UNIVERSIDAD DE VALLADOLID

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

Design of methods to solve the RWA problem in Wavelength-Routed Optical Networks

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. Introduction

In this lab exercise you are supposed to be employed by Optelca I+D, a virtual company in the field of research and development in telecommunication technologies. Your first task consists in working in the European project COMMON (*COntrol and Management of Multiwavelength Optical Networks*). That project has many partners from the industry and from universities. The goal of COMMON is to analyse different control and management systems for optical networks. In the last part of the project, a real testbed will be developed in the locations of one of the operators that belong to the project consortium. The project was divided into different work-packages and your company, Optelca I+D, is in charge of determining the best method to establish dynamic lightpaths (i.e., all-optical connections between two nodes of the network even when they are not adjacent in the physical topology) in the network. Then, that method will be integrated in the control node of the final testbed. Your objective in this lab exercise is to determine which dynamic routing and wavelength assignment algorithm (DRWA) provides the lowest blocking probability for the incoming lightpath requests depending on the number of wavelengths in each fibre and the network load. The main results of this research work should be disseminated in a paper that will be sent to Virtual Conference on Optical Networks (VICON 2015)

In this document you can find the problem statement, the description of the simulator that will be used in the network and the classes and methods that you will require in order to implement the simulation.

Objectives of this lab exercise:

At the end of the exercise, the student should:

- Understand the theoretical concepts associated to Wavelength-Routed Optical Networks. In particular, the establishment of dynamic optical circuits or lightpaths.
- Be able to implement the routing and wavelength assignment (RWA) algorithms studied in theoretical classes.
- Know the basic techniques to simulate optical networks.
- Be able to work in a relatively simple research project and write a scientific paper in order to present the main results from that research.

2. Simulation environment: OMNeT++

As described in [1], "*OMNeT++ is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators. "Network" is meant in a broader sense that includes wired and wireless communication networks, on-chip networks, queueing networks, and so on. Domain-specific functionality such as support for sensor networks, wireless ad-hoc networks, Internet protocols, performance modeling, photonic networks, etc., is provided by model frameworks, developed as independent projects. OMNeT++ offers an Eclipse-based IDE, a graphical runtime environment, and a host of other tools. There are extensions for real-time simulation, network emulation, database integration, SystemC integration, and several other functions.*

Although OMNeT++ is not a network simulator itself, it is currently gaining widespread popularity as a network simulation platform in the scientific community as well as in industrial settings, and building up a large user community.

OMNeT++ provides a component architecture for models. Modules can be assembled into larger components and models using a high-level language (NED). Reusability of models comes for free. OMNeT++ has extensive GUI support, and due to its modular architecture, the simulation kernel (and models) can be embedded easily into your applications.

Modules can be connected with each other via gates (other systems would call them ports), and combined to form compound modules. The depth of module nesting is not limited. Modules communicate through message passing, where messages may carry arbitrary data structures. Modules can pass messages along predefined paths via gates and connections, or directly to their destination; the latter is useful for wireless simulations, for example. Modules may have parameters that can be used to customize module behavior and/or to parameterize the model's topology. Modules at the lowest level of the module hierarchy are called simple modules, and they encapsulate model behavior. Simple modules are programmed in C++, and make use of the simulation library."

"*OMNeT++ simulations can be run under various user interfaces. Graphical (tkenv), animating user interfaces are highly useful for demonstration and debugging purposes, and command-line user interfaces (cmdenv) are best for batch execution.*" [1]

3. Design of a dynamic WRON

Two different physical topologies will be analyzed: the 6-nodes topology and the NSFNet. The first of them will be used to check whether the methods are properly implemented in the simulator or not, as it is enough small to analyse step-by-step some use cases that can validate that everything works fine. Then, the results that will be presented in the paper will be obtained using the NSFNet as a physical network. In order to make the exercise easier, a centralized control method will be used. Therefore, a control node will be included in the topology and it will be in charge of solving the DRWA problem for each lightpath request that arrives at the network. Moreover, this control module will keep the TED[1] updated with the availability of each network resource. An example of that architecture can be seen in Fig. 1. The network architecture and the control node has already been developed, so there is no need for you to learn NED (the OMNeT++ definition language). Your task will be to program simple modules using the methods that are presented in this statement using C/C++ programming language.

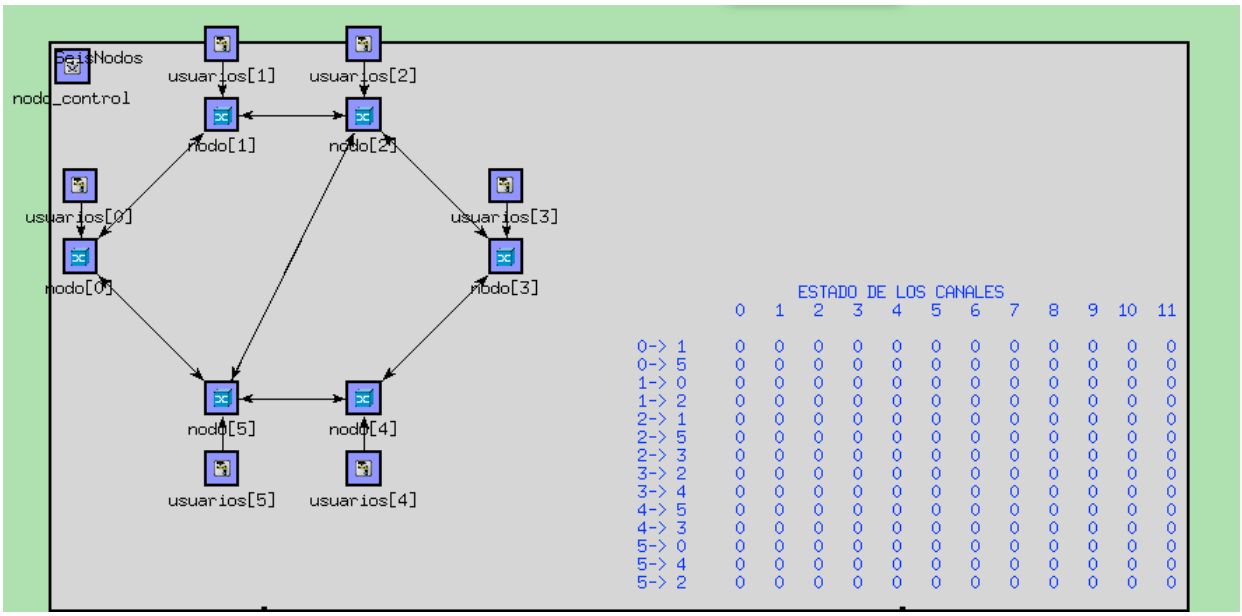

Fig 3.1. Six-nodes network topology and a simulator view

As can be seen from Fig. 3.1, the proposed architecture has different nodes and users who dynamically request the establishment of new lightpaths. Each user requests lightpaths according to a network load set by the network designer. In real network, the network load would be obtained with traffic monitors. Considering that *N* is the number of nodes in the network, the average time between requests can be modelled as an exponential distribution with mean:

[1] Traffic Engineering Database

$$\overline{time\ between\ requests} = \frac{\overline{lightpath\_duration}}{N \cdot (N-1) \cdot load}$$

The source and the destination node of each lightpath is chosen according to a uniform distribution and an exponential distribution is also used to determine the duration of each lightpath. It is important to remark that the establishment of more than one lightpath between each pair of nodes is allowed (if there are enough available resources).

Random number generation is a very important issue when using event discrete simulations. In this lab exercise, the default random number generator from OMNeT++ will be used as it shows very good properties. It is called Mersenne-Twister and was proposed by M. Matsumoto y T. Nishimura. It has a length of 2^{19937}-1.

Within the Control Node (Fig. 3.2), the next components are included:

- **State**: It is a simple OMNeT++ module. It emulates the behaviour of the TED and it contains the information about the availability of network resources. E.g., it contains a table which stores whether a certain wavelength is used in a certain fibre, or if it is idle and thus can be assigned for a new lightpath. Even when this module is not directly connected with the rest of the modules in the control node, the rest of the simple modules in control node can read and write information in its tables. Moreover, this module also implements the algorithm for finding the k-shortest paths between each pair of nodes. Paths are ordered from the shortest to the longest ones. In order to retrieve information from the TED a set of methods have been developed. You will need to use those methods to complete the lab exercise. The methods are:
    - *bool checkTxAvailability(int source)*: It returns the availability (or not) of a transmitter in the *source* node.
    - *bool checkRxAvailability(int destination)*: It returns the availability (or not) of a receiver in the *destination* node.
    - *bool checkChannelAvailability(int link, int wavelength)*: It returns the availability or not of a certain *wavelength* in a certain *link*.
    - *int numPath(int source, int destination)*: It returns the number of possible paths from the *source* node to the *destination* node.
    - *int pathDistance(int source, int destination, int path)*: It returns the distance (measured in number of hops) of a certain *path* from the *source* node to the *destination* node.

- *int linkInPath(int source, int destination, int path, int hop)*: It returns the fibre that is used in a certain *hop* of the path from the *source* to the *destination* node.

Figura 3.2. Control Node

- **Control_Node_Dynamic_WRON** (it belongs to **Dynamic_WRON** module). This compound module contains all the behaviour of the dynamic WRON and it is the module in which you have to work during this lab exercise. This module is in charge of the following actions
  1. It solves the RWA problem for any lightpath incoming request and it also checks that there is, at least, one idle transmitter in the source node and one idle receiver in the destination node. The objective of this lab exercise is to develop this module by the implementation of the RWA algorithms that were presented in the theoretical classes.
  2. If there are enough idle resources to establish the lightpath, it will establish that optical circuit and will update the TED
  3. Once the lightpath duration has been reached, the resources are released and the TED is updated.

Within the **Control_Node_Dynamic_WRON** two modules have been included:

- **Dynamic_RWA_Control**. It is a simple module with the following functions:
  - Receive the lightpath establishment requests
  - The request is sent to the module in charge of executing the RWA algorithm
  - When it receives the response from the module that solves the RWA problem with the set of resources required to establish the lightpath, it performs the following operations:
    i. When there are not enough resources to establish the lightpath,

the lightpath is not established.

ii. If there are enough resources to establish the lightpath:

1. The lightpath is established.
2. TED is updated.

- Release the lightpaths when they reach their maximum duration.

- **Dynamic_RWA_method**: It solves the routing and wavelength assignment problem. In this lab exercise, you have to implement the following algorithms: **fixed routing (shortest path)** for routing, and four different methods to solve the wavelength assignment: **First-Fit, Random, Most Used** y **Least Used**. In this lab exercise it is assumed that there are no wavelength converters in the network and, therefore, you have to ensure that each lightpath use the same wavelength from the source to the destination node.
- Algorithms should be compared (in terms of blocking probability) depending on both the traffic load the number of wavelengths per fibre.

4. Acknowledgment

Natalia Fernández Sordo has collaborated in the development of this lab exercise.

5. References

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