, vol. 14. No. 5, June 1996.