Framework For Performance Analysis of Optical Circuit Switched Network Planning Algorithms

Master Thesis by
Sonia Mayor Alonso

École Polytechnique Fédérale de Lausanne (EPFL)
and
Universitat Politècnica de Catalunya (UPC)

EPFL Assistant:
Sébastien Rumley

EPFL Supervisor:
Dr. Christian Gaumier

UPC Supervisors:
Dr. Josep Solé Pareta,
Dr. Davide Careglio
and Oscar Pedrola Escribà

June 2010

The most important thing in communication is hearing what isn't being said.
Author Unknown

Acknowledgements

To Sébastien
for his invaluable help and guidance all along this project and his always good mood

To Christian
for having hosted me in the TCOM laboratory and make this possible

To Oscar, Josep and Davide
for having had confidence in me and having supported me

To Ester, Patri, Maite, Cristina, Pati and Laura
for having made those months unforgettable, thank you my Lausanne friends!

To Matthieu, Jérôme, Émilie, Basile and Sidonie
for having give me the opportunity to live with them taking care of me as my family

To my family
for having supported me at any time

And to Esteban, Hugo, Laura, Eliana, Mariam, Carlos and Cristina
for having visited me in this magnific country!

Resumen

Hoy en día Internet es uno de los medios de comunicación más utilizados y debido a todas las aplicaciones de alta velocidad y servicios que ofrece, la necesidad de ancho de banda para la transmisión de información entre usuarios crece sin cesar. La tecnología propuesta para gestionar esta imparable necesidad son las fibras ópticas capaces de proporcionar grandes anchos de banda. Esta habilidad se da gracias a su capacidad de soportar varios canales de luz (o longitudes de onda) multiplexados en una sola fibra, es la llamada multiplexación por división de longitudes de onda (WDM). Cada longitud de onda tiene una cierta capacidad para el transporte de información entre usuarios. Además, para que una red de fibra óptica funcione, debe establecerse una cierta política de transporte de información para que el tráfico pueda circular convenientemente. Para ello existen 3 tecnologías básicas: Optical Burst Switching (OBS), Optical Packet Switching (OPS) y Optical Circuit Switching (OCS).

En este trabajo nos hemos centrado en el estudio de las redes que usan OCS. Esta tecnología orientada a conexión hace que, cuando se genera una demanda desde un nodo origen hasta un nodo destino, se establece un circuito siguiendo un camino concreto y usando unos recursos determinados (ciertas longitudes de onda dentro de las fibras de cada conexión entre nodos vecinos, o links). Este circuito establecido es lo que llamamos un lightpath. El mismo circuito se mantiene a lo largo de la duración de la conexión. Cuando ésta finaliza, los recursos usados se liberan para poder ser utilizados posteriormente por otras conexiones.

Nuestro objetivo final a través de este trabajo es el estudio del comportamiento y respuesta de las redes ópticas que usan Optical Circuit Swiching (OCS). Para ello hemos estudiado qué es exactamente la tecnología OCS, qué elementos participan pasiva o activamente en el funcionamiento de estas redes y hemos realizado un framework, es decir, hemos implementado toda una serie de herramientas programando en Java y sobre una base ya existente llamada Javanco, para la simulación de dichas redes bajo un abanico de situaciones distintas. Las simulaciones nos darán los resultados y valores necesarios para el posterior estudio y extracción de conclusiones sobre en qué grado las combinaciones de elementos estudiados favorecen la baja probabilidad de bloqueo, aprovechan de la mejor manera los recursos disponibles o resultan más económicas energéticamente hablando. Lo que queremos estudiar, en resumen, es el rendimiento de las redes ópticas que usan OCS. Después de un estudio teórico de los factores que participan en las redes OCS llegamos a la conclusión de que se pueden distinguir 3 grandes grupos: los inputs (o entradas al sistema), los modelos y configuraciones y los algoritmos que hacen que el sistema funcione. Y sabiendo concretamente qué elementos participan en cada unos de los grupos hemos llevado a cabo la implementación de las siguientes partes. En lo que respecta a los inputs hemos considerado

las diferentes topologías que puede tener una red, creando además una herramienta específica para el análisis de éstas, y a su vez hemos considerado las matrices de tráfico que estipulan a priori la cantidad de tráfico que circulará en nuestra red OCS. Los modelos y configuraciones implementados en este trabajo forman un amalgama con los algoritmos que los harán funcionar. Así, los generadores de tráfico que podrán ser de tipo incremental o dinámico y que son un modelo para la red, serán caracterizados por un algoritmo de ordenación (para el tráfico incremental) o de generación (para el dinámico) de demandas. Otro de los aspectos que definirá una configuración de red es si el sistema conservará la misma longitud de onda en todo el lightpath, la llamada wavelength continuity constraint, o si, por el contrario, la longitud de onda podrá variarse en alguno o todos los nodos. En nuestra herramienta hemos contemplado las dos opciones. Para las dos situaciones es necesario un algoritmo de enrutamiento que defina qué camino ha de seguir el lightpath. Los algoritmos de enrutamiento programados son: el que sigue el camino más corto (shortest path) o el que tiene en cuenta una utilización umbral de las fibras del link (occupancy depending). A continuación, si tenemos wavelength continuity constraint deberemos de decidir qué longitud de onda concreta se usará en el camino determinado anteriormente. Los algoritmos de asignación de longitud de onda tratado son: el que coge la primera longitud de onda libre (First Fit), el que escoge la menos utilizada disponible (Least Used) y la que elige la más utilizada disponible (Most Used). En este caso deberá también de configurarse la red con fibras y longitudes de onda numeradas en cada link. En el caso de que no tengamos en cuenta la restricción de las longitudes de onda, bastará con dar a cada link una capacidad que irá siendo utilizada a medida que el sistema evolucione.

Todos estos elementos forman ya un amplio abanico de posibilidades que, gracias a la estructura modular que le hemos dado al programa pueden ser combinados y modificados dependiendo del interés concreto del estudio e incluso se podrían añadir nuevos inputs, configuraciones y/o algoritmos de manera muy sencilla. Este conjunto viene acompañado de un sistema de filtros que recopila la información de las simulaciones a la vez que la registra y la representa gráficamente para que los resultados queden reflejados y puedan ser estudiados.

Hemos confeccionado también una herramienta para el estudio del consumo energético de las redes que usan OCS en función de las topologías virtuales que usemos para estudiarlas puesto que éstas determinarán cuántos componentes físicos deben utilizarse en cada caso.

Con todo esto tenemos ya las herramientas necesarias para estudiar cuáles son los elementos más favorables para el rendimiento de las redes OCS. Los gráficos obtenidos nos mostrarán la probabilidad de bloqueo, el tráfico ofrecido en cada caso y el consumo energético que pueden ocasionar algunas circunstancias. Lo que traducido en

la vida real (puesto que es lo que nos interesa realmente) equivaldría a juzgar en qué grado las demandas de los usuarios serían aceptadas, en qué cantidad máxima podrían enviar los usuarios sus demandas y cuánto les costaría eso energéticamente. El resultado de estos datos incumbe tanto a las operadoras que pretenden dar el mejor servicio como a los usuarios que buscan la plena satisfacción en el funcionamiento de su red.
Todos los módulos del programa han sido validados por estudios teóricos que nos ayudan a certificar si nuestras implementaciones son lo suficientemente cercanas a la realidad.

Tras varias simulaciones y pruebas, nos hemos dado cuenta de que no hay una solución absoluta ni un algoritmo perfecto que dé el mejor de los resultados para todas las situaciones. Por lo tanto, el rendimiento de una red depende totalmente de los elementos que participen en ella y no se puede predecir con toda seguridad cuál será la mejor de las propuestas para garantizar un rendimiento máximo. Un estudio completo debería de realizarse una vez sabidos los componentes o elementos que tendríamos de base para poder llegar a una conclusión más concreta.

Así pues, de cara al futuro, sería interesante plantear casos de estudio claros sobre redes OCS y por supuesto continuar el desarrollo del software aquí realizado para completar otros inputs, modelos, configuraciones o algoritmos posibles.

Contents

List of Figures

Chapter 1

Introduction

The Latin root of the word "communication" is "communicare" and means to impart, share or make common. Communication is the transference of a message, through a media, from a sender to a receiver. Human beings need to communicate in order to transmit ideas, thoughts and feelings as a social interaction and this communication can be done by speech, writing or signs. The need to communicate the further, the faster and to reach the higher number of people made compulsory the development of new ways to transfer information. And this is when telecommunications born.
At the beginning communications were made by visual signs as pictorial drawings or smoke until the writing started. The growth of the commerce between nations and empires boost the first telecommunications by messengers using horses and carrier pigeons until the invention of the wheel. Since the revolutionary discovery of the electricity by Benjamin Franklin in 1752 big inventions were created: the Telegraph by Samuel F. B. Morse in 1836 and the Telephone by Alexander Graham Bell in 1876. Heinrich Hertz discovered in 1887 the electromagnetic waves establishing the basis for the wireless telegraphy. In 1906 the first radio emission was transmitted in USA. The photography and the projection of images in television or cinema (first without color or sound and later with both) were developed for the image and sound transmission.

In 1940 the first computer was invented by the German Konrad Zuse and since then a lot of advancements technologically speaking, mixing communications and data processing, have been made until now. Nowadays, from computer networks (which are a collection of transmitters and receivers who are able to send messages to one another through communication channels) to satellites, we can redefine the concept of telecommunications as the exchange of information over significant distances by electronic means.

1.1 State of the Art

Nowadays Internet is one of the most used media of communication. Everyday more people have access to the net [1] and the offered high-speed applications and services keep accruing. [2] is a forecast from 2009 to 2014 made by Cisco Visual Networking Index (VNI) about the growth and use of the global IP traffic. The study splits the obtained forecast both by world zones and by services (file sharing, Internet video, Internet video to TV, video calling, online gaming, VoIP, Web/Data, email) as well as by Business IP traffic and Mobile Data and Internet traffic. The main conclusion is that the IP traffic will increase each year 34 %. In [3], a part from explaining that Internet has been growing analogically to the Moore's law (approximately doubling each year), a solution for the always increasing need of bandwidth (even for future Internet traffic rates) is introduced: optical fibers using Wavelength Division Multiplexing (WDM).

Indeed, optical fibers can afford huge bandwidths (from hundreds of MHz to tens of GHz) at least to cover nowadays needs. In addition, the low loss of signal, the immunity to some interferences, the small size and weight and the easy way to find where fibers are broken by a simple telemetric process, among others, are powerful reasons why optical fibers are strong competitors in the telecommunication technologies.
The ability of a single fiber-optic cable to carry various channels of light or wavelengths is called Wavelength Division Multiplexing (WDM). This technique has been developed gradually, the earliest WDM systems could only support about ten wavelengths per fiber (*Coarse* WDM), nevertheless, nowadays over than one hundred wavelengths can fit into a fiber (*Dense* WDM or even *Ultra Dense* WDM). The capacity of a single wavelength has also increased, in the mid 1990s, the maximum capacity was 2.5Gb/s and right now it is 100Gb/s or more. The combination of the rise of both number of wavelength per fiber and capacity per wavelength throughout this last period has expanded the capacity of this systems by several orders of magnitude [4]. Hence, due to the huge number of advantages and abilities of the photonic technologies, optical networks have been set up as the best solution to solve the enormous and never stopping growth of demand of bandwidth.

To sum up, as the Internet Protocol (IP) is getting relevance and in consequence the demand for bandwidth is growing very quickly and taking into account the advantages that WDM technology can provide, the IP-over-WDM technology is becoming the right choice for next-generation Internet networks [5].

1.2 Motivation

Three principal switching technologies have been investigated in order to decide which one could the most suitable. Even if the Optical Circuit Switching (OCS) is the most common nowadays, WDM technology is evolving to Optical Burst Switching (OBS) and Optical Packet Switching (OPS) [5]. In OCS technology an end-to-end connection (or lightpath) is established. The route of this connection (called path) is determined by some decision algorithms.The same circuit is kept until the connection is terminated and then, the resources used by the established circuit are released. OBS and OPS explore sub-wavelength switching schemes which means that they are able to split and use the capacity of a wavelength.
Fair and quantitative performance comparisons between those technologies are difficult because the resulting metrics of the analysis are not comparable. For example, the rate measuring how many elements have not been accepted in the network called *blocking probability* for OCS and *burst or packet loss ratio* for OBS and OPS are not directly comparable. The solution offered in [6] and in [7] is to evaluate the performance using an identical network topology, traffic matrix and network capacity even if it keeps being a debate among the research community.
The study of all those techniques is becoming compulsory in order to decide which are more suitable to use and to set up in real life to cope this non-stop need of bandwidth. The motivation is then to study the behavior of networks using OCS technique and decide whether if it would be interesting to use it or not. We are looking to decide if OCS is a performing technology, that is to say, if the setting up of OCS in networks, would be reliable, would cover the highest number of the needs, would be suitable energetically speaking and would not be expensive.
Optical Burst Switching technology has already been treated by a previous student in the context where we will work and our aim this time is to create an analogous collection of tools for Optical Circuit Switching.

1.3 Goal of the Work

The main objective of the carried out work is to create a simulation tool for optical networks using OCS in order to study its performance. Simulation is the easiest and cheapest media to reproduce artificially what would happen in real life. Therefore, to lead to our purpose, we will create a framework which is a collection of supporting programs and code libraries to help to develop a software project. In this case, the software project will be in charge to simulate and obtain results about OCS networks in many different conditions. The final aim is to, through the analysis of the obtained results during simulations, extract conclusions to decide in which degree OCS aspects permit a higher number of accepted demands, make the most of the available resources in the network and cost the less economically and energetically speaking. This is what

we call the study of the performance. All in all we will create a framework to study the performance of OCS networks. We will be able as well to highlight the tradeoffs we have to face in OCS networks.
The ideal final work would be to consider all the different conditions that could participate in an OCS network such as inputs, configurations and algorithms but here we are limited in time and we will take into account as much factors as possible. The whole simulation tool (or framework) has been fully implemented in Java. This programming language has allowed us to create a modular tool to mix and compare easily various interesting OCS aspects.

1.4 Overview of the Report

This report starts with a background chapter giving general information about what is an OCS network, why do we need them and which are the main characteristics. Furthermore there is an introduction to the energy issue in optical networks in general. In the 3rd chapter, the fundamental tool of this project, Javanco, is lightly described. There is as well an introduction about what a simulation is and the first developed tools to get familiarized with Javanco. Chapter 4 deals with all the developed tools in order to study the performance of OCS networks. The 5th chapter describes in detail a specific tool who allows to modulate the traffic offered. Chapter 6 makes an energetic approach explaining the initial development of a tool to analyze energetic consumption in WDM networks. In chapter 7 there are some simulation and result analysis of the previous described tools applied to different topologies. The final chapter makes a short summary of the whole work and launch some questions for further research.

Chapter 2

Background

In order to understand the main factors to take into account in this project about OCS networks we will describe firstly the context in a general way. Then, we will go through details explaining each one of the parts more specifically. The main objective is to get an overview about the situation we are dealing with and about the elements that could be considered in the development of it. Even if in this work we have not studied all the elements presented below, it is still useful to know that they exist in order to be taken up again in the future.

2.1 Contextualization

Before entering in any detail it is important to get a general description of the system we are going to study and which elements participate in its operation.

The start point of the whole situation is the need from a certain number of users to send information from where they are to other users away from them. To enable the telecommunication between those users the utilized media is the network. The information is transported from the sender to the receiver using finite resources of this network and the journey of the information can be done in many ways. The goal of the way the information travels is to permit that most of the users can connect and send what they want successfully and as quick, reliable and cheap as possible.

Indeed, users of optical networks send amounts of information at specific times to other users. That is what we call **demand traffic generation**. A **demand** is generally characterized by a node pair, an amount of data (or **traffic rate**) and an arrival and end time determining the duration of the demand. It can happen that demands are scheduled or that, once demands start, they are never ended. This aspects are detailed in sections 2.3.1 and 2.3.1.

As network is the media to transmit information we can say that here we have a **telecommunication network** composed by a collection of links and nodes that connect together making possible the communication between users. A physical network node is an electronic device capable of sending, receiving or forwarding information. Links connect nodes together and in our case, physical links are fiber optic links in charge of data transmission. Physical links are characterized by a number of fibers, a number of wavelengths per fiber supporting a certain traffic rate (Cf. section 4.3.2) and a source and destination node composing a node pair. The layout of the connected devices is called **topology** (Cf. section 2.3.1).

Until here we have seen two important elements in our system: the network (represented by a topology) and its components as well as the generation and the kinds of traffic that users provide. Those elements constitute entrances of the system and that is why we call them **inputs**.

To ensure the success, speed, reliability and economy in the maximum number of demands generated by users, or, in other words, to get a preforming system, we can provide the network of some abilities. For example, if there is a burst in a fiber we can furnish to the network the ability to avoid it (this is what we call **protection**) or if the information, along its journey through the network gets deteriorated, we can provide some nodes of the ability to recover the information damage (we call it **regeneration**). To carry out the provision of those capabilities we have to configure the network either furnishing specific components or planning the way to face some conditions, that is what we will call **network configurations**. We develop each configuration in section 2.3.2.
The particular way to plan how things will work in the network, how the information will be transmitted and which resources will be used to reach a performing network is what we call **planning algorithms**. I we consider that an **algorithm** is an effective method for solving a problem expressed as a finite sequence of instructions we understand that each way to plan how the network works has to be defined, modeled, and translated step by step in order to be included in the work. The algorithms we will develop and other are explained and defined in section 2.3.3.

We are trying to split each one of the parts participating in the optical network operations in order to understand what are their roles and which part occupy in the system. It is important to understand everything separately without forgetting that

all in all forms a whole group working together. One element without the other would not have sense.
This division of elements it is also important because they will be detailed and treated following this order during the work. It has helped us as well to organize the implementation of the project in the programming language.

To follow, we will detail deeply some of the elements explained above. Later we will use all this information to carry out our objective: create models to develop a framework for OCS networks to study the performance.

2.2 Optical Circuit Switching (OCS)

In this section we describe Optical Circuit Switching (OCS) that is to say: the way the network works when this technology is used , the main characteristics of this type of switching and which are the key elements participating on it.

Definition 1 *A lightpath is defined as point to point optical circuit based on wavelength division or time division multiplexing that has a deterministic behavior and guaranteed bandwidth. Unlike a typical routed connection it is all yours and is not subject to the changing demands of others. In general lightpaths are only practical for very high bandwidth applications such as large ongoing data transfers from a detector to a remote site [18].*

The first important step to describe how OCS works is to know what a lightpath is (Def. 1). This definition allow us to say that OCS technology is **connection-oriented**: a connection from a start node to an end node (end-to-end connection) is established before any transmission is done. Connection-oriented networks ensure high reliability and guarantee that data arrives in the right order: once a circuit between source and destination is set up, the same circuit is kept even if no actual communication is taking place in the dedicated circuit, and in consequence there is no lost information once the connection is established. When the connection is terminated a disconnection phase starts up, the resources used by the established circuit are released. Besides, one of the most important advantage is that the switched optical signal do not always need electrical conversion at intermediate nodes so OCS networks can provide all-optical circuits (Cf. 2.3.2) and the signal can be switched independently of its modulation, throughput or protocol. OCS is suitable for long connections that can extend from a few minutes to even months due to the fact that to set up or release a lightpath it takes a few hundred milliseconds. That is why this technology cannot be used to support constantly changing user traffic. On the other hand, each lightpath in an OCS network uses the entire bandwidth of the dedicated channel and the unused bandwidth becomes useless for other lightpaths. This circumstance implies a waste of bandwidth [19].

Other switching architectures as Optical Packet Switching (OPS) or Optical Burst Switching (OBS) have been developed to support in a better way IP-over-WDM traffic. Both techniques try to occupy more efficiently the channels' bandwidth allowing a higher granularity (meaning that the wavelength' capacity can be broken down into smaller parts) and try to make shorter the connection duration. Anyway, OCS is the nowadays chosen technology because of his feasibility. OPS and OBS still a theoretical concept even if little by little they are gaining strength.

So until now we have an idea of how OCS networks work. Since this point we will focus on particular elements or factors already described in the contextualization (2.1: inputs, network configurations and planning algorithms.

2.3 Main Elements taking part in OCS Networks

Now that we know in what consists an OCS network we are ready to get interested in the elements having specific roles on it. In Figure 2.1 there is a diagram to understand easily the organization. This scheme represents the whole system of an OCS network. It includes the main configuration that we give initially to the network, the entrances of the system (the inputs) and the planning algorithms with which the network will operate. Putting together those elements it would be important to know in which degree users are satisfied having transmitted their information in the best conditions and in consequence if the network has been performing enough in those conditions.

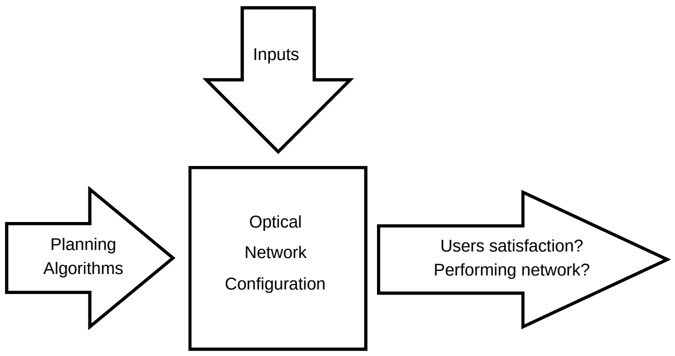

Inputs
Planning Algorithms
Optical Network Configuration
Users satisfaction? Performing network?

Figure 2.1: OCS network elements organization

2.3.1 Inputs

In this section we explain in details the inputs participating in the system. Here we consider the network and its components represented by the corresponding topologies(2.3.1), how much and when the traffic is generated by users (2.3.1) and how long demands stay in the network (type of traffic, 2.3.1).

Topology

As said before, the topology refers to the layout of connected devices. **Network topology** is defined as the interconnection of nodes and links of a telecommunication network. In this work we take into account two types of topologies: **physical** and **virtual** topologies. The physical topology represents the physical design of a network including the devices, location and cable installation. The virtual topology represents how data actually transfers in a network (more details in section 6.1). In general, the most common network topologies are ring, mesh, star, line, tree or bus topologies (Figure 2.2). There are also some existing topologies which cover huge territories in the world as ARPA, ARPA2 and NSFNet (Figure 2.3) for the US backbone network or EON and COST266 for the European backbone.

Topologies, a part from representing series of nodes and links, they are characterized by intrinsic metrics. For a given topology we could calculate distances between node pairs (from a source node to a destination node) or the relative importance of nodes or links among the rest. Those metrics are useful to depict topologies and to compare or classify them. Other metrics and details can be found in section 3.3.

We are interested in topologies because it is one of the basic elements defining the network. As network is the media of the data transmission we want to characterize them to have criterions to decide either if it is a large network or if it is a highly connected network or even how the weight of links and nodes is distributed. A study about this is made in 7.1.

Types of Traffic and Traffic Rate

The type of traffic is defined as the way the connection requests between node pairs are scheduled. Three main types of traffic are *Static* Traffic, *Incremental* Traffic and *Dynamic* Traffic. The traffic demands between node pairs can be represented in a

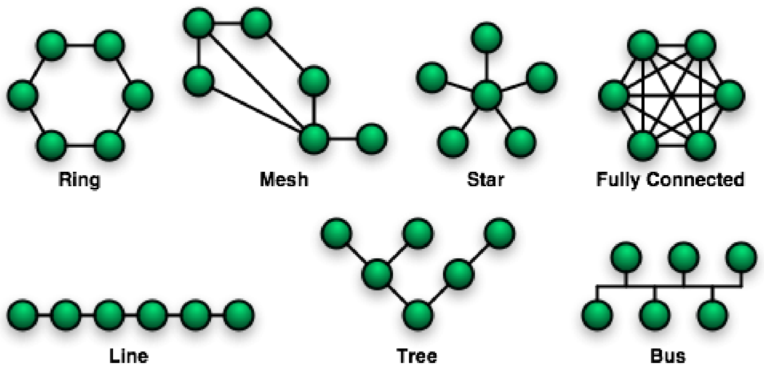


Figure 2.2: Common Topologies

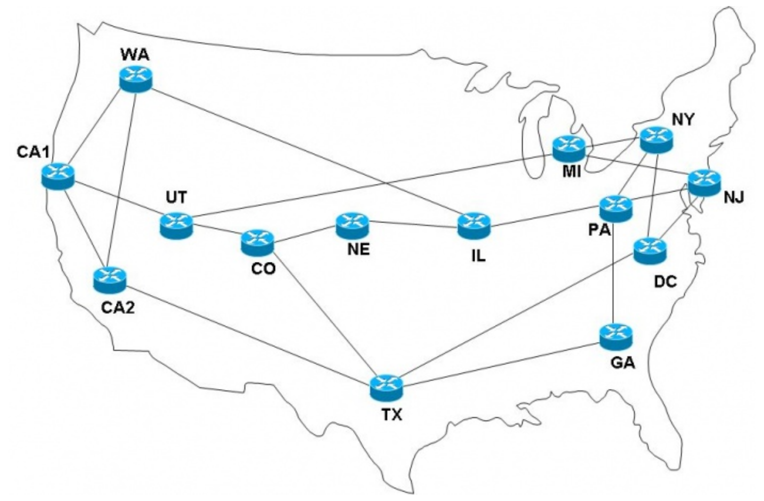


Figure 2.3: NSFNet: US backbone topology

matrix called **traffic matrix** where each element means the number of demands or the needed capacity (traffic rate) for a specific couple of nodes.

The main types of traffic have the following characteristics:

- **Static Traffic :** The entire set of connections is known in advance. The most important step is to set up lightpaths for these connections in order to minimize the network resources such as the number of wavelengths or fibers in the network. The general name of the way wavelengths are selected and granted to a connection for static traffic is known as Static Lightpath Establishment (SLE).
- **Incremental Traffic :** The connections requests arrive sequentially. A lightpath is established for each connection and it remains in the network indefinitely.
- **Dynamic Traffic :** A lightpath is set up for each connection request as it

arrives and the lightpath is released after some finite amount of time. The objective (as for the incremental traffic) is to set up lightpaths and assign wavelengths in a manner that maximizes the number of accepted demands (or minimizes the amount of blocking probability, Def. 2). This problem is called Dynamic Lightpath Establishment (DLE).

Definition 2 Blocking probability *is the statistical probability that a connection cannot be established due to insufficient transmission resources in the network. Usually expressed as percentage or decimal equivalent of blocked connection by accepted and block connections.*

Traffic Generation

In this work, the traffic generation is defined by a statistical distribution or follows a Markovian model. In a further chapter the studied distributions will be detailed. This section gives a general idea about the Markovian model because it will have great relevance in some parts of the work in the traffic generation field.

Markov chains are useful mathematic tools to represent and analyze discrete stochastic processes in systems where the state changes randomly as time goes [8]. They are commonly represented as in Figure 2.4. The birth-death process is a special case of continuous-time Markov process where the states represent the current size of a population and where the transitions are limited to births and deaths. When a birth occurs, the process goes from state n to n+1. When a death occurs, the process goes from state n to state n-1. The process is specified by birth rates $\{\lambda_i\}_{i=0..\infty}$ and death rates $\{\mu_i\}_{i=1..\infty}$. This process would be the right to represent Dynamic Traffic . On another hand, the pure birth process (thus a Markov process with null death rate) would represent Incremental Traffic (Cf. 2.3.1).

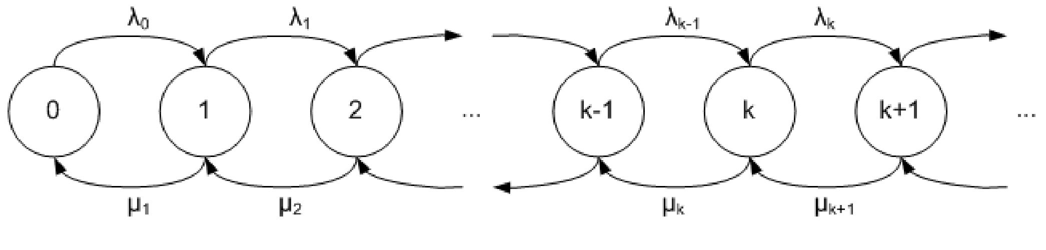


Figure 2.4: Simple diagram of the Markov birth-death process

Other fundamental characteristics of the Markov processes are:
- the states of the system are discrete, finite or infinite
- the states are exclusive: the system can be in one and unique state at the same time

- the evolution of the system is memoryless: the state change does not depend on any past states but only on its present state
A part from the Markov process properties, a Poisson process is defined as follows [8]:
- is a pure birth process
- $\lambda_i = \lambda$ for all $i \geq 0$ and λ is the birth rate
- the time between arrivals has an exponential distribution
- the number of events in time interval $(t\ ,\ t + \tau\ ]$ follows a Poisson distribution with associated parameter λT (so λT is the mean of arrivals in T seconds)
- the increments are independent and stationary

Until now we have gone over some important inputs to take into account in a network. To follow, we will consider which configurations could be found in a optical network.

2.3.2 Network Abilities and Configurations

We remember that the configuration of a network is the rules and components that has to have a network to reach a certain ability. In this part we will describe potential capabilities of optical networks and the needed components to attain the objective. Even if in the resulting work during this project does not consider all the points, it is interesting to know which are the possibilities for optical networks.

Wavelength Division Multiplexing (WDM)

Starting from the fact that the information send by users is converted to optical signals in order to cross optical fibers, we consider that the basic ability of the network is the way the signals are multiplexed. In the whole studied cases the Wavelength Division Multiplexing (WDM) is the chosen technique. This multiplexing method has been rapidly gaining acceptance as a mean to handle the increasing bandwidth of network users. The principle of this technique has been explained in 1.1 but to summarize, it consists in a technology that multiplexes multiple optical signals on a single optical fiber by using different wavelengths to carry those signals. Figure 2.5 schematize how WDM works.

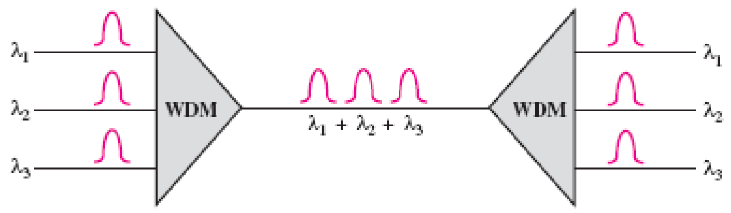


Figure 2.5: Wavelength Division Multiplexing

All-Optical Networks (AON)

Another characteristic to define optical networks is whether they are all-optical networks (AON) or not. AON is defined as a network where the user data transmission, from the source to the destination is made entirely in the optical domain. Otherwise, transitions to the electrical domain at physical nodes would be necessary. Due to the fact that network capacities are increasing, electronic switching nodes cannot process optimally and quickly all the incoming optical data and are not able to be independent of the data rate and the protocol. The result of the electric passage is that electronic switching nodes become bottlenecks, wasting resources and minimizing the performance. If optical signals could be switched without conversion to electrical form we would be able through optical switching to increase the switching speed, and thus, the network throughput, and to decrease the operating power reaching, in consequence, a decrease in the overall system cost.
Having gave this description we understand there are different domains in which a network can work and for that we have to provide nodes of abilities to enable the optical or electric transmission. The main component that will define what occurs in a node regarding this aspect is the type of switches it contains. A short description of the different kinds of switches most commonly used remains interesting to know which elements we can find in nodes [13].

1. *Optical Cross-Connects (OXC)* : Commonly used to provisioning lightpaths and to being reconfigure to support new lightpaths. OXCs are the basic elements for routing optical signals in an optical network system. There are 2 main kinds of OXC:

   - *O-E-O switching* : In those switches the optical signal which is first converted to the electrical domain, then, the signal is switched by electrical means and finally converted back to optical signal [4].
     The advantages of this technique is it that allows 3R-regeneration (Cf. 2.3.2), permits the monitoring of the signal (i.e. to determine if there are errors in the signal) and enables wavelength conversion.
     The disadvantages are the existence of challenges in cost, power and heat

dissipation due to the amount of electronics, the technique is neither data-rate nor data-format transparent and switching bandwidth of electronics cannot keep up with the capacity of optics.

- *O-O-O switching* : Those are all-optical cross-connects. The advantages are that they support optical bypass, they are independent of data-rate and data-protocol and they allow a reduction in cost, size and complexity. The negative points are that the regeneration cannot be done and the signal monitoring is difficult.

2. *Optical Add/Drop Multiplexing (OADM)* : OADMs residing in network nodes insert or extract wavelengths to or from the optical transmission stream without electronic processing (Figure 2.6).

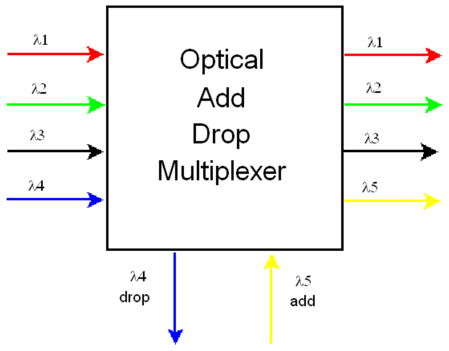


Figure 2.6: Optical Add Drop Multiplexer (OADM) for a four wavelength fiber

3. *Optical Signal Monitoring (OSM)* : OSM receives a small optical portion of the aggregated WDM signal, separates the signal into its individual wavelengths and monitor each channel's optical spectra for wavelength accuracy, optical power levels and optical crosstalk.

During this work we will consider that all nodes have optical switches. Indeed, electrical switches consume more energy than the optical ones participating to a higher performance energetically speaking.

Wavelength Conversion with WDM

In OCS networks lightpaths can whether occupy the same chosen wavelength all along the path to be set up, that is what we call **wavelength continuity constraint**) or switch at some nodes to other diverse wavelength allowing the called **wavelength conversion**. If wavelength continuity constraint does not exist it means that some nodes are composed by wavelength converters to convert the data arriving on one

wavelength on a link into another wavelength before forwarding it on the next link.
There exist some wavelength-convertible networks with full wavelength-conversion capability at each node. In those cases the only important problem is to know which route will have to follow a lightpath to go from its destination node to its end node. Moreover, by contrast to the rest of OCS networks, it has no sense to choose and assign a wavelength to a lightpath because a each node it would change.
There is also a tradeoff to face due to the fact that wavelength converters are expensive and cannot be placed anywhere. On the other hand they could be very useful at some nodes: the type of converters and their placement must be studied. So the question coming to us is *where optimally place a few converters in an arbitrary network?*
In this work we will not deal directly with this problem but we will study what would happen in a network where all nodes are fully wavelength-convertible and we will compare it to a network where wavelength continuity constraint is imposed in order to decide in which cases wavelength conversion could be suitable or more performing.

To permit wavelength conversion we can find optical or electrical technologies. In [9] there is a comparison between both technologies. In general, it is suitable to keep the optical domain that is why researchers try to avoid the passage through the electrical domain developing new technologies. In [10] two possible technique to stay in the optical domain are presented: wavelength selective cross-connects (WSXC) and wavelength interchanging cross-connects.
In the literature there is a lot of other interesting case studies and useful information about this network ability but due to fact that our work does not really consider which kind of converters we have, this discussion rests interesting for further research.

Regeneration

From here we will describe possible abilities for OCS networks even if we do not treat them in our work. The following configurations are interesting to be known and could be investigated afterwards.

In OCS networks, the quality of the traveling signal through links and nodes from a source to a destination can suffer some damages due to the degradation effects such as noise, linear impairments and nonlinear fiber effects. To solve it we can provide **regeneration** at some nodes: the aim is to reconstruct the incoming endamaged signal to make up the original signal. There exist 2 types of signal regeneration: 2R (signal reshaping and reamplification) and 3R (same as 2R plus retiming ability through a clock recovery) [11]. Recent advances in fiber and optical communications technology have reduced signal degradation so far that regeneration of the optical signal is only needed over distances of hundreds of kilometers. Until now, the signal had to be converted to the electrical domain to be regenerated. This fact has a high cost. In [12] we discover incoming techniques to make 2R and 3R regeneration maintaining the optical domain.

Here face again the discussion between the optical and electrical domain and we observe that the new techniques are focused on obtain all optical operating components either for switches, wavelength converters or regenerators. But is it always useful to avoid the electrical domain? Are those abilities as performing in the optical domain as in the electrical one? Which are the energetic consumption savings using optical components rather than electrical components?

The regeneration would participate actively in the performance of the network allowing the right reception of the transported information. The better and reliably the information is received, the more the users are happy with the service.

Waveband Switching

Talking about switching, it is also interesting to pay attention not only to the wavelength switching but also to the waveband switching. This technique would permit to make the most of the transmission capacity allowing multiple wavelengths to be switched together as a single unit. Waveband switching has been proven to reduce the switch sizes considerably in large networks and in consequence to decrease the needed energy to make it work. We are then participating in the energetic performance. Many ways to permit waveband switching are being developed ([14] and [15]) for different types of traffic.

Grooming

The traffic grooming is a procedure of efficiently multiplexing/demultiplexing and switching low-speed traffic streams onto/from high-capacity bandwidth offered by WDM in order to improve bandwidth utilization, optimize network throughput and minimize network cost participating all in all in an improvement of the performance. The work in [16] makes a review of several models and methods for the grooming technique. As happened before, the grooming can be done optically or electronically. In [17] different optical grooming switches are presented and studied.

Protection

Another essential part of the model is the network protection. With WDM networks, the failure of a single link or component may cause the simultaneous failure of several optical channels. For that reason it is important to grant some protection for those networks. Later some mechanisms to solve the failure problems will be explained. Protection contributes as well to to performance of the network. When a component gets broken it would be necessary to find some ways to transmit the traffic between node pairs affected by the failure. At the same time the network is impelled to act rapidly in front of those kind of problems offering effective and performing solutions taking into account the state of the network. This way, users would be satisfied of their service and that is mainly what we are looking for at the very end through every step we take.

To summarize, we have seen different configurations taking part in the operation of an OCS network. Starting by the wavelength division multiplexing technique (considered in the whole work) we have dealt with the optical/electrical discussion in general as well as in detail through other abilities permitting both domains depending on the technology. This fact is as well taken into account in the energetic balance of the network and in consequence, in the energetic performance. Wavelength Conversion, regeneration, waveband switching, grooming and protection seem to be potential capabilities of a network that, utilized correctly forming specific configurations can increase the performance of the network or, what is the same, increase the satisfaction of users.

2.3.3 Network Planning Algorithms

Once having defined which are the inputs and the possible configurations of an OCS network, the next step is to plan the manner the whole system will work. So we will explain some of the algorithms that can be included in the system operation. It will be interesting to propose several algorithms to reach the same objetive in order to compare which one returns the best results, that is to say, accepting the higher number of demands o saving as energy consumption as possible.

Routing and Wavelength Assignment Algorithms (RWA)

The Routing and Wavelength Assignment (RWA) algorithms are in charge of, for a given demand request, find a route (or path) from the start node to the destination node and assign a specific wavelength channel in this selected path to send the information. Both parts, routing and wavelength assignment are planned generally apart and we can conceive different RWA depending on the inputs (topologies and type of traffic) and on the configuration of the network (for example, as said before, if all node of the network allow wavelength-conversion then the wavelength assignment does not make sense while routing still be important). The application of many different RWA algorithms to OCS networks has the objective in this work to compare them in order to be able to decide which one would accept the higher number of demands for a fixed number of resources and distinguish the most performing.

Definition 3 **Linear Programming (LP)** *is a mathematical method for determining a way to achieve the best outcome (such as maximum profit or lowest cost) in a given mathematical model for some list of requirements represented as linear equations. If the unknown variables are all required to be integers, then the problem is called an integer programming (IP) or integer linear programming (ILP) problem.*

Definition 4 *An* **heuristic algorithm** *is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality.*

As explained before, we face the Routing and Wavelength Assignment (RWA) in many ways depending partially on the type of traffic (Cf. section 2.3.1). For example, the Static Lightpath Establishment (SLE) problem for static traffic can be separated in two defined sub-problems: (1) routing and (2) wavelength assignment solved individually. The solutions can be given through Integer Linear Programs (ILP, def. 3) or through heuristic algorithms [20], [21] (def. 4). Due to the fact that Dynamic Lightpath Establishment (DLE) is a harder problem related to dynamic traffic, heuristics methods are generally employed. In the literature we find some studies and comparisons between DLE and SLE solutions applied to various topologies [22], [23] distinguishing which ones make the most of the resources and has lower blocking probabilities.
The aim of RWA is to maximize the number of connections that can be established for a fixed number of wavelengths and a set of connection requests.

1. Routing Algorithms : There are many ways to route that is, to find a path to go from a start node to an end node crossing other elements in the network. The most common is the shortest path algorithm and its objective is to choose the shortest path between a node pair. Different variables can be taken into account to choose which would be the shortest path:

   - Geographical Shortest Path : The path with the shortest length (i. e. calculated in meters) would be the chosen one.

- Topological Shortest Path : The path with the fewest number of hops from the source node to the destination node will be chosen.

In Figure 2.7 there is an example of the shortest paths described above.

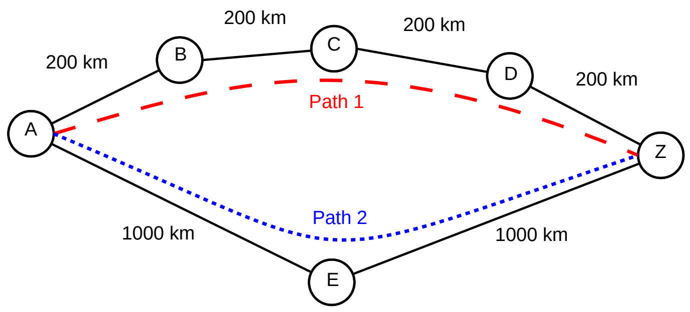


Figure 2.7: For the node pair from A to Z, Path 1 (A-B-C-D-Z) has the shortest geographical path while Path 2 (A-E-Z) has the shortest topological path

These paths can be achieved with the following algorithms :

- Dijkstra : From a source-node in a topology with positive edge path costs, we calculate step by step the cost to arrive to next nodes. We register the paths that have the lower cost. The cost can represent whichever attribute we want to consider to calculate it such as the length of the path or the capacity of the edge.
- Breadth-first Search (BFS) : Proceeds by considering all one-hop paths from the source, then all two-hop paths from the source, etc. until the shortest path is found from source to destination. If there are multiple paths that are tied for the shortest, the BFS finds the shortest path with the fewest number of hops. This can be helpful in network design because fewer hops can potentially translate into lower cost or less wavelength contention.

There are also some specific types of routing such as [24] :

I) Fixed Routing : Always choose the same fixed route for a given source-destination pair. Even if this approach is very simple, the disadvantage is that if resources along the path are tied up, it can potentially lead to high blocking probabilities in the dynamic case, or may result in a large number of wavelengths being used in the static case. Also, fixed routing doesn't consider a network fail in one or more links.

II) Fixed-Alternate Routing : Each node in the network is required to maintain a routing table that contains an ordered list of number of fixed routes to each destination node. A primary route between a node pair is defined as first route in the routing table. An alternative route is a route between the same node pair that does not share any links (is link-disjoint)

III) Adaptive Routing : The route from a source node to a destination is chosen dynamically, depending on the network state. The adaptive shortest-cost-path routing and the adaptive least-congested-path (LCP) routing are explained in [24]. An alternate implementation is to always give priority to shortest paths, and to use LCP only for broken links.

When routing it is common to have to face some tradeoffs, i.e. between the fewest-hops path and the shortest-distance path or between the shortest-distance and the minimum-regeneration path (the path with shortest physical distance is not necessarily the path with minimum regeneration) [4]. Those tradeoffs will play important roles in the performance because one of the parts will bring higher performance but the other one will decrease it.

2. Wavelength Assignment Algorithms : Algorithms dealing with the wavelength assignment and ensuring the wavelength continuity constraint are such as:

   I) Wavelength-Assignment Heuristics : For incremental or dynamic traffic, heuristic methods must be used. Some heuristic solutions could be:

      - Random Wavelength Assignment : Among the free available wavelengths in all the links all along the previously chosen path, one is chosen randomly.
      - First Fit (FF): In this scheme, all wavelengths are numbered, the first available wavelength all along the path is then selected [25].
      - Least-Used (LU) : Selects the the least used free wavelength in the network, trying to balance the load among all the wavelengths.
      - Most-Used (MU) : Selects the most used free wavelength in the network. In [26] it is said that MU ”outperforms LU and FF doing a better job of packing connections into a fewer wavelengths and conserving the spare capacity of less used wavelengths”.

      Those algorithms and others are clearly explained and compared in [26] but those above will be our studied cases in the project.

   II) Routing Wavelength and Rate Assignment (RWRA) : Due to the fact that telecommunication networks employing WDM are expected to be increasingly heterogeneous and support a wide variety of traffic demand a new problem called Routing Wavelength and Rate Assignment (RWRA) arises. The work in [30] considers a single link that can have a combination of bit rates, and that each physical link in the network may support a combination of bit rates, each on a separate wavelength. The aim is to design transparent Mixed Line Rates (MLR) networks that will reduce the amount of electronic signal processing at the nodes and the waste of bandwidth.

   III) Routing Wavelength and Time Slot Assignment (RWTA) : Another solution to reduce inefficiency of links is described in [31] where a combination of WDM and Time Division Multiplexing (TDM) is made to obtain the Routing Wavelength and Time-slot Assignment (RWTA)

Furthermore it exists other RWA algorithms based on different variables as *RWA for scheduled lightpath* detailed in [27] and [28] or *RWA based on statistics* explained in [29].

Some of those algorithms have been implemented to obtain a feedback about the performance.

Wavelength Conversion Aware Algorithms

Wavelength conversion tends to be very useful as it permits to save resources when it is well used. Place sparse converters [32] or construct different heuristic algorithms [33] are diverse proposals dealing with wavelength conversion as optimally and performing as possible. The main conclusion of [34] is that the benefits of conversion are largely dependent on the network load, the number of available wavelengths, and the connectivity of the network.
In the literature there are a lot of other situations where the wavelength conversion takes part. Usually we find it mixed with a specific RWA condition or other restrictions.

In this project we will try to observe what happens in an OCS network where all nodes have the ability to convert wavelengths and comparing with other routing and wavelength assignment algorithms to decide in which situations we obtain the lowest blocking probability.

Since here other kind of algorithms for different objectives will be explained in a very general way because they have not been taken in consideration in the final work of this project but they could be interesting to take into account in further research.

Regeneration Aware Algorithms

The algorithms allowing regeneration will try, at some selected regenerator nodes to receive the incoming signal and will reshape, reamplify and retime it as the original signal. The main disadvantage is that regenerators add substantial cost to the communication system and so the principal aim is to minimize their use and to put them strategically [35], [36]. The placement depending on the number of resources, the RWA planning [37] or on the geographical length of lightpaths (3 heuristics methods are implemented in [38]) are described in literature. In [39] it is showed that the combined approach of regenerator placement and routing, both based on physical degradation effects, may decrease the blocking probability. So here again we find a tie with the energetic performance related with the domain in which nodes of the OCS network works.

Grooming Aware Algorithms

The grooming technology, trying to pack or separate (multiplexing or demultiplexing) efficiently switching low-speed traffic streams onto/from high-capacity bandwidth has also to be planned in order to use it when it is worth it.
Traffic grooming can be classified depending on the number of lightpaths allowed in a connection route as single-hop traffic grooming (SH-TG), which is a grooming done all along a single link, and multi-hop traffic grooming (MH-TG), which the grooming is done in various links in a row. In [40] it is said that MH-TG is more general and resource efficient than SH-TG, because it allows connections from different source-destination pairs to share the bandwidth of a lightpath. In [40], [41], [42], [43] and [44] we find detailed algorithms to obtain grooming in WDM networks.

Protection Aware Algorithms

The very main objective of the protection aware algorithm is to react quickly and efficiently in front of a failure in the network at any point. That is why we can plan protection at different levels:

1. Capacity Protection :

    I) Preplanning : Pre-allocate spare resources when a lightpath is established or, if the traffic is static, during the planning of the network. It exists different implementations:

    - *Dedicated Protection* : Reserves spare resources for each element in the network. Reduces the complexity of failure recovery but requires at least 50 % WDM channels reserved.
    - *Shared Protection* : Reserves some WDM channels or fibers to be shared by more than one element. Reduces the amount of spare resources and improving network utilization for working traffic but increasing the recovery procedure complexity.

    II) Provisioning : Plan the network with an amount of resources that exceeds the real working-traffic in order to activate new connections to restore the faulty ones in case of failure. This method increases the flexibility of the network system and improves resource utilization for the working traffic giving the network the chance to survive to multiple simultaneous failures.

Some examples applied to a 4-node ring network and to a mesh network can be found in [45].

2. Elements Protection :

   - Path protection : In case of failure each single interrupted path is switched on its preplanned protection path.
   - Link protection : This mechanism reacts to a failure bypassing the damage components and passing through an alternative path.
   - Node protection : There is the possibility to provide the nodes the ability to resist to a failure in its switching subsystem or to provide nodes to additional resources (for example with a redundant number of line cards [46])

3. Disjoint Alternate Path (DAP) : In [47] DAP protection algorithm is set up. The aim is to find, from a given virtual topology, a route for each of the lightpaths in the photonic network such that a single optical link failure leaves the virtual network connected. The performance of the algorithm is tested with NFSNet and the ARPA2 physical networks.

To evaluate if those algorithms are good enough some parameters can be extracted and compared to qualify the performance: the spare network capacity, the switching speed of connection recovery and the cost of protection mechanism. Of course there are tradeoffs between them but from those parameters we would be able to judge the performance of the WDM network in protection aware. Some other parameters and examples are presented in [48], [49] and [50].

2.4 Energy Aware

Apart from switching technologies and algorithms there is also another challenge that is gaining relevance nowadays, the energy consumption and the heat dissipation of the IP Over WDM networks. Even if the energy consumption of Internet represents a small fraction of the total electricity supply (about 1% of the total electricity consumption in broadband enabled countries [51]) it will keep growing. Some techniques are being developed in order to **green the Internet** [52]. Internet usage, bandwidth demanding and components capacity are increasing day by day thus the energy consumption of network equipment is rising participating to the greenhouse effect. The transmission and switching stages are mainly the energetic consumers. In [53] mixed integer linear programming (MILP) models and heuristic approaches are proposed to reduce the energy consumption from 25% to 45%. In [52] the idea is to put network interfaces and other router and switch components to sleep when they are underutilized.

Since here we have dealt with the different stages to consider while we are designing and planning an optical network as well as several algorithms to simulate them in order to calculate the performance of this network. Moreover we have introduced the energetic problematic which is starting to be an important issue because of the increasing energy consumption but also because it is damaging the environment.
Nowadays we are looking forward to make networks both efficient and cheap (economically and energetically speaking). The main challenge of the engineers is, considering a given topology, a type of traffic, and some specific techniques explained above, to optimize the tradeoffs of those networks. That is why we are looking for a tool having the ability to furnish networks of all those elements and to compare the results to decide which one is the more suitable and in which cases.
For that reason we will start to present the way in which we have carried out the objective to study the performance of OCS networks through different inputs, configurations and algorithms.

Chapter 3

Initial Developed Work

The question at this point is: *how will we carry out the way to judge the OCS network performance for a given collection of inputs, configurations and algorithms?*
The answer is: through simulation.

"**Simulation** seems to be the best available alternative to the deployment of expensive and complex testbed infrastructures for the activities of testing, validating and evaluating optical network control protocols and algorithms" [54]. Indeed, nowadays simulation is used to perform experiments that would be costly, dangerous or too time consuming with real systems. But to go through simulation we will have to conceive models (def. 5) to represent most of the elements of the previous chapter. In [55] it is said that "provided simulation models are adequate descriptions of reality (they are valid) and experimenting with them can save money, suffering and time".

Definition 5 *In the most general sense, a* **model** *is anything used in any way to represent anything else. However a* **conceptual model***, may only be drawn on paper, described in words, or imagined in the mind. They are used to help us know and understand the subject matter they represent.*

To create and run a valid simulation there are some basic steps that must be followed [57]:

1. Problem Formulation : Formulate a clear and understandable statement of the problem.

2. Setting of Objectives and Overall Project Plan : Define which questions have to be answered by the simulation and the appropriate methodology to achieve them.
3. Model Conceptualization : Elaborate a model abstracting the essential features of the problem until a useful approximation results.
4. Data Collection : Adapt the input data collection to the complexity of the model.
5. Model Translation : Select the appropriate simulation language and program the model.
6. Verification and Validation : Answer to the following questions: is the model free from logical errors? does it do what we expected to? and is the model an accurate representation of the actual system?
7. Experimental Design : Determine alternatives as length of the simulation, number of replications or initializations.
8. Production Runs and Analysis : Produce of results and analyze them to estimate system performance.
9. Eventual Additional Runs : Proceed to more runs if necessary based on previous analysis.
10. Documentation and Reporting : Create program documentation to understand the way the program works and eventually modify it and report chronologically the work done and the decisions made.

If we consider our goal in this work we should be able to reconstruct step by step the simulation process. So the formulation of our problematic is that for a given OCS network, characterized by some inputs, configurations and algorithms, which are the most performing combinations or, in other words, what would be the system that ensures a maximum number of accepted demands, the high quality and reliability of the received information and the higher economic and energetic saves. The rest of the steps will be considered all along this project.

The aim of the software developed in this work is to be flexible and heterogenous in order to consider as much inputs, configurations and algorithms as possible. In the literature we find some other simulation tools for optical networks as *Optical WDM Network Simulator (OWns)* [58] suitable for Wide Area Networks (WAN) or *Automatically Switched Optical Network (ASON)*, both conceived as extensions of *NS-2 Simulator* [59]. *NS-2 Simulator* "provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite)

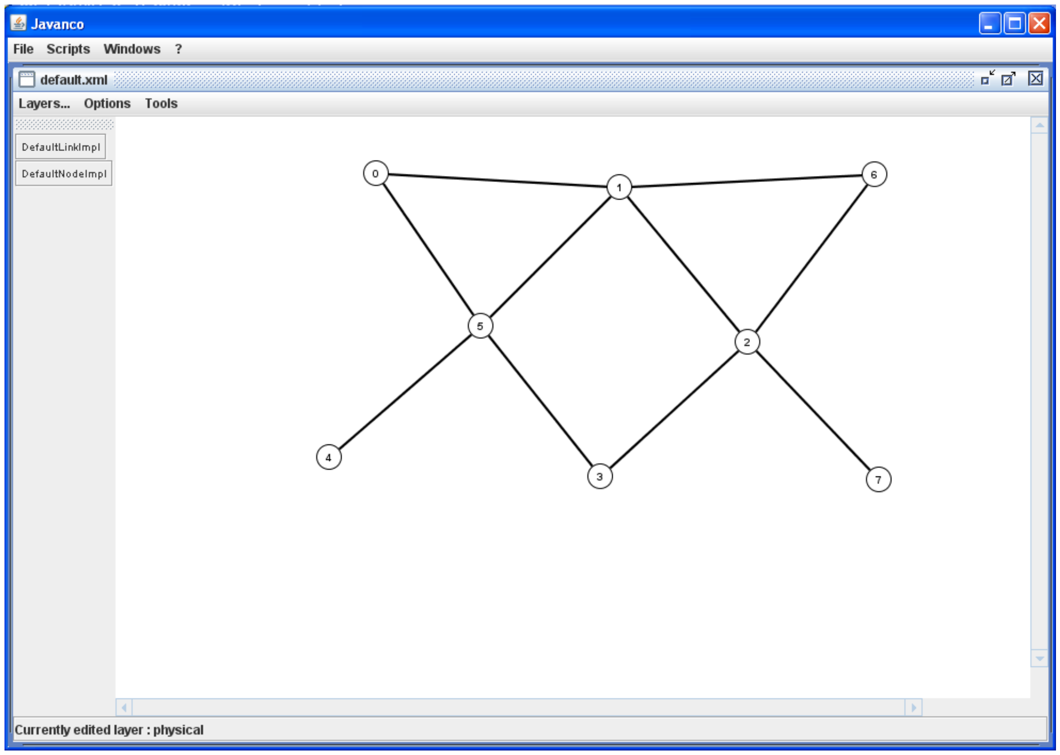


Figure 3.1: Graphical representation of an existing file (default.xml)

networks". There are also *Hegons* [60] which is a Heterogenous Grooming Optical Network Simulator or *SimulNet* [54] which is the most similar simulator to Javanco (Cf. section 3.1).

3.1 Javanco

The work made in this project is on top of an already existing framework called Javanco. Javanco has being developed since 2006 in the TCOM laboratory at the EPFL and it has been conceived to be a powerful tool for network simulation. As it is programmed in Java language it permits an object oriented structure and an simple way to develop and test network configurations and algorithms applied to a huge range of topologies. Moreover Java has a modular conception allowing easily the addition of functions that can be applied on various parts of the existing (or future) code.

Since its creation, Javanco has been furnished of different abilities: an interface to create and represent topologies loaded and saved in XML (eXtensible Markup

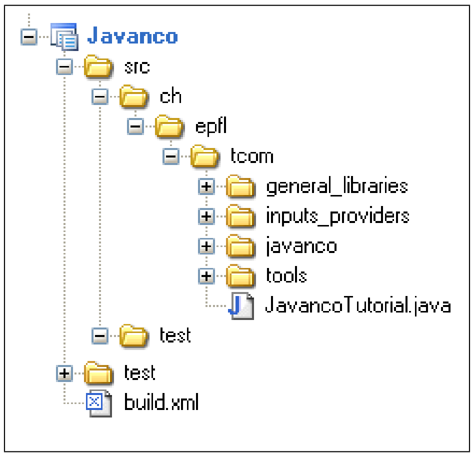


Figure 3.2: General structure of *Javanco* project in JCreator

Language) file format (Figure 3.1) [1], Javobs which is a framework to simulate OBS networks [61] and a tool to compare various RWA algorithms. All of those tools are complemented with a performing interface to represent graphically the obtained results during simulations. This interface can be used during any main function execution to get output data.

For the whole Java programming we will use the **JCreator** Integrated Development Environment (IDE) [65]. In a workspace called *TCOM_Sonia* we will define a project called *Javanco* to work on it and develop each part. In Figure 3.2 we find the general structure of the project we are working with. The folder `src.ch` contains all the already integrated elements in Javanco and the `src.test` folder allows all those who are working with Javanco to construct and try new tools to check results and decide afterwards if they are interesting enough to integrate them to the final framework.

[1]Note that in this interface other nodes and links could be added through the buttons *DefaultNodeImpl* and *DefaultLinkImpl*respectively. The currently edited layer is displayed on the bottom of the interface (”physical” in this case) and the menu *Layers...* offers possibilities as *New Layer...* to create a new layer, *Visibility* to choose the displayed layer and *Actually Edited* to define which layer is being modified. The menu Options permits the use of offsets, animations or 3D representation. No menu appears when *Tools* is pressed. On the top there is *File* which allows the creation of a new representation by *New*, the opening of an existing file (default.xml here) by *Open...*, the save of a created or modified topology with *Save* and the exit of the application with *Quit*. Groovy files can be created or loaded by the menu deployed in *Scripts* and if other representations are opened at the same time, *Window* permits the choice.

3.2 Getting familiarized with Javanco

The very first step to start programming has been to get familiarized with Javanco tools. With this objective a `JavancoTutorial` class has been created in order to understand carefully the Javanco process.

Firstly Javanco needs to be initialized detecting the root of the Javanco project (the JAVANCO_HOME) to deduce many files settings and configuration as well as setting up Javanco properties. From here Javanco is ready to start being used through the Javanco main object: the *AbstractGraphHandler* or commonly named "agh". There are 2 methods to get one of those objects as those objects cannot be instanciated:
- using directly the function:
`AbstractGraphHandler agh = Javanco.getDefaultGraphHandler();`
- creating a *GraphHandlerFactory* first and then obtaining a new GraphHandler (specially useful if more than one agh have to be created):
`GraphHandlerFactory ghf = Javanco.getDefaultGraphHandlerFactory();`
`AbstractGraphHandler agh = ghf.getNewGraphHandler();`

Once the agh is created we can:
- load an existing XML file with the function: `agh.openNetworkFile("<name>.xml");`
- create a new network with our chosen name: `agh.newNetwork("<name>");`

Two other functions connecting the agh with the XML file must be highlighted:
- `agh.activateMainDataHandler()` permits the listing of new added nodes in the XML file
- `agh.activateGraphicalDataHandler()` permits the representation of the XML in the right position in the 2D plane.

And from here we can do whatever we want as add, remove or modify nodes, links and layers. In fact these three elements are defined as *containers* (extending from an abstract class *AbstractElementContainer*, Figure 3.3) and they are characterized by properties, attributes and contents that can be added, checked or modified depending on the specific role of the element in the network. Later some clear examples will be exposed.

Therefore, now we are able to begin to play with this objects and functions in order to create the first tool of the work, a **Topology Analyzer**.

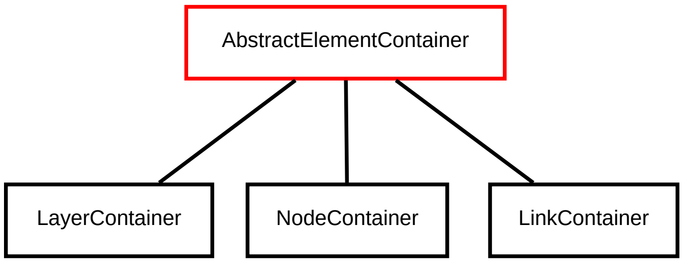


Figure 3.3: Diagram of the *Container* classes organization. Key in A.1

3.3 Topology Analyzer

As said before, topologies are considered as one of the inputs of the OCS networks. That is why, even if it is conceived as a tool to get practice with Javanco it keeps being very useful in this framework.
To begin we have chosen an allocation for this tool in the Javanco organization. The most coherent place is in an existing folder called *tools* (Cf. Figure 3.2) that contains other analysis tools. We create inside the file *topology_analyser*.

The goal of this tool is to analyze topologies so the first step is to create them. As explained before, topologies can be drawn by hand and saved in an XML file, can be loaded from an already existing XML file or can be created by another Javanco tool that permits the automatic generation of topologies. With this tool we can obtain a huge range of topologies from polygons and tessellations to random topologies with a certain % of node pair connections.

Some definitions have to be made to understand clearly the meaning of many metrics [62].

Definition 6 **Centrality** *is a measure for the importance of a vertex or an edge. It is often applied in social networks, sometimes for marketing purposes. How to measure importance depends on the application and is mostly based on intuition and trial and error.*

Definition 7 *The* **degree** k_v *of vertex* v *is the number of edges incident in* v*.*

Definition 8 *The* **distance** $d(v, w)$ *between two vertices* v *and* w *is the length of the shortest path between* v *and* w*. By definition* $d(v, v) = 0$ *and* $d(v, w) = \infty$ *if there is no path between* v *and* w*.*

Definition 9 *The* **transitivity** *of a graph is the proportion of times two neighbors of a vertex are neighbors among themselves.*

Definition 10 *A* **component** *of a graph is a subgraph in which any two vertices are connected to each other by paths.*

We remark that the object furnishing the shortest path between a given pair of nodes is already done in the Javanco framework and uses a Dijkstra algorithm. This object is called `JavancoShortestPath` and has the ability to receive the name of an attribute (for example "length" or "hops") and give the shortest path based on this attribute.

As Java programming permits to make modular structures we can take advantage of it and split the topology metrics we want to calculate in different blocks. So the final organization ordered by files is represented in Figure 3.5 and the diagram of the internal classes structure is in Figure 3.4. We observe that the main root of this tree is an object called `AbstractExperimentBlock` and we find it each time we develop a new "experiment" that is to say, each time we create a collection of objects with the aim to be tested.

The file organization of the topological analyzer is detailed below following the structure of Figure 3.5.

1. `link_metrics` : Those are the metrics concerning links by themselves thus the **link betweenness** which means the proportion of shortest paths passing through a link. This metric is relative because it depends on all the links of the network and each link has his own normalized value. It is considered as a centrality metric (def. 6).
    - `LinkBetweennessComputer`: For a given `LinkContainer`, calculates the result of how many times this link is crossed by all the shortest paths and divided by the total times all links are crossed.
    - `LinkCentralityComputer`: Is the abstract class above the previous one that collects for each `LinkContainer` the corresponding metric.
2. `network_metrics`: The classes in this folder get all the metrics concerning the whole network.

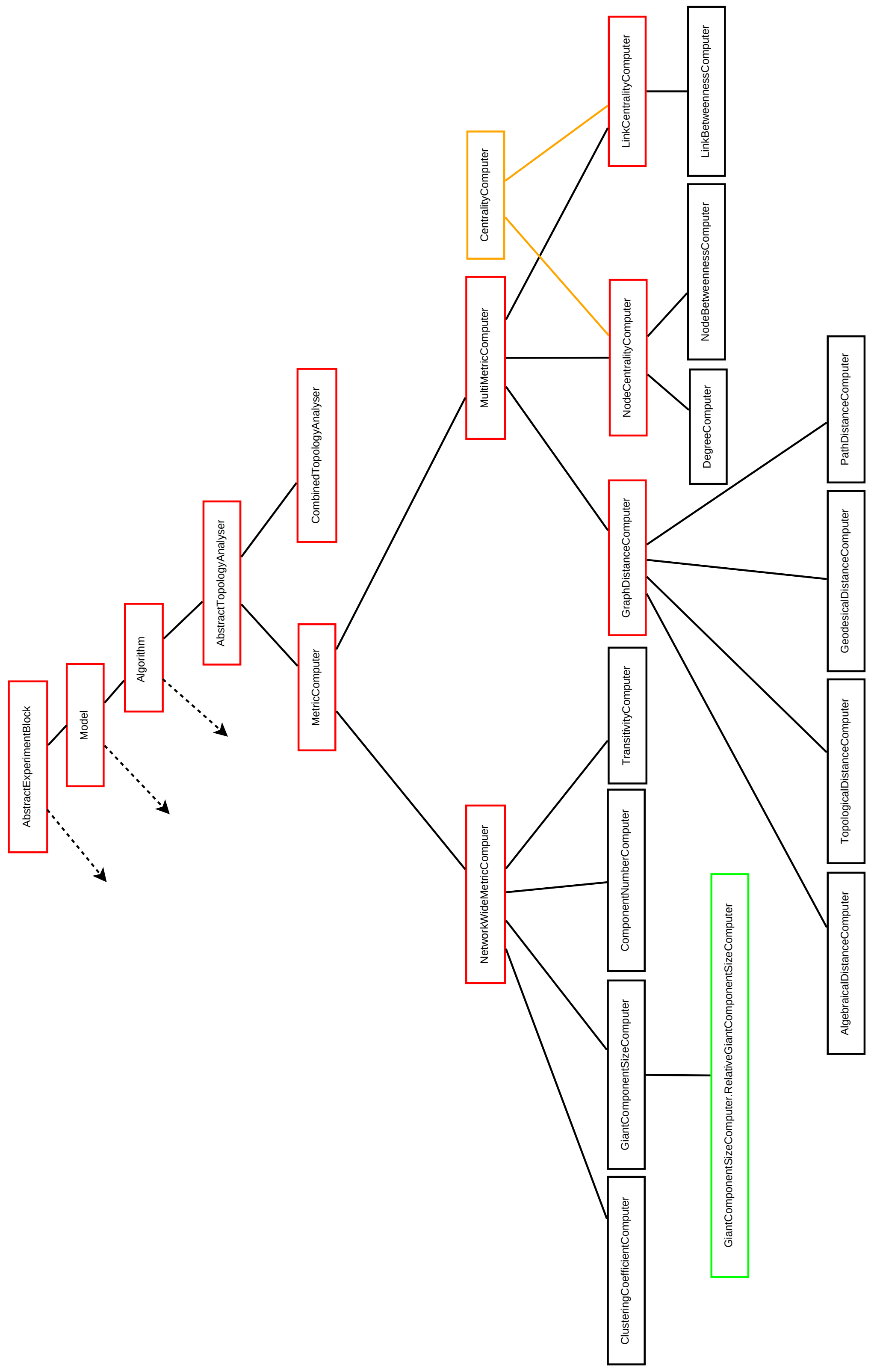
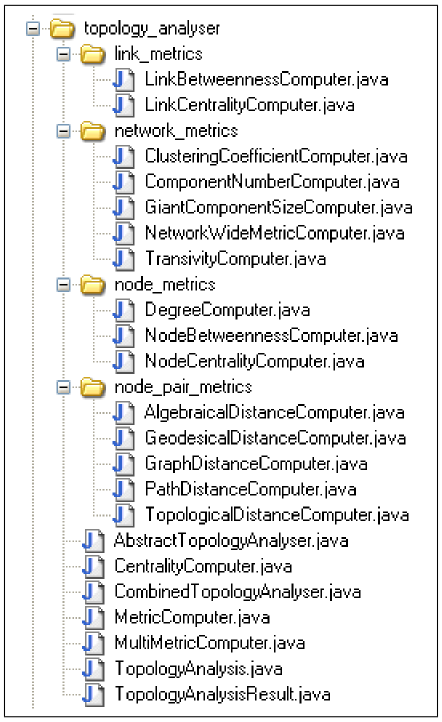


Figure 3.4: Diagram of the objects organization for the topological analyzer, key in A.1



```
topology_analyser
    link_metrics
        LinkBetweennessComputer.java
        LinkCentralityComputer.java
    network_metrics
        ClusteringCoefficientComputer.java
        ComponentNumberComputer.java
        GiantComponentSizeComputer.java
        NetworkWideMetricComputer.java
        TransivityComputer.java
    node_metrics
        DegreeComputer.java
        NodeBetweennessComputer.java
        NodeCentralityComputer.java
    node_pair_metrics
        AlgebraicalDistanceComputer.java
        GeodesicalDistanceComputer.java
        GraphDistanceComputer.java
        PathDistanceComputer.java
        TopologicalDistanceComputer.java
    AbstractTopologyAnalyser.java
    CentralityComputer.java
    CombinedTopologyAnalyser.java
    MetricComputer.java
    MultiMetricComputer.java
    TopologyAnalysis.java
    TopologyAnalysisResult.java
```

Figure 3.5: File structure of the *topology analyzer* tool

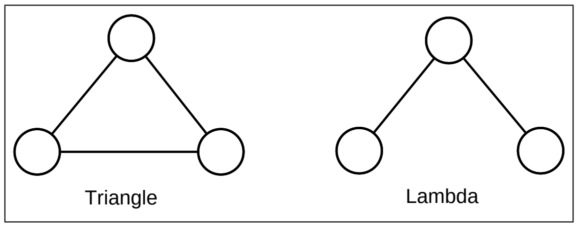


Figure 3.6: Node triangle and node

- `ComponentNumberComputer`: Returns how many components (def. 10) has the studied topology. It is important to know if a graph has one or more components because some metrics only have sense in one component graphs.
- `GiantComponentSizeComputer`: Calculates the number of nodes of the biggest component.
- `TransitivityComputer`: With the definition in 9 and considering the two subgraphs in Figure 3.6 we obtain the transitivity as

$$T = \frac{3 \times \text{number of triangles in the graph}}{\text{number of lambdas in the graph}} \tag{3.1}$$

- `ClusteringCoefficientComputer`: The clustering coefficient differs from the transitivity in the way ratios between triangles and lambdas are calculated. In this case we calculate:

$$C = \frac{1}{|\nu|} \sum_{v \in \nu} \frac{\text{number of triangles with vertex v}}{\text{number of lambdas with vertex v}} \tag{3.2}$$

where ν represents the number of edges in the graph.

- `NetworkWideMetricComputer`: Holds the method to count the number of triangles of a given agh. Returns the value of the objects inherited from him.

3. `node_metrics`: The contained files are relative to single nodes properties in a topology.

   - `DegreeComputer`: For a given `NodeContainer` calculates his degree (def. 7).
   - `NodeBetweennessComputer`: For a given `NodeContainer` calculates the result of how many times this node is crossed by the shortest paths divided by the total times all nodes are crossed.
   - `NodeCentralityComputer`: Is the abstract class above the previous classes and collects for each `NodeContainer` of the agh the corresponding metrics.

4. `node_pair_metrics`:

   - `AlgebraicalDistanceComputer`: Has a single method and returns a matrix with the results of the `JavancoShortestPath` based on the attribute "hops".
   - `GeodesicalDistanceComputer`: Calculates for each pair of nodes the distance. It uses the coordinates of the source and destination node and do geometrics. So this is not the distance of the shortest path but the distance in straight line.
   - `PathDistanceComputer`: Returns the matrix with the `JavancoShortestPath` results based on the attribute "length".
   - `TopologicalDistanceComputer`: Gets the `JavancoShortestPath` results based on the attribute "length" and then it fills in a matrix counting the number of hops considering the previous obtained paths.

5. The rest of objects in `topology_analyser` are abstract classes or useful objects to simulate (Cf. section 7.1).

In this section we have described how do we have organized and implemented several metrics based on topologies in order to be used to characterize our studied topologies.

3.4 Erlang Analysis

Until here we have dealt with already existing objects in order to understand how they work and how we can extract information from them.

From now we create new objects to generate and simulate types of traffic (Cf. section 2.3.1) which are considered as inputs of our OCS network system and so they are useful as well to our further study of performance.
As we do not consider WDM configuration yet (we do not have fibers and wavelengths in the links), we create a `FiniteCapacityLink` object which puts in each `LinkContainer` of the network a certain initial capacity (a capacity unity would correspond to a wavelength channel in WDM thus each demand will occupy one capacity unit in links all along the path). The capacity is decreased or increased when links are used or released.
The idea to put a particular`FiniteCapacityLink` in each `LinkContainer` is a clear example to describe how the *containers* work: each link is a `LinkContainer` in the network and apart from his default attributes (start and end node) we can add abilities as in this case, another object in charge of defining a certain capacity to the link. Like this we can define little by little how are the links in the network, which properties do they have and how do they work. The same idea can be extrapolated to nodes and layers.

Definition 11 **Traffic offered** *is the total number of calls (including call attempts) submitted to a group of trunks or switches. Its average intensity in* **erlangs** *is given by the mean number of calls arriving during the mean holding time [63].*

Definition 12 *The* **erlang** *is the unit of traffic intensity. If the average number of simultaneous calls in progress in a given period over a particular group of trunks is N, the traffic intensity is said to be N erlangs; i.e. one permanently engaged circuit has a traffic intensity of one erlang [63].*

We create as well a basic traffic generator with time dependency. This one receives as parameters an agh, a number of desired demands, a λ parameter designating the arrival rate (or the mean of the exponential distribution defining times between arrivals), a μ parameter designating the service rate (or the mean of the exponential distribution defining the demands duration) and a capacity to furnish to links. The object called `TimeDependingModel`, modeling a dynamic traffic generator, creates n demands between random nodes of the agh topology, where n is the number of demands given as parameter. Each demand will occupy a capacity of 1 and the start and end times will depend on exponential distributions defined by λ and μ. All links will be provided with the capacity given as parameter. The routes of the resulting demands are chosen by a shortest path routing algorithm.

With those elements we have all the ingredients to create an Erlang model to prove that in a network receiving a defined offered traffic (def. 11) generated by a dynamic traffic generator and with limited channels, a number of demands are rejected with a certain blocking probability.

Theoretically, the relation between those parameters corresponds to the Erlang formula. In B there are the Erlang formula and the results for several parameter values, particularly those we will study below.
We want to validate the model showing that the output data of the simulation are close to the theory.

The arrival (λ) and the service (μ) rates determine the offered traffic T; in our model we will fix $\mu = 1$, with $\lambda = [1, 2, 3, 4, 5, 6, 7]$ (thus the mean of the distributions will be $\frac{1}{\lambda} = [1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}, \frac{1}{7}]$). We choose a 2 nodes and 1 link network and we provide this link with capacities or number of channels $N = [1, 2, 3, 4]$. The blocking probability will the output data after simulation. It is obtained counting the number of lost demands by the total number of demands (processed or lost)during the simulation.

Before continuing with the Erlang problem we have to highlight the way random numbers are generated and how output values are represented.
Indeed, a Random Number Generator (RNG) is an essential tool in this whole work. Thanks to the RNG we are able to obtain random demands (choosing two random numbers to form a node pair), random times (based on specific distributions) and random numbers to be compared to thresholds. A truly random number sequence consists of a totally unpredictable stream of numbers that are in consequence unreproducible. In practice, it is a very difficult task to obtain real random generators based on the observation of some naturally random physical process. Mechanisms to generate *pseudo* random numbers have been introduced extensively to skip this limitation. The biggest advantage of the Pseudo RNGs (PRNG) is that they generate numbers according to some algorithm and so reproducibility of a given statistical experiment is possible [55]. So for our experiments we will use PRNGs. The PRNGs work based on an initial seed to generate random numbers. In order to obtain more reliable results it will be necessary to run each experiment several times changing the seed of the PRNG and getting the mean of the obtained values.
The results represented in the graphical interface of our work will be always the mean

value of the results obtained for at least 5 different seeds.

Taking up the Erlang problem, we show the results of the simulation in Figure 3.7.

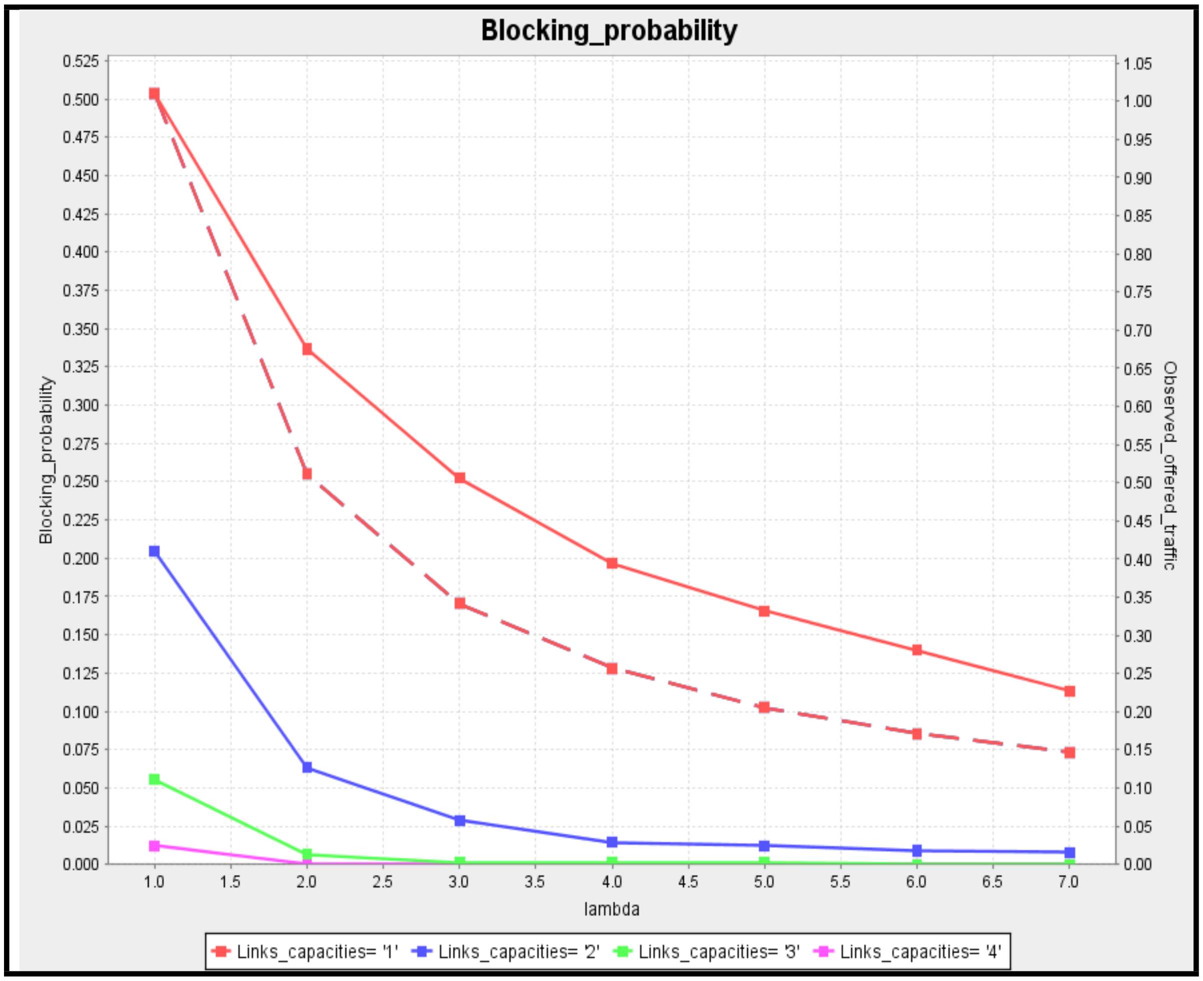


Figure 3.7: Erlang model simulation

In the Figure 3.7 there are represented in solid lines the different blocking probabilities (P_b) for the studied range of λ and capacities. In dotted lines there is the traffic offered T calculated based on the results. In this graphic interface we can extract the exact obtained values in an excel sheet and this is how we will report the simulated values. Moreover, the interface allows to calculate the confidence interval to indicate the reliability of the results at a certain % and we can even regulate the % of confidence we want. In Tables 3.1 and 3.2 there are the reported the most relevant numerical data and confidence intervals at 95% of the simulation and in Table 3.3 the theoretical data for the blocking probability.

λ	μ	Theoretical traffic offered T	Simulated traffic offered T'	Confidence interval 95%
1	1	1.00	1.00	[0.89, 1.12]
2	1	0.50	0.51	[0.45, 0.56]
3	1	0.33	0.34	[0.30, 0.38
4	1	0.25	0.26	[0.22, 0.28]
5	1	0.20	0.20	[0.18, 0.22]
6	1	0.17	0.17	[0.15, 0.19]
7	1	0.14	0.15	[0.13, 0.16]

Table 3.1: Theoretical and simulated offered traffic

In Table 3.1 there are the theoretical and simulation values of the offered traffic as well as the confidence interval at 95% of the simulation results. The simulation values are different from the theoretical ones first, because we are simulating with a model which represents the theoretical concept and second, because simulations are run for several seeds for the PRNG and because of this the returned values are different at each time, at the end the mean is done.
We observe that all the theoretical values are in the confidence interval so with 95% of probability the simulated values would be in this interval. Then, even if the simulated values are not exactly the theoretical ones, we can say that they are correct as the theoretical value is in the confidence interval.

Channels	P_b $(T = 1)$	P_b $(T = 0.5)$	P_b $(T = 0.33)$	P_b $(T = 0.2)$	P_b $(T = 0.14)$
1	0.503 [0.463, 0.542]	0.336 [0.293, 0.379]	0.252 [0.216, 0.287]	0.165 [0.135, 0.195]	0.113 [0.087, 0.139]
2	0.204 [0.162, 0.246]	0.063 [0.039, 0.086]	0.028 [0.015, 0.041]	0.012 [0.003, 0.020]	0.008 [0.001, 0.014]
3	0.055 [0.027, 0.083]	0.006 [-0.002, 0.015]	0.001 [-0.001, 0.003]	0.001 [-0.001, 0.003]	0

Table 3.2: **Simulated** blocking probabilities results

Channels	P_b $(T = 1)$	P_b $(T = 0.5)$	P_b $(T = 0.33)$	P_b $(T = 0.2)$	P_b $(T = 0.14)$
1	0.500	0.333	0.250	0.166	0.125
2	0.200	0.076	0.040	0.016	0.008
3	0.062	0.012	0.004	0.001	0

Table 3.3: **Theoretical** blocking probabilities represented on Annex B

We find the same situation with Tables 3.2 and 3.3: the theoretical values are in the confidence interval of the simulation results (written in [#, #] beside the simulation value). As a brief remark we explain that it is normal to find negative values in some confidence intervals because they are representing a limit and even if we know that those values have to be positive, the statistical range could return negative values. Moreover, the last value for 3 channels is 0 and it does not exist confidence interval because it is not probable to obtain another value.
Having explained this we can confirm that our model is a good one and we have

validate it.

Topology and Erlang analysis have helped us to understand the main organization of Javanco. Through this we have applied the simulation methodology to conceive a model and validate it. The next step is to organize a whole structure to make possible the OCS networks performance analysis for several inputs, configurations and algorithms and the further energetic study. Anyway we have already implement two elements that can be used as inputs in our OCS network performance study.

Chapter 4

OCS development

Once we have get familiarized with Javanco and some of the basic functions and properties it is time to implement and construct OCS networks to reach our final objective. For this, we have conceived a coherent modular structure achieving the OCS network simulation tool.

4.1 Main Organization

To develop the OCS technology framework we have created a very flexible tool that permits to get results and evaluate a huge number of combinations considering as parameters as desired.
The very general organization is represented in Figure 4.1. This diagram is an overview of the elements participating in the simulation of an OCS network operation. Through this chapter we will enter in each represented box to describe in detail their content. What is important now is to associate the elements in the diagram with what we have treated since now.

The starting point of everything is the definition of an `DynamicOCSExperiment` which puts together all the elements we want to provide to the OCS network operation.
As we have said, the main parts of an OCS network are the inputs and the configuration or the OCS Model combined with the algorithms to define how the network will work.

To start the simulation we create a `DynamicOCSExperiment` which configures a complete dynamic OCS network. This experiment will receive as parameters: the

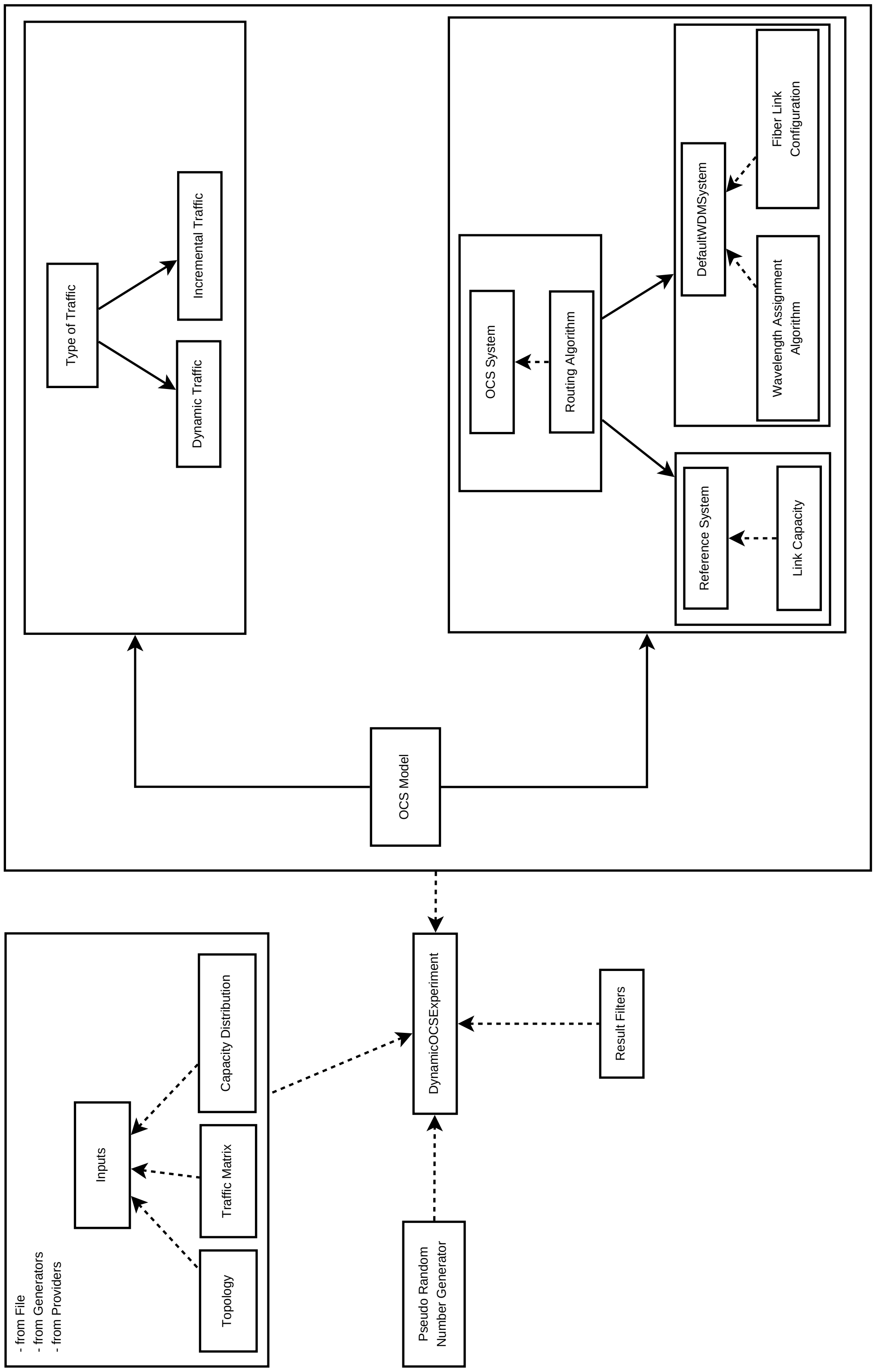


Figure 4.1: Diagram of the OCS Experiment organization

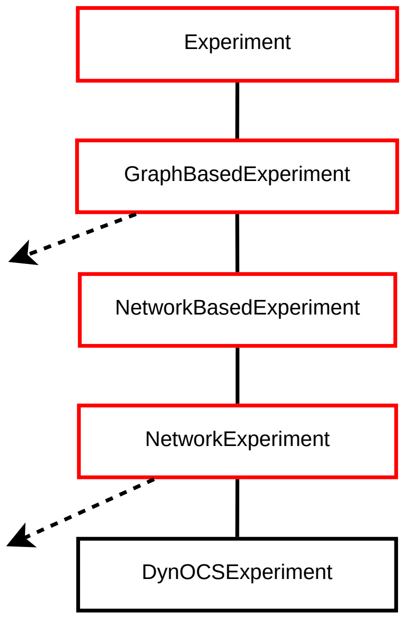


Figure 4.2: Diagram of objects organization for `DynamicOCSExperiment`

definition of the inputs, the configuration of the network, a generator of random numbers and a bank of filters in charge to process and obtain the results of the simulations. The results given by those filters will be the material to reach the final objective: the study of the performance of the OCS networks through comparisons and analysis of several simulation results.
In the diagram 4.1 we recognize the inputs box composed by a topology, a traffic matrix and a link capacity distribution, the OCS model box composed by the way the traffic is generated and the way the demands will be processed defining either the fiber configuration of links or the Routing and Wavelength Assignment algorithms, the Pseudo Random Number Generator useful for the whole system and the filters box to extract the values of our interest.

The central element which is the `DynamicOCSExperiment` comes from the object tree depicted in Figure 4.2.
Our aim from now is to describe in detail each block participating in the OCS Experiment. As said before we have not been able to implement all the different possibilities for OCS networks but we have focused in some of them. We will see which ones.
Each section below describes a branch of the diagram in 4.1 and the considered options. We will not dedicate an entire section to the PRNG because we know that its function is to generate random values for times, demands, etc. Each time we will need a random value at any step or box, this PRNG will be in charge to decide the asked value based on his seed. For that reason we will have to execute the same simulation many times changing the seed. That way we will obtain a higher reliability to the results of our models.

4.2 Inputs

In this section we explain all the elements taking part in the definition of the inputs of our system. We consider then the box in the left superior corner in Figure 4.1 and we represent a more detailed diagram in 4.3. Below there is an explanation of each part of this diagram that we can easily recognize in the diagram.

Actually, the way to define the inputs is through an `NetworkProblemInputCreator` object. This one can receive the input parameters in many ways:
- from a file: all the parameters are determined in an already existing file
- from providers: the `NetworkProblemInputCreator` receives a topology, a traffic matrix and a capacity through providers. Those providers construct the inputs we define and send them as parameters to the input creator. It is the option we commonly use for our simulations.
- from generators: receives a topology, a traffic matrix and a capacity generator (meaning that they have to be constructed).

The first input we define is the **topology**. Topologies can be generated by many already existing generator objects able to construct polygons, tessellations (of squares, triangles or hexagons), or even random topologies among others deciding how many nodes do we want and other parameters that we do not detail here. This topology is translated to an `AbstractGraphHandler` (agh) in order to be manipulated during the experiment. We could as well draw a topology in the graphic interface, save it and send it to the `NetworkProblemInputCreator`. All in all we can generate or create an infinity of topologies based on what we need.

The second input we provide is the **traffic matrix**. This matrix will define for each node pair in the network, the initial traffic rate that will go from the start to the destination node. It is possible to obtain two types of matrix to work with during an OCS simulation.

1. Uniform Traffic Matrix Generator: This object has the ability to provide all connections between node pairs of a given and fixed traffic intensity defined by the float parameter `intensity`. As a result we have an uniform traffic matrix.
2. Distribution Based Traffic Matrix Generator: This object has the ability to provide each connection between node pairs of a traffic intensity based on a

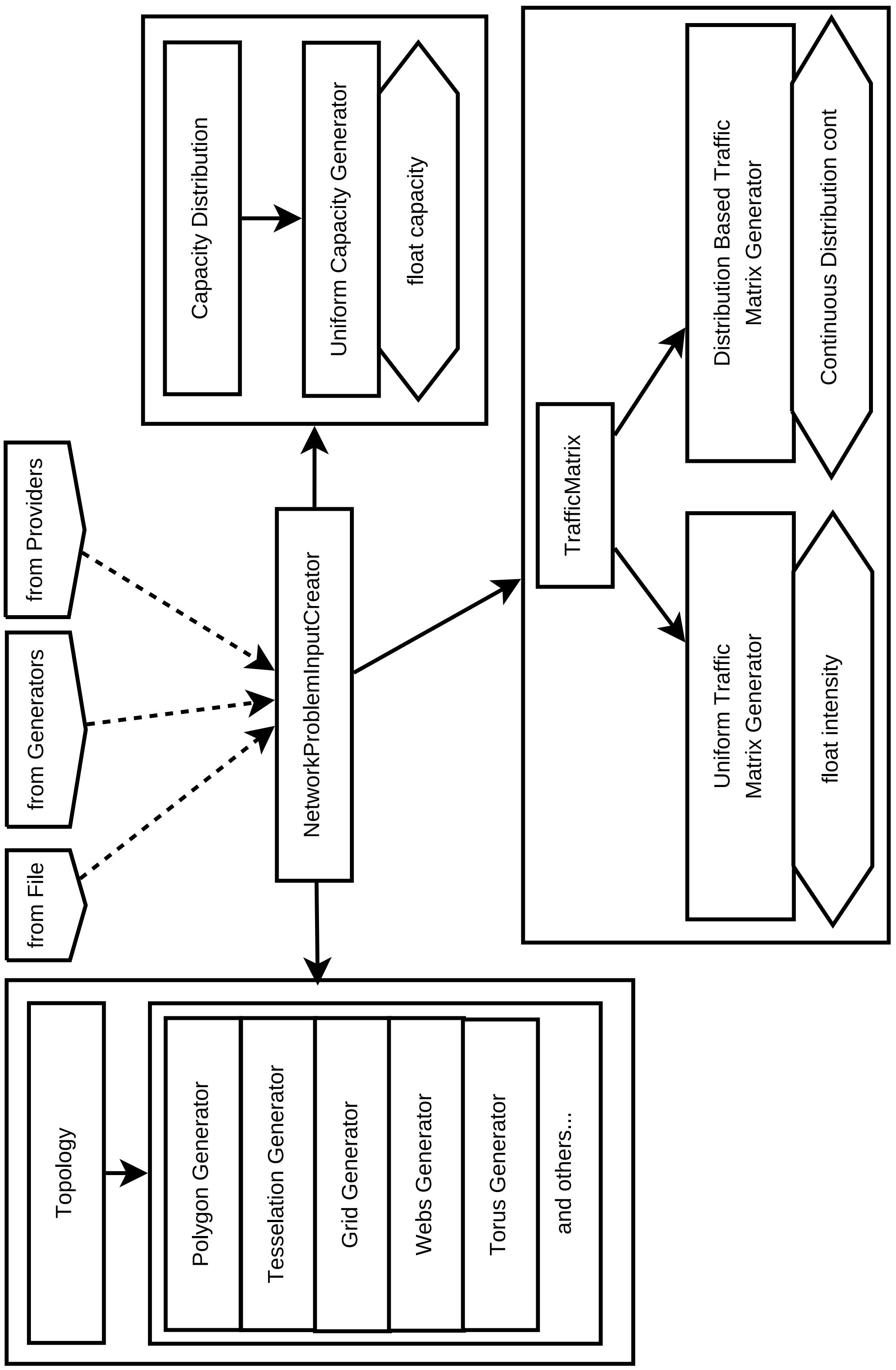


Figure 4.3: Diagram of the inputs organization

given continuous distribution that we will call `cont`. As a result each node pair will have different traffic intensities related by a specific distribution. The continuous distributions are provided by another existing framework developed at the Université de Montréal [56].

Finally, we can select a **default capacity** for each link. In our implementation we have a single object that provides the links of the topology of a *uniform* capacity defined by the float parameter `capacity` (meaning that all links will have the same number of channels in the network). This object does not decide which will be the capacity of the links in itself but defines if links will have the same number of channels or if there will be some links provided with more capacity than others. Indeed we could consider a distance-dependent decision (where the shortest links would be provided of higher capacity) or hub decision (where the links around the central node would have higher capacities) [64].

In general we will use inputs given by providers defining a specific topology, a uniform or distribution based traffic matrix and a capacity which in this case will be always uniform.
Once we have gone through the inputs, we will focus on the OCS Model.

4.3 OCS Model

The OCS Model represented in the box on the right on Figure 4.1 is split in two main parts: the one dealing with the generation of traffic and the one related to the system we have (whether if it is WDM or not). Both participate to the configuration of the network and at the same time there are specified the algorithms used for each configuration.
The object in charge to support configurations and algorithms is called `DynOCSModel` which receives as parameters a traffic generator and an OCS system (Cf. 4.3.1 and 4.3.2 respectively). Although having said that traffic generators constitute an input, here we consider it as a model: depending on which generator is used, the whole network could be considered in many ways (i.e. as if the whole network has a unique server, or if each link of the network operates as a server, etc.).
Through this OCS Model integrated in the experiment we are able to configure the network and to define algorithms with which the OCS network will work. The model is a basic element in our framework to carry out the simulation of the network. We remark the final objective which is to study the performance of OCS networks. This OCS Model will allow us to take into account in the performance study many configurations and algorithms (Cf. sections 7.2 and 7.3).

4.3.1 Traffic Generators

In this section we present the main categories of traffic generators we can find in this framework which corresponds to the upper box in the OCS Model in Figure 4.1. Each type of traffic generator works in a different way. In Figure 4.4 there is the whole organization for the called "Traffic Generator" box in the general structure of the experiment. There are some specific objects that we detailed later inheriting from `ServerAndThresholdPerNodePairEmulatedDynamicTraffic` and `EmulatedDynamicTraffic` (Cf. section 5.1).

For the whole generators we have chosen a **discrete-event models** meaning that the operation of the system is represented as a chronological sequence of events. That is why the traffic generators pend from an abstract class: `AbstractSequentialTraffic`. The simulation of dynamic discrete-event models can be done deploying two techniques of chronological progression:

1. *Fixed Increment Time Progression*: Generally used in the simulation of systems where events are known in advance. The simulation advances by constant amounts of time, called ticks. After each tick we check if some of the scheduled events should occur during the elapsed tick and then they are processed. The whole system state is updated. Unfortunately this method could be inefficient because selecting a short tick period could involve cycles with no useful event work and a larger interval would reduce the event time resolution.
2. *Next-Event Time Progression*: In this case the time of future events are generated and the simulation clock is advanced to the time of the first of these events, which is processed. Then the system is updated and the simulation clock is advanced to the most imminent event. The advantage of this technique is that the periods of inactivity in the system are skipped. This omission of inactive periods allows the simulation to proceed efficiently.

In our developed generators we have dealt basically with *next-event time progression* but in some cases we have provided the system of the possibility to be in a waiting state, but this case will be detailed later (in sections 4.3.1 and 5.1).

From the traffic generators presented in Figure 4.4 we have to make a principal distinction about the way they work. All the traffic generators except the real dynamic traffic do not depend directly on the time although following a chronological order. In the real dynamic traffic either arrivals or ends of demands depend on a continuous distribution that defines for each demand specific arrival and duration times. On the

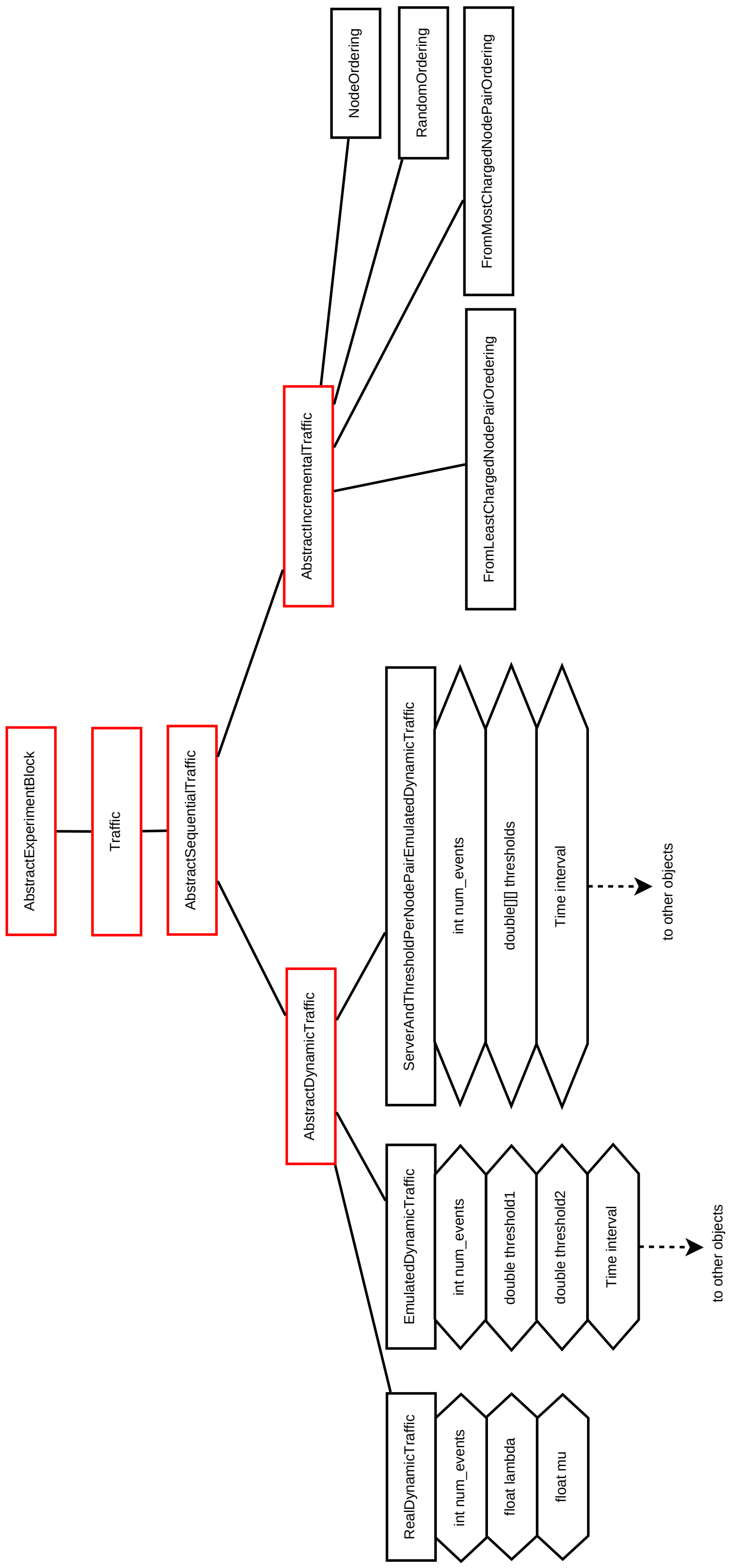


Figure 4.4: Objects organization for the generation of traffic, key in A.1

other hand, the rest of the generators do not base their demand generation on the time: even if each arrival or end of a demand happens in a specific range of time, the resulting action is associated to an entire number corresponding to the number of the time-step we are considering. This difference implicates the definition of different objects for the temporal generators (real dynamic traffic) and the atemporal ones.

Abstract

Before starting the description of the traffic generators in themselves and taking into account the time-dependence in some generators we define the objects we have implemented to integrate in the generator models.

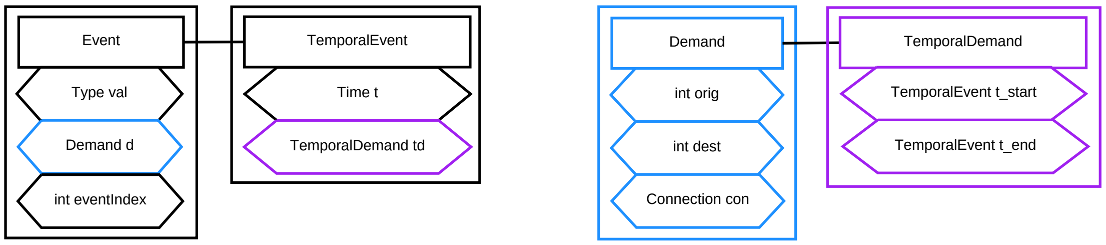


Figure 4.5: Objects organization for the `Event` and `Demand`

In Figure 4.5 there are the objects we mainly used by the traffic generators. The first element on the left is the `Event`. This one receives 3 parameters:
- an objet `Type` that we call `val`. `Type` is an *enum* type and it is a type whose fields consist of a fixed set of constants, particularly in our case: ARRIVAL, END and SIM_END. The parameter `val` defines whether if we have an `Event` which is an arrival of a demand (ARRIVAL), an end (END) or if it is the last demand of the simulation (SIM_END).
- an object `Demand` represented in blue and detailed in the blue box beside. A `Demand` is an object that defines an start node index (int orig), a destination node index (int dest) and a `Connection` object that contains the path (once it has been calculated) of this specific demand.
- an integer indicating the index of the event we are dealing with (or the time-step where this event takes place). But as said before, there are some time dependent generators and this type of `Event` only can be used by atemporal generators. For that reason there is another object inheriting from `Event` called `TemporalEvent` that preserves the `Type` of the event but the `Demand` is a `TemporalDemand`. The index of the event is now an object `Time` that records the specific time where the event takes place. The structure of the `TemporalDemand`, colored in purple in Figure 4.5, inheriting from `Demand`, preserves the start and destination node indexes and also contains the two corresponding `TemporalEvent` to register the initial and end time

of the defined demand.

Now that we know which objects participate in the development of the traffic generator models we explain how they have been implemented.

Even if the whole operation of the experiment will be explained later (Cf. section 4.5) it is interesting to describe here the mechanism of the traffic generators. The main operation of traffic generators is to return each time they are asked to, to return a list of `Event` that should be generated (and then processed by the rest of the experiment) at a certain interval or index of event.

With those explanations we are ready to proceed to the description of the generators.

Real Dynamic Traffic

The Real Dynamic Traffic is a time-depending discrete-event traffic generator. The reason of the creation of this generator is to compare the efficiency in generating traffic between this model depending on time and scheduling in advance the end of the demands with other time-independent models in which events are chosen randomly at each time interval or index.

In Figure 4.4 we see that the Real Dynamic Traffic receives as parameters:
- a maximum number of events
- a lambda parameter which is the mean of the distribution we use to define the time between arrivals
- a mu parameter which is the mean of the distribution we use to define the duration of each demand

When the simulation starts Real Dynamic Traffic creates a `TemporalDemand` with a start and end time chosen based on the chosen distributions for interarrival and duration times with a random node pair. As said, here time is important and that is why we use a `TemporalDemand`. Moreover, each time a temporal demand is created the two corresponding `TemporalEvent` will be registered in a `TreeMap<Time, TemporalEvent>` called `tree`. The `Time` of `tree` is the time corresponding to the associated `TemporalEvent`. Like this, we maintain an automatic ordered list of events by time as a TreeMap sort its entries by the key which in this case is the time of the temporal events. When the simulation runs, the first entry of `tree` is taken (then the first chronological temporal event is chosen). The chosen event will correspond to the n-event of the simulation where n is the index meaning the number of times the loop has been executed. If the considered temporal event is an arrival (knew checking the parameter `Type val` of the event) then another `TemporalDemand` is created and added to `tree` (scheduled with the corresponding interarrival and duration time distribution). Otherwise, if the temporal event is an end, anything else happens. This `TemporalEvent` is added to the list of `Events` to be finally returned to the conduction of the experiment.

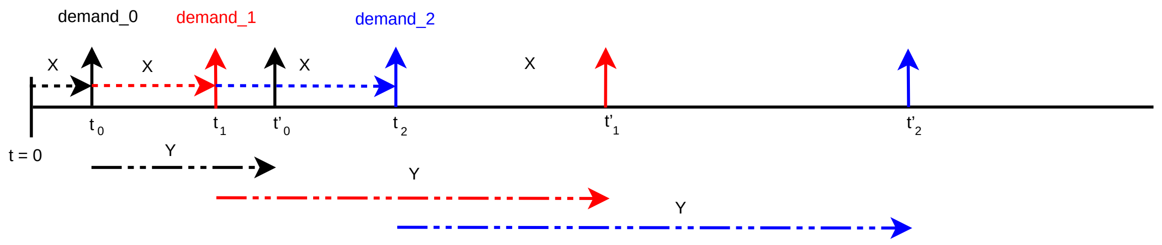


Figure 4.6: Representation of the Real Dynamic Traffic operation

In Figure 4.6 there is a graphic example to show the main idea of the real dynamic traffic. In the case represented, the (temporal)demand_0 would be the first created considering the X and Y defined Random Variables for the interarrival and duration time of demands respectively. The values t_0 and t'_0 would be put in `tree` with the corresponding start and end temporal events. In the first loop the arrival of the demand_0 would be returned and as it is an arrival, the (temporal)demand_1 would be created defining t_1 and t'_1 based on X and Y and sorted in `tree`. The next returned temporal event would be the arrival of the demand_1, and then (temporal)demand_2 is created. The following loop would return the end of the demand_0 so nothing would happen. The same process is done until the last event would be returned.

In this case we maintain a relation with the "real time" because events are related between them by distribution parameters and are not independent among them. We validate this model in section 7.2.1.

Emulated Dynamic Traffic

In this generator, neither arrival time nor end time of any demand are scheduled. The mechanism of this generator is based on thresholds that will define the limits to decide if we will have a new demand arrival, an end of a demand or if nothing happens (waiting state).

The constructor receives as parameters:
- a maximum number of events
- 2 thresholds to take decisions with the condition that

$$threshold1 \leq threshold2 \leq 1 \tag{4.1}$$

- an interval of time in which events will be chosen

Each time the Emulated Dynamic Traffic has to return an event, it generates a random number r between 0 and 1. The decision of which kind of event we will have is taken with the following criteria:

$$\begin{cases} 0 \leqslant r{<}threshold1 \text{ then it is an arrival} \\ threshold1 \leqslant r{<}threshold2 \text{ then it is an end} \\ threshold2 \leqslant r{<}1 \text{ then nothing happens (waiting state)} \end{cases} \tag{4.2}$$

If it is an arrival a random node pair and a random time in the interval of time we are, are chosen. With those elements a `TemporalEvent` is created and returned. The assignation of a time in the interval is a way to define the demand, what is important here is that the index of the event is also registered. We use this index to calculate the traffic offered instead of the time. In this model we focus on the frequency of arrivals and ends of demands and not on the real time distributions. The created demand will be put later in a pending list (the pending list is not directly managed by this object) in order to be ended later, in another execution of the loop.
If it is an end, a demand among the pending list is chosen and returned. If there are not demands in the pending list, then nothing happens as well as if we would had a waiting state.

In Figure 4.7 there is a graphical representation of what happens in the emulated dynamic traffic. The time is split in intervals. At each interval a decision is taken and we can obtain an arrival of a random demand, an end or just nothing. The time in which the event takes place it is not important, we just remark that it is in the n interval for the n event.

In 5.1 the different implemented cases for this traffic generator are detailed.

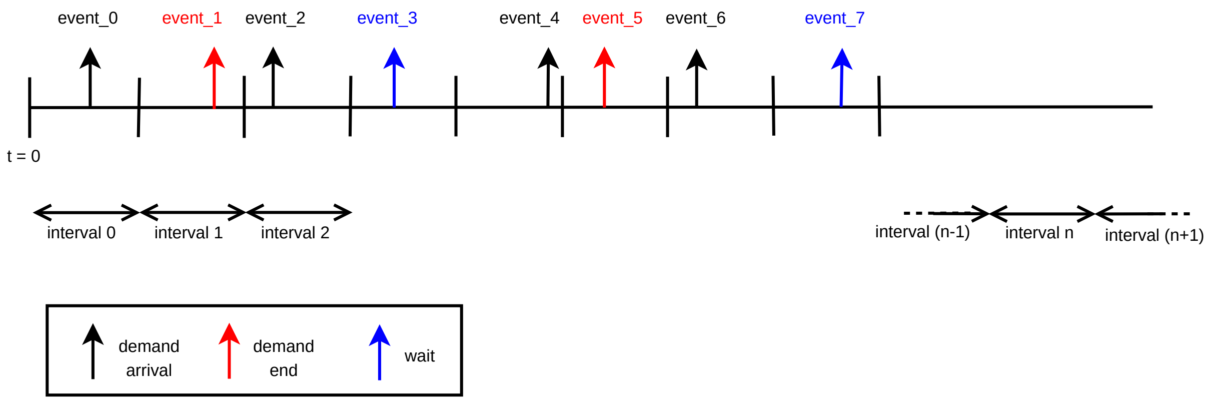


Figure 4.7: Representation of the Emulated Dynamic Traffic operation

Server and Threshold per Node Pair Dynamic Traffic

The main principle of this traffic generator is exactly the same as in the Emulated Dynamic Traffic unless that:
- we receive as parameter a thresholds matrix containing the threshold of decision for each node pair so they could be all differents
- at each execution of the loop all the node pairs pass through the decision process. For each node pair (i, j) a random number $r_{i,j}$ with $i \neq j$ is generated to go through the decision. If we have an arrival, a temporal demand is created for this specific node pair. There is as well a pending list for this specific node pair (again managed by another object). The arrival event of this demand is then added to the list of events to be returned. If we have an end, the pending list of this node pair is checked. If it is empty then nothing happens in this node pair. - due to the fact that there is only one threshold by node pair there is no possible waiting state except if the pending list of the demands of the node pairs is empty.

Depending on the thresholds we choose, this model allows to charge the network with traffic more quickly than the emulated dynamic traffic as this model consider each node pair as a server instead of considering the whole network as a server.
It could be interesting as well to put different thresholds based on parameters of the topology or on the traffic matrix because it would be a way to control the load of the network. This load modulation could participate to the increase of the performance of the network avoiding bottlenecks or waste of resources.

Incremental Traffic

The Incremental Traffic represented on the right of the Figure 4.4 is in charge to represent those demands that arrive, are set up and never leave the network. This type of traffic generator is a good way to observe how the OCS network is getting loaded and little by little new demands are rejected until there is no more free resources. Through this traffic generator we will study which is the best way to process scheduled demands in a traffic matrix to accept the higher number of demands or, what is the same, obtain the lowest blocking probability. This will be obviously a way to judge the performance of OCS networks.

The traffic to be served in an incremental way is given by a previously created traffic matrix (Cf. section 4.2). The order in which demands are processed is an important aspect because depending on the metric on which we base the selection we can obtain different interesting results to analyze. The criterions explained below will be considered in our study:

Random Ordering:

Chooses randomly a node pair to serve from a list containing the node pairs which traffic matrix value is greater than 0. Once the demand is set up, the matrix value is decreased. The process finishes when all the elements of the traffic matrix are 0.

Node Ordering:

Scans the traffic matrix row by row serving (if possible) the demands obtaining a node ordering (from the lowest node index to the highest).

From Most Charged Node Pair Ordering:

Sorts the node pairs from the most to the least charged and serves the demands following this order.

From Least Charged Node Pair Ordering:

Sorts the node pairs from the least to the most charged and serves the demands following this order.

As we want to know how those models work in an OCS network we will make simulations to observe what happen. In section 7.2.3 we can find this study.

4.3.2 OCS System

As explained before, in OCS network configurations we can distinguish if we are dealing with wavelength continuity constraint or not and choose among many routing and wavelength assignment (RWA) algorithms.

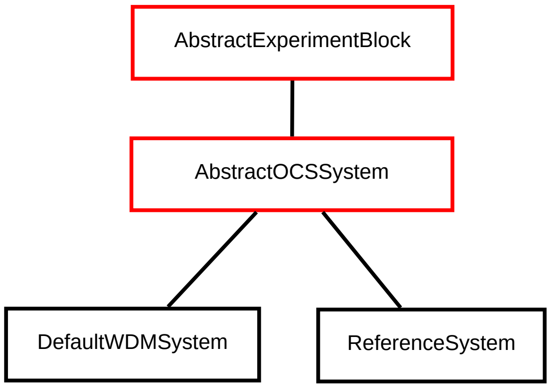


Figure 4.8: Diagram of the objects organization to enable or disable wavelength continuity constraint, key in A.1

It seems that without wavelength continuity constraint networks are more flexible because wavelength assignment is done arbitrarily since there are free wavelengths. On the other hand, there are some RWA that works better than other depending on some variables. In order to study and discuss those ideas we have created the tool that permits to enable or disable the wavelength continuity constraint and to select which routing or wavelength assignment do we want.
On the top of the whole objects organization we have the `AbstractBlockExperiment` once again (Cf. Figure 4.8). The object `AbstractOCSSystem` not only is in charge to define whether there is wavelength continuity constraint or not but it also configures the agh with the provided RWA algorithm.

Taking up the `DynOCSModel`, we represent the detailed structure of the OCS system in Figure 4.9. We recognize in this diagram the OCS system branch in the OCS model and that the starting point is the `AbstractOCSSystem`. This abstract class receives as parameter the routing algorithm: whether we have wavelength continuity constraint or not, the route between a node pairs has always to be decided during the operation.

We distinguish here the implemented routing algorithms that `AbstractOCSSystem` needs as parameter.

Routing Algorithms

The two routing algorithms we have developed are the shortest path and the occupancy depending algorithms. As we see in Figure 4.9 both algorithms pend from an abstract class: `AbstractRoutingAlgo`. And again, the top of the whole collection of object defines an experiment (Figure 4.10).

- Shortest Path Routing Algorithm: This routing algorithm was already implemented but this specific object collects the results of the `JavancoShortestPath` based on the "length" attribute and configures the agh of the given topology with this results. No parameter is needed in this case.
- Occupancy Depending Routing Algorithm: This object receives a double (called `capacity` in our case) between 0 and 1 that indicates the maximum accepted occupancy in a link. If a link surpasses the given percentage of occupancy then another path has to be selected. The algorithm that returns the path between a start node and destination node with this routing is:

Algorithm 1 Occupancy Depending Routing Algorithm

```
Require: Maximum occupancy $max\_occ \in [0, ..., 1]$
Ensure: The path p of the route between start and dest
  Get the shortest path p
  if p is null then
    return null
  while At least one of the links of p has $link\_occupancy\_percent > max\_occ$ do
    disable the first surcharged link
    recalculate the path p
    if p is null then
      return null
  return p
```

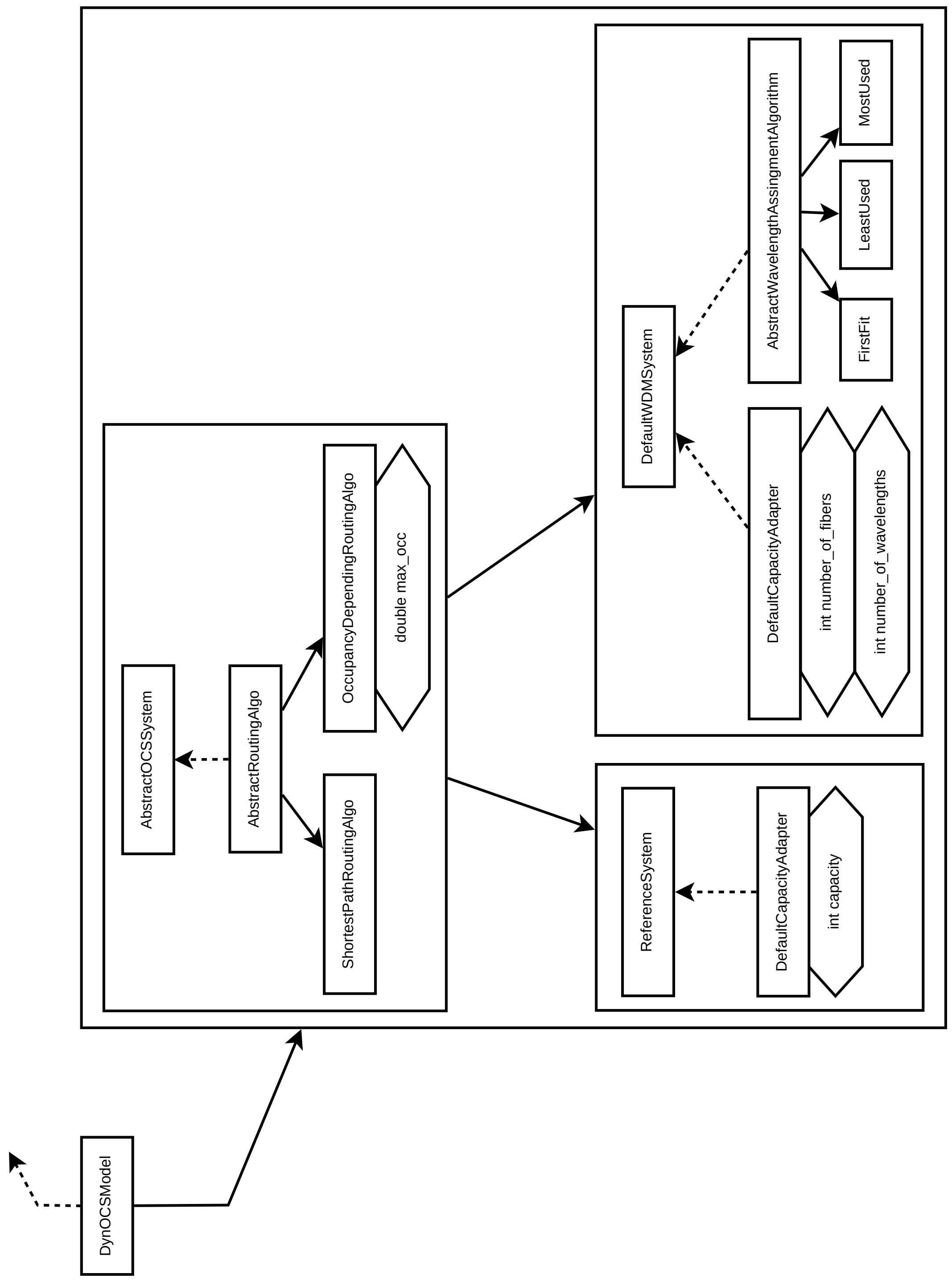


Figure 4.9: Diagram of the OCS system organization

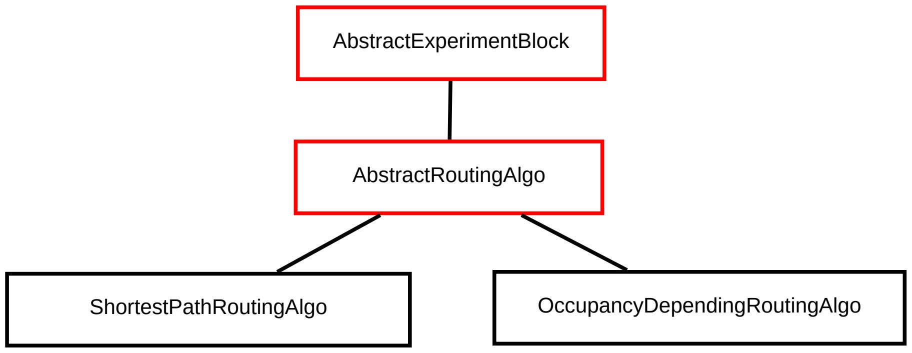


Figure 4.10: Diagram of the objects organization for routing algorithms, key in A.1

> In a following version this routing algorithm could be adaptive and change automatically the maximum occupancy based on the occupancy balance of the network or have a different thresholds of maximum occupancy for each link.

Once the routing of the OCS system is decided we focus on whether the network has wavelength continuity constraint or not. To separate clearly both option we have on one hand (Cf. Figure 4.9) there is `ReferenceSystem`, this one disables the wavelength continuity constraint. On the other hand there is `DefaultOCSSystem`, this one enable this ability.

`ReferenceSystem`

This object receives a routing algorithm to define the `AbstractOCSSystem` and a `DefaultCapacityAdapter` with certain integer capacity to be assigned to each link in the network. `DefaultCapacityAdapter` is in charge to configure the links of the network as desired. As this object does not take into account the wavelength continuity constraint, it is not necessary to define how many fibers and wavelengths do we need but only provide links of a finite capacity. To proceed to the configuration of the network, each `LinkContainer` of the agh is furnished of a `FiniteLinkCapacity` with the capacity given as parameter through the `DefaultCapacityAdapter`. Like this, there is no distinction between wavelengths but at each time a wavelength is needed, the remaining capacity in the link is decreased. When a lightpath is released, the remaining capacity of each link is increased. Thus, with this model, we do not have wavelength continuity constraint.

The last branch in the diagram 4.9 defines the `DefaultOCSSystem`.

`DefaultOCSSystem`

In contrast to the previous object, this implemented model enables wavelength continuity constraint. The received parameters are a routing algorithm to define the `AbstractOCSSystem`, a wavelength assignment algorithm and a `DefaultCapacityAdapter` defining the wanted capacity split in number of fibers and number of wavelengths per fiber. Thanks to the Java language we can use the same object to define different things depending on the parameters it receives. In this case, we choose two integers that `DefaultCapacityAdapter` will recognize as number of wavelengths and number of fibers (in the previous case, the fact to define one single parameter activates another completely different configuration of the network). So in this case, when a lightpath has to be set up, a concrete wavelength in a concrete fiber is chosen based on the given RWA algorithm. The same set up lightpath is released when the demand finishes.

There are still two important things to explain: how the fiber-optic links configuration is achieved and which could be the wavelength assignment algorithms.

Fiber Network Configuration:

The principal characteristic of an OCS network is that it works with Wavelength Division Multiplexing. In consequence, our goal is to furnish the links of a given physical topology, of a fiber link configuration. The main organization to achieve this configuration is represented in Figure 4.11 and each element is detailed below.

- `Wavelength`: Is the smallest unity in this system and is characterized by an own number. Has pointers to the `Fiber` and `LightPathLink` where it is contained. To know if a specific wavelength is free we check if the pointer to the `LightPathLink` is null, if it is, then it means that this wavelength is free for in that light path link because it has not been assigned yet to a lightpath.
- `Fiber`: Characterized by an own number . Has an array of pointers to all the contained `Wavelength` and a pointer to the `FiberLink` where it is contained.
- `FiberLink`: Defines a single optical link with a number of fibers and the number of wavelengths in each fiber. Has an array of pointers to all the contained `Fiber`.
- `LightPathLink`: Has a pointer to the used `Wavelength` in this specific link of the path and to the `LightPath` where it is contained. It is the media to register which wavelength and fibers of the links traversed by the lightpath has been used.

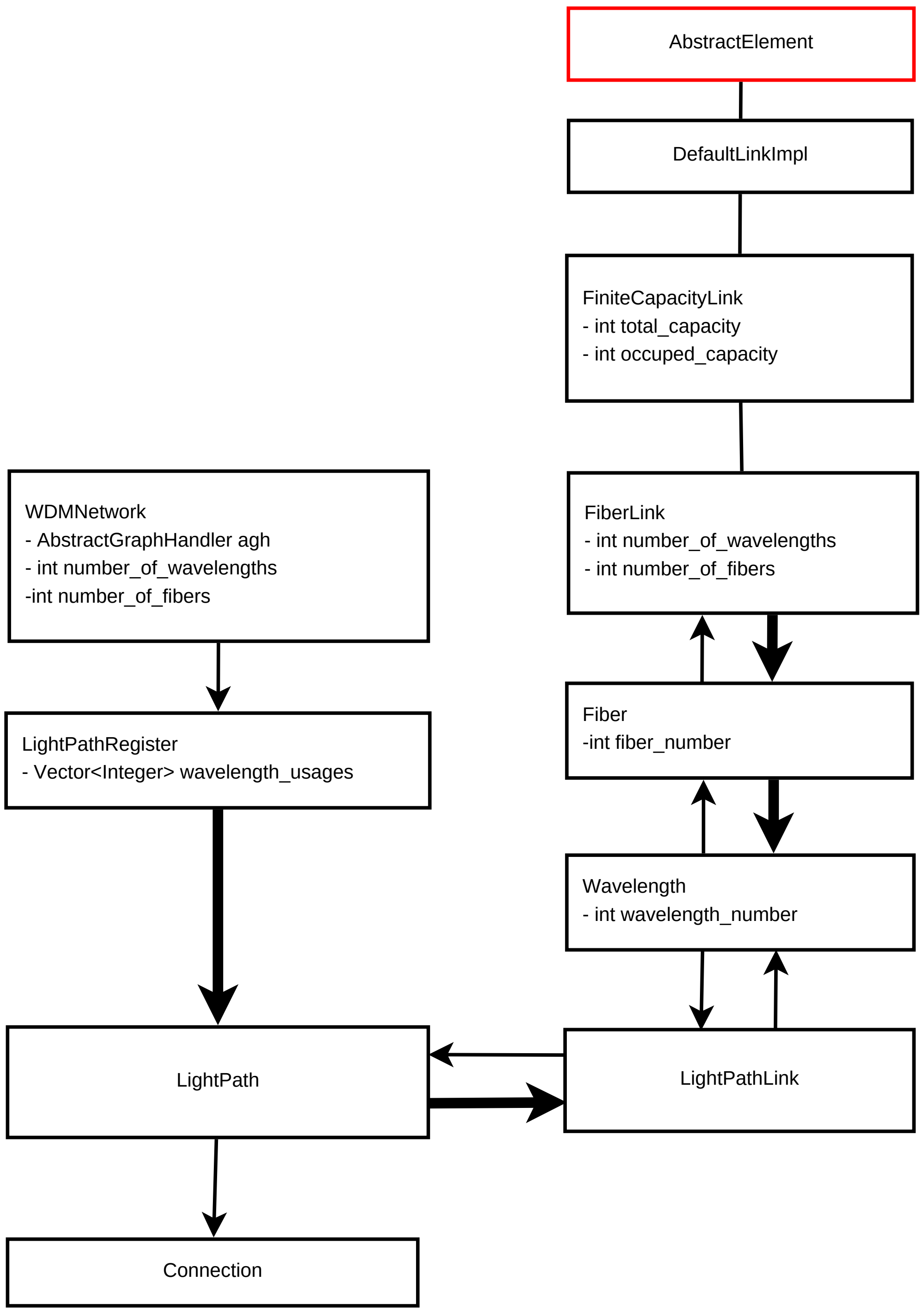


Figure 4.11: Diagram of the objects organization for an optical network configuration and their attributes, key in A.1

- `LightPath`: Has an array of pointers to all the contained `LightPathLink`. This object extends from `Connection` that defines the path.
- `LightPathRegister`: Has an array of pointers to all the contained `LightPath` and counts how many times wavelengths have been used (this will be useful for those Wavelength Assignment algorithms that take into account the wavelength usage) and which lightpaths have been set up.
- `WDMNetwork`: The constructor of this object is in charge to set up the optical configuration to each link for a given agh, number of fibers and number of wavelengths per fiber. It has a pointer to the `LightPathregister`

To link it directly with the framework we are constructing we have to place it in the `DefaultCapacityAdapter` in the `DefaultWDMSystem` of the Figure 4.9. As this capacity adapter receives a number of fibers and wavelengths, the configuration is made sending this parameters to a `WDMNetwork` which will use the agh and the fibers and wavelength quantities to configure our network.

To follow, we describe the wavelength assignment algorithms that has been considered in our work and that we can use to configure the OCS network.

Wavelength Assignment Algorithms:

Here we consider three wavelength assignment algorithms. When we mix one of them with a routing algorithm we get a whole RWA algorithm that can be tested and simulated in whichever topology.
The object tree concerning the wavelength assignment is represented in 4.12. Obviously, the root is again the `AbstractExperimentBlock` defining a new block of objects to experiment with.

All those heuristic algorithms only have sense if wavelength continuity constraint is considered so in the `DefaultWDMSystem`. Otherwise, a random free wavelength in each link could be chosen independently of his number.

- First Fit: The First Fit algorithm (Algorithm 2) looks for the shortest path in the network and assigns to this path the first free wavelength among all the fibers in the network.

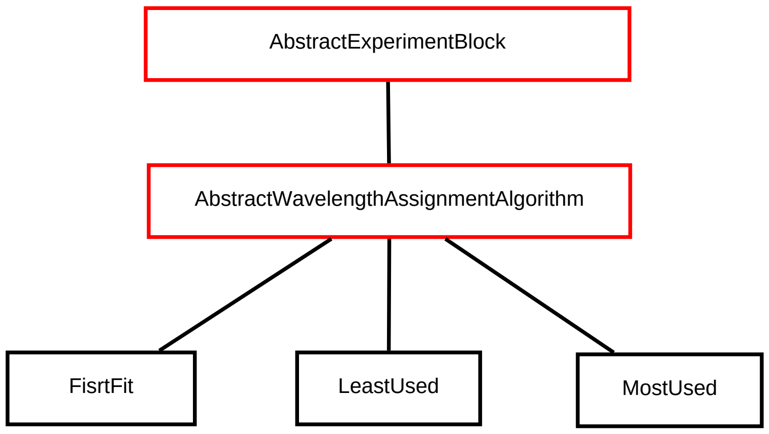


Figure 4.12: Diagram of the objects organization for wavelength assignment algorithms, key in A.1

Algorithm 2 First Fit Wavelength Assignment Algorithm

```
Ensure: The first free wavelength for the path p in the WDM network
  for $i = 0$ to $i < max\_number\_of\_wavelengths$ do
    if wavelength $i$ is available in path p then
      return $i$
  return -1
```

Algorithm 3 Most Used Wavelength Assignment Algorithm

```
Ensure: The most used free wavelength for the path p in the WDM network
  List and sort the wavelength usages (with a reference to the number of wavelength)
  in the actual network in a list called used
  if the size of used is $> 0$ then
    for the values of used taken from the top to the bottom do
      if the actual wavelength related to the value is available in the path p then
        return number of wavelength in value
  return -1
```

- Most Used: The Most Used algorithm (Algorithm 3) looks for the shortest path in the network and assigns to this path the most used wavelength in the whole network
- Least Used: The Least Used algorithm (Algorithm 4) looks for the shortest path in the network and assigns to this path the least used wavelength in the whole network.

Algorithm 4 Least Used Wavelength Assignment Algorithm

```
Ensure: The least used free wavelength for the path p in the WDM network
  List and sort the wavelength usages (with a reference to the number of wavelength)
  in the actual network in a list called used
  if the size of used is > 0 then
    for the values of used taken from the bottom to the top do
      if the actual wavelength related to the value is available in the path p then
        return number of wavelength in value
  return -1
```

Those algorithms seems to be very close the ones to the others but in complex networks it happens that they give results. As a first impression we can say that, Least Used tries to balance the load over the wavelength set but it disfavors long demands that would need the same wavelength within the path. On the other hand Most Used and First Fit try to maximize the utilization of available wavelengths but overloads rapidly some wavelengths.

This would be the last block to explain in the implemented OCS model. Next we will finish the description of the OCS experiment explaining how the tools to collect results work.

4.4 Result Filters

To get results of what happened during the execution we use some filters represented in the box "Result Filters" in Figure 4.1. With that goal we use a `DynOCSResults` which receives the whole experiment and pass it through a list of specific filters that we will add in charge of counting or processing some variables of our interest. There are 3 available filters attached to an abstract class (Figure 4.13)

The function of each of those filters is:

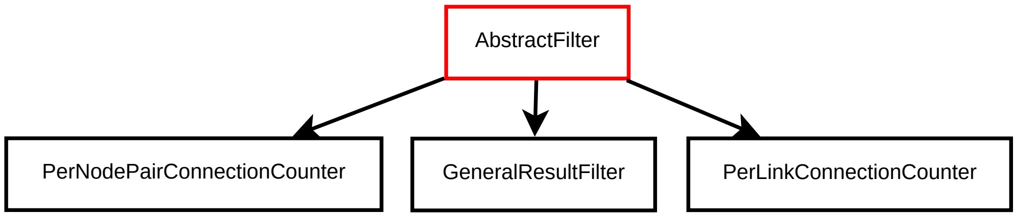


Figure 4.13: Objects organization for the OCS result filters, key in A.1

- `GeneralResultFilter`: Counts the accepted, refused and released connections. When the simulation finishes, it calculates the estimated offered traffic (equation 4.3), the acceptance ratio (equation 4.4, the maximum number of demands at the same time in the system and duration time statistics.

$$T_{off} = \frac{\text{(total number of demands)} \cdot \text{(mean duration of accepted demands)}}{\text{total capacity in the network} \cdot \text{simulation time}} \tag{4.3}$$

$$A = \frac{\text{accepted demands}}{\text{accepted demands} + \text{refused demands}} \tag{4.4}$$

- `PerLinkConnectionCounter`: Counts how many times physical links are utilized during the simulation. It registers a reference (a time or a number of event), the number of connections at this precise reference and if the value is a maximum, it is also registered.

- `PerNodePairConnectionCounter`: Counts how many times node pairs are utilized.

Once we know what is the role of each filter we can add them to a list of `AbstractFilters` (Figure 4.14) enclosed with the `DynOCSExperiment` that will give the information to those filters at some check points as the execution runs.

The filters have as well the capacity to register all the information (parameters and results) and represent it in a graphical interface. Those obtained graphics will be the basic tool to study and compare what happens in OCS networks with different settings and to extract conclusions about the main important point in this work, the performance of those frameworks.

In Figure 4.15 there is schematically the general procedure to create and define an OCS network model in order to be simulated. Each detail in every point has been treated in the sections above. We highlight that this structure is very flexible because we are able to add or change things as desired using all the implemented inputs,

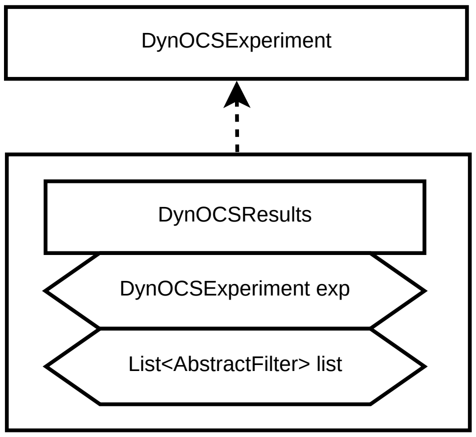


Figure 4.14: Diagram of the OCS filters configuration

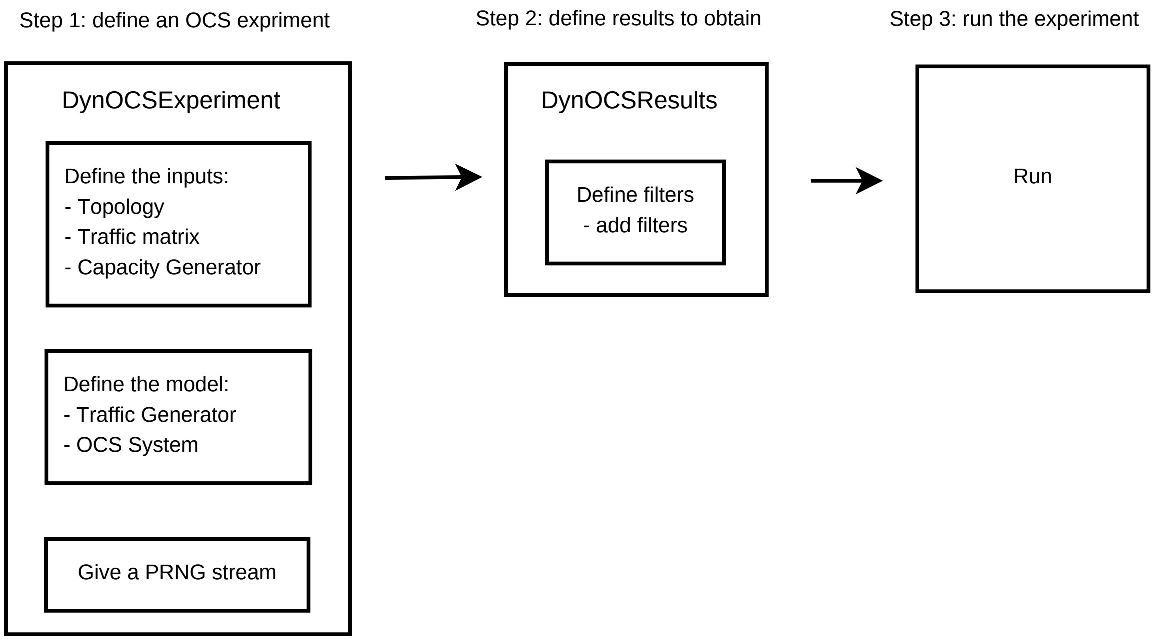


Figure 4.15: OCS network definition procedure

configurations and algorithms. It is feasible to add, modify or eliminate anything we want. Furthermore, if we want to provide the OCS network of an extra configuration as protection, grooming or other ways to route (among others) we just have to implement and add them to the OCS model definition.
We have achieved a modular structure able to get adapted to any input, configuration or algorithm. This was the main goal we wanted to accomplish.

As we see in Figure 4.15 there is a step that we have not dealt with yet which is the run of the experiment step. This step starts the execution of the model with the configured inputs, configurations and algorithms. We shortly describe below how the execution is carried out.

4.5 Experiment Execution

Finally we have completed the explanation of how the whole system is created and configured and we have seen all the different possibilities we can consider to create a `DynOCSExperiment` and its results. But, how does this system work? which are the steps of the execution of the main experiment?

The very first step is to create and configure the OCS network experiment with the combination of inputs, models and filters we would like to test and simulate in a main method. We can create lists of objects to be considered in a collection of simulations. Those elements in the network can be executed doing `for` loops considering all the objects of the lists. The limitation is the memory of the graphical interface and the computing power of the computer. Once we have defined everything we proceed to run the experiment.
Supposing that we have a `DynOCSExperiment` called "exp" with the inputs and model we want and a `DynOCSResults` called "res" we execute the instruction `exp.run(res)`. This instruction conducts us to the highest class in the experiment tree (Cf. Figure 4.2) called `Experiment` that launches the operation of the experiment through a method `conductExperiment(Experiment exp)` placed in the `DynOCSModel` that returns `DynOCSResults`. And the simulation begins.
First we check which kind of traffic generator we are using because depending on it some things are susceptible to change:
- Real Dynamic Traffic will have to define that we will work with `Time` variables and not with `integers` as in Emulated Dynamic Traffic. Know this is important especially for the filters when they will have to calculate the traffic offered (in Real Traffic we make the addition of times and in Emulated Traffic, we count how many events have passed between from the start to the end of a demand)
- for Incremental Traffic we disable the pending lists as demands never end

We then start an infinite loop that will be broken once the last event, the one with the SIM_END val, arrives and return all the obtained results during the execution.
The first instruction of the loop is the call to the traffic generator to return the list of `Event` at this specific index of event or time interval. The explanation of how the events are chosen and returned by the different implemented traffic generators can be found in section 4.3.1.
Once having the list of events (or temporal events) we process them one by one in a `for` loop until the list is empty. Then we will be able to advance to the next step in the loop.
So we take the `Events` as they are ordered in the list and we check the value `Type` to know if it is whether an arrival or an end.

If it is an arrival then we resort to the OCS system (Cf. section 4.3.2) to ask for a route and eventually a wavelength based on the chosen Routing and Wavelength Algorithm (RWA). Once there are available resources we can set up the lightpath and mark the used fiber and wavelengths (o used capacity). Finally, the result filters count the accepted demand of the event (apart from other useful variables for their function) and, if the traffic generator is not incremental, we add the event to the corresponding pending list. It is also possible that there are not enough resources to allocate this incoming demand and it has to be refused. Then, of course, it is not added to the pending list (and that is why we manage the pending list here, because we have to know if they have been set up or not to add demands) and the result filters are informed of this rejection.
If it is an end we are sure that the associated demand was previously set up (thanks to the pending lists or the scheduled organization of the Real Dynamic Traffic) so, through the OCS System, we will remove this demand (or what is the same, release the resources used by the lightpath). We communicate to the filters the release.
We repeat this loop until the last event arrive. Before returning the bank of filters with the results, those ones are impelled to calculate the final values taking into account the information compiled during the execution obtaining the blocking probabilities, the traffic offered or the maximum number of accepted demands.
Once we are again in the `DynOCSExperiment` with all the results of the execution we just have to store all the information in the graphical interface.
Now we are ready to begin a new execution with another experiment inputs or configurations. Results will be registered and accumulated until the last simulation is finished. Then, all the result values will be displayed.

Until here we know which tools we have, which are their functions and why it could be useful to study them. We know as well how to put everything together to be able to configure an experiment about OCS networks. This is what we call a framework.
Moreover, we are capable to simulate those elements as a whole forming an experiment

and, the most important thing is that we can obtain graphical output data to analyze, compare and extract information about the performance of the network.
Other complementary tools are explained in the next 2 chapters.

Chapter 5

Traffic offered problematic

This section deals with various types of emulated traffic generators leading in a traffic offered study.

5.1 Emulated Traffic Generator Models

We have already considered many different kinds of traffic generators models (Cf. section 4.3.1). Here we want to enter in detail and study which kind of Emulated Dynamic Traffic we can imagine. On one hand it is interesting to observe how those models work based on their characteristics and on the other hand to find a way to modulate the traffic offered depending on the traffic generator we are dealing with (Cf. section 7.3).
The Figure 5.1 [1] represents the types of Emulated Dynamic Traffic we will describe. This representation is the easiest way to take a general perspective of how those traffic generator models are conceived. In the first stage there are only the principal models (M/M/1 and M/M/∞) working at network scale, in the second level we have a server (or servers in M/M/∞) for each node pair (we are in a node pair scale). The third level is more heterogeneous and allows different decision thresholds per node pair. Until here the properties have been added level by level to the models. Besides, the last level do not consider a threshold per node pair but a possible waiting state or the ability to accept or release 2 demands in the same interval.
Through this models we will gain other tools to study the OCS networks operation and in consequence, its performance. The used notation for the emulated dynamic models is detailed in Annex C.

[1] the ∞ symbol is represented as 8 (we cannot represent the symbol in the drawing software)

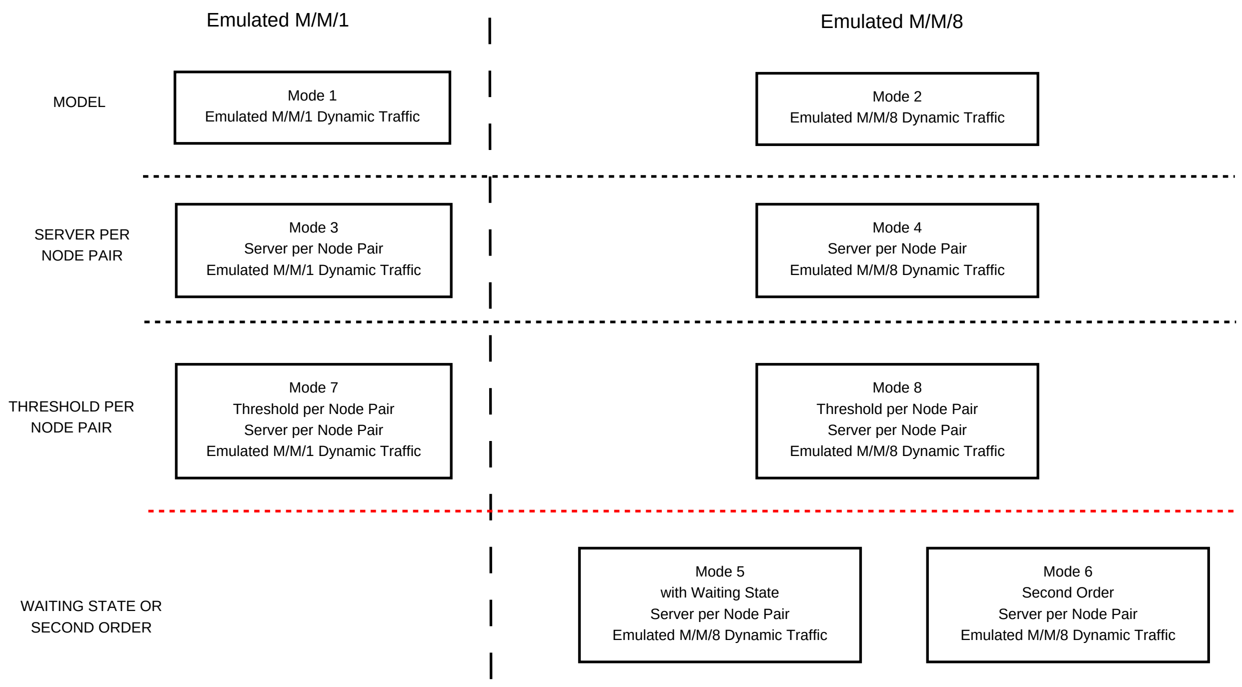


Figure 5.1: Emulated Models organized by characteristics

5.1.1 Mode 1, Emulated M/M/1 Dynamic Traffic:

It is the simplest mode: one server and an infinite queue for the whole network. It is based on a birth-death Markov process with constant birth and death rates: λ and μ. To model this generator we take the `EmulatedDynamicTraffic` already implemented (section 4.3.1). The equivalent of the birth rate λ would be $threshold1$ and for the death rate μ it would be $threshold2 - threshold1$. This model do not accept waiting states then we should impose another restriction which is that $\lambda + \mu = 1$ or, what is the same, $threshold2 = 1$. Indeed, threshold1 and threshold2 would mean the probability to have a demand arrival or end which is perfectly compatible with the fact that in certain interval of time there are λ arrivals and μ served demands.
The whole network could be represented as in the Figure 5.2.

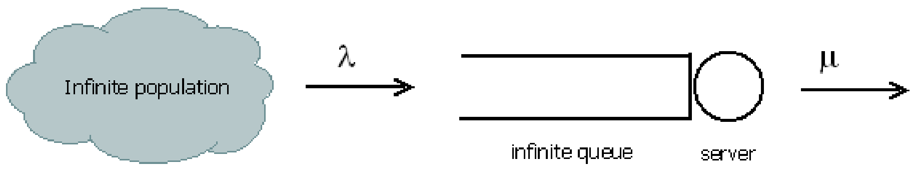


Figure 5.2: M/M/1 system

To prove that the model works we try to find graphically the curve of the mean number of elements in the system (or the mean load of the network) which theoreti-

λ	μ	ρ	theoretical N value	simulated N value	confidence interval (95%)
0.05	0.95	0.053	0.055	0.064	[0.055, 0.071]
0.10	0.90	0.111	0.125	0.117	[0.105, 0.128]
0.20	0.80	0.250	0.333	0.340	[0.318, 0.361]
0.30	0.70	0.429	0.75	0.729	[0.694, 0.765]
0.40	0.60	0.667	2.00	1.988	[1.912, 2.064]
0.45	0.55	0.818	4.50	4.449	[4.296, 4.603]

Table 5.1: Theoretical and simulated values for the mean number of elements in a M/M/1 system

cally corresponds to $N = \frac{\rho}{1-\rho}$ in respect of ρ defined as $\rho = \frac{\lambda}{\mu}$ (full demonstration in C.1). The value of ρ is known because we fix λ and μ (or threshold1, threshold2 is not necessary because its value is 1) and we can calculate it: $\rho = \frac{\lambda}{\mu} = \frac{\text{threshold1}}{1-threshold1}$. Through this expression we see that:

$$\lim_{\rho\to 1} N = \lim_{\rho\to 1} \frac{\rho}{1-\rho} = \infty \tag{5.1}$$

Then if we want a finite number of elements we have to impose ρ<1 or, what is the same, λ<μ. This result make us understand that the traffic offered will never be higher than 1 since the birth rate is smaller than the death rate and all the entering traffic in the network will be served.

If we consider that $\lambda \in [0, ..., 1]$ that $\mu = 1 - \lambda$ and that λ<μ we will simulate first a range for λ from 0 to 0.45.

In Figure 5.3 [2] we see graphically the output data of the simulation for 50000 events, for 7 different seeds and for a 2 nodes polygon. In the x axis there is the arrival probability (λ) and in the y axis the mean number of elements in the system. There are represented both curves: theoretical and simulated values for the mean number of elements, called "Total_elements" in the figure. We report those values in Table 5.1) and we add the confidence interval. We observe that all the theoretical values are contained in the confidence interval of 95%. Then we can accept the model as valid.

In the Figure 5.4 we obtain the simulation results of λ values theoretically impossible, thus λ>0.5. The solid red line is the mean value of the simulation. From an arrival rate superior than 0.5 the mean number of elements increases linearly. When

[2]In the graphic key it is said "Emulated Dynamic Traffic" but this corresponds to the Emulated M/M/1 system

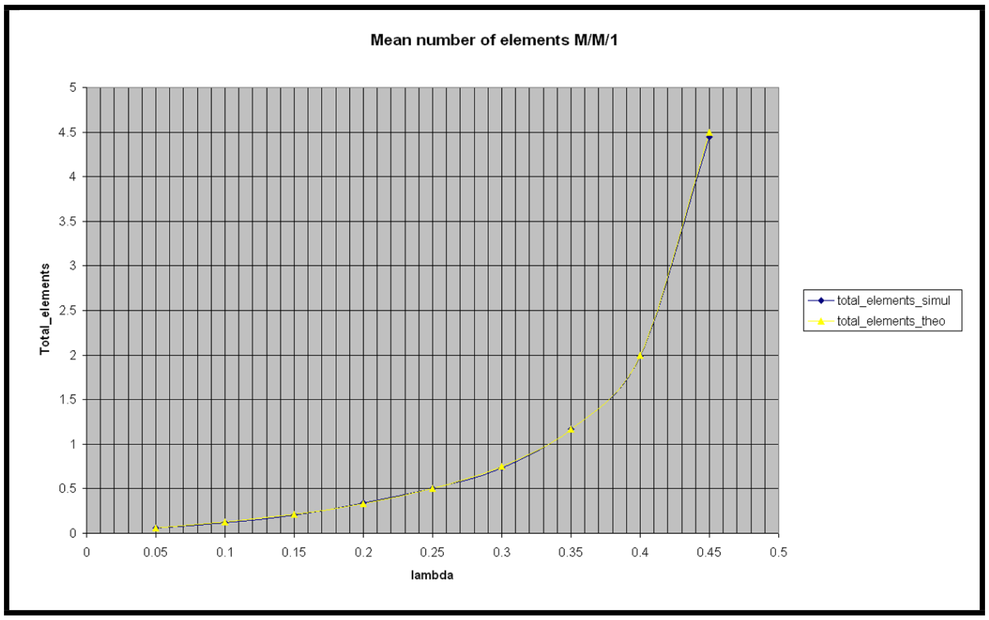


Figure 5.3: Mean number of elements in a emulated M/M/1 system simulation results with arrival rate $0.05 \leqslant \lambda \leqslant 0.45$

λ reaches 1 then everything becomes constant because it is the maximum mean value the simulation allows and all the events are arrivals since then.

5.1.2 Mode 2, Emulated M/M/∞ Dynamic Traffic:

This object models the whole network as a single system with infinite servers as shown in Figure 5.5.

This configuration permits to modulate the offered traffic. The more elements are in the system, the lower is the probability to have a new arrival. This generator corresponds to a birth-death process as well but this time the death rate is increased each time there is a new arrival. The Markov process for this system is represented in Figure 5.6

As in the previous system a random number and a random node pair are generated but in this case the arrival and end thresholds are modified to be adapted to accomplish the 5.6 process. We make a little explanation:
We suppose at the beginning, then for the state 0, that we have $\lambda = 0.4$ and $\mu = 0.6$. The addition is 1, everything works. If now we have an arrival event we will go to the state 1 but the rates still be the same. But if now the network receives another

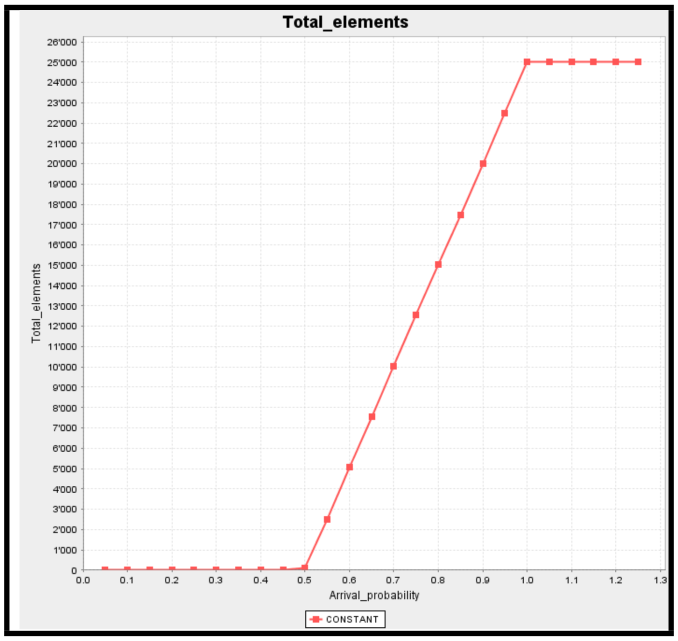


Figure 5.4: Mean number of elements in a emulated M/M/1 system simulation results with arrival rate $0.05 \leqslant \lambda \leqslant 1$

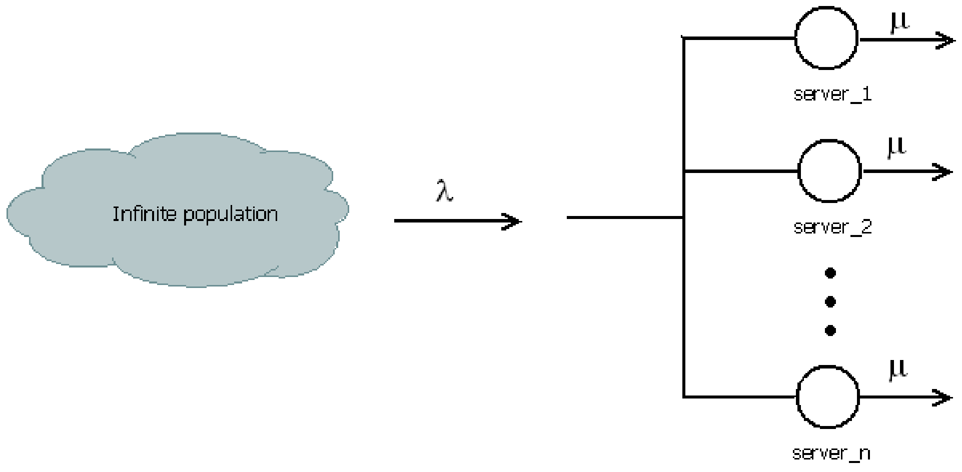


Figure 5.5: M/M/∞ system

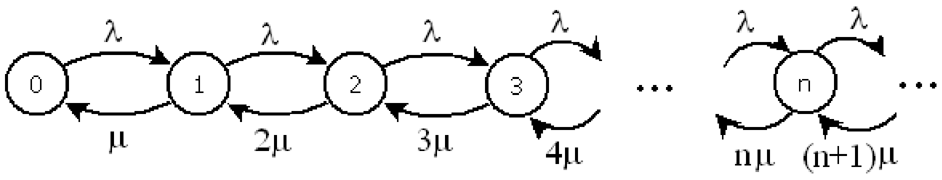


Figure 5.6: Markov process for M/M/∞ system

λ	μ	theoretical N value	simulated N value	confidence interval (95%)
0.05	0.95	0.053	0.057	[0.053, 0.071]
0.10	0.90	0.111	0.117	[0.108, 0.134]
0.20	0.80	0.250	0.295	[0.286, 0.327]
0.30	0.70	0.429	0.529	[0.500, 0.555]
0.40	0.60	0.667	0.855	[0.793, 0.863]
0.50	0.50	1.0	1.252	[1.207, 1.290]
0.60	0.40	1.5	1.855	[1.845, 1.945]
0.70	0.30	2.333	2.737	[2.650, 2.294]
0.80	0.20	4.0	4.452	[4.377, 4.524]
0.90	0.10	9.0	9.395	[9.299, 9.521]
0.95	0.05	19.0	19.370	[19.327, 19.650]

Table 5.2: Mean number of elements in a emulated M/M/∞ system simulation results

demand that would be served by the second server, then the rate to go to state 1 would be $2\mu = 1.2$ and this do not has sense because it is greater than 1 and it does not correspond to a probability anymore. What we have to do to make things right is to normalize λ and 2μ to the new situation. Then

$$\lambda' = \frac{\lambda}{\lambda + 2\mu} = \frac{0.4}{0.4 + 1.2} = 0.25 \tag{5.2}$$

and

$$\mu' = \frac{2\mu}{\lambda + 2\mu} = \frac{1.2}{1.6} = 0.75 \tag{5.3}$$

and as supposed

$$\lambda' + \mu' = 1 \tag{5.4}$$

This normalization has to be done at each stage and in general we will obtain, supposing n the state in which we are (and except when $n = 0$):

$$\lambda_n = \frac{\lambda}{\lambda + n\mu} \tag{5.5}$$

and

$$\mu_n = \frac{n\mu}{\lambda + n\mu} \tag{5.6}$$

To prove that this system corresponds to the real model we represent graphically the mean number of elements in the system (as in the last case). This time the theoretical result has to be $N = \frac{\lambda}{\mu}$ (demonstrated in C.2.

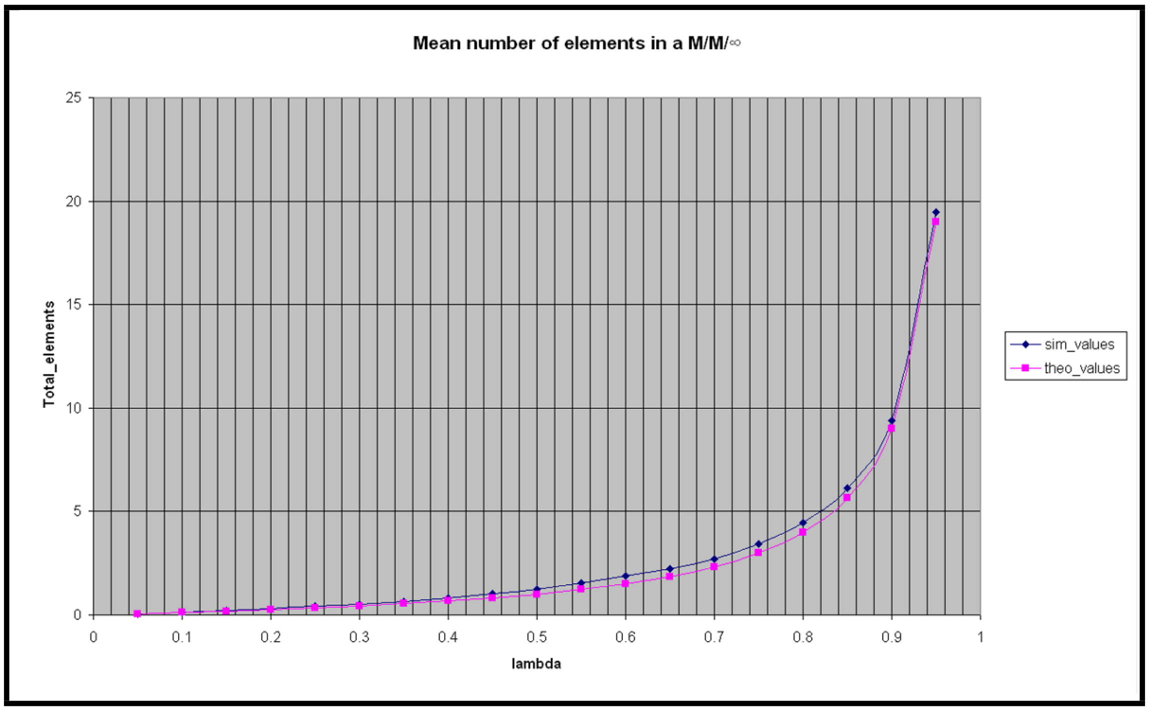


Figure 5.7: Mean number of elements in a emulated M/M/∞ system simulation results

Assuming the same conditions as in the M/M/1 case but choosing any $\lambda \in [0, ..., 1]$ the graphical results of the simulation are shown in Figure 5.7. Then this curve correspond to a M/M/∞ system.

We find the theoretical and simulated results in Table 5.2 with the corresponding confidence interval of 95%. This time results of the simulation are not so close to the theoretical ones. In the comparison of the curves in Figure 5.7 we observe how from the λ value of 0.35, curves start to get separated. The graphic observation corresponds to the values given in Table 5.2 because from $\lambda = 0.40$ the theoretical values are not in the confidence interval. We will suppose that errors are due to imprecisions during the simulation and we will validate the model to continue the work. Anyway, even if values are not the same, the tendency is similar (growing through simulation a little bit faster).
We can also take advantage of the fact that results are good enough for λ values lower than 0.4 to compare M/M/1 with M/M/∞. The comparison of systems corroborates that M/M/∞ maintains in a better way the control of the mean number of elements in the system: M/M/∞ permits a higher arrival rate without the network getting overcharged.

5.1.3 Mode 3, Server per Node Pair Emulated M/M/1 Dynamic Traffic:

The principle in this case is the same as in the Mode 1 but here we provide each node pair with a server. This operation allow us to multiply the traffic offered by the number of node pairs in the network.
The simulation in this case consist on fix two thresholds and generate a random number for each node pair in the network. As a result the object returns a list of `Event` containing an event for each node pair in the network. Like this we speed up the generation of traffic in the same range of time. In this case the traffic offered is multiplied by the number of node pairs but still it remains limited.

5.1.4 Mode 4, Server per Node Pair Emulated M/M/∞ Dynamic Traffic:

This mode follows the same idea than Mode 3 but for each node pair. We have to take into account that the probabilities for each node pair change constantly and independently the ones from the others so we are impelled to register and refresh each node pair thresholds at each execution.

5.1.5 Mode 5, Server per Node Pair Emulated M/M/∞ Dynamic Traffic with waiting state:

Mode 5 has the same principle as the Mode 4 but this time the possibility of a waiting a waiting state is given to the system. The waiting state means that being in a concrete state there is a w probability to stay in the same state after the decision. In Figure 5.8 there is the diagram for the state n including waiting state. The existence of a waiting state will cause a delay on the reach of the mean desired load of the network: the higher the waiting state is, the lower number of events will be in our network and in consequence, the more late we will reach the mean load in the network. The waiting state will in consequence introduce a transitory regime until the mean load is attained. This transitory may cause a bias in the results of the simulation representing the mean load. It would be interesting to know which waiting states we have a remarkable error or how long is the transitory stage for different loads and waiting states.
The waiting state is achieved adding a third threshold. Then for each node pair we generate a random number r and we take a decision depending on:

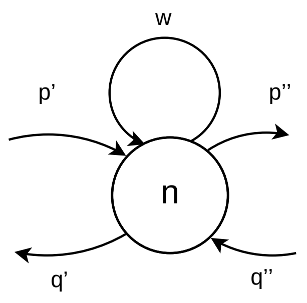


Figure 5.8: Diagram of state n with waiting probability

$$\begin{cases} 0 \leqslant r < threshold1 \text{ then it is an arrival} \\ threshold1 \leqslant r < threshold2 \text{ then it is an end} \\ threshold2 \leqslant r < 1 \text{ then nothing happens (waiting state)} \end{cases} \tag{5.7}$$

The validation of this model and other explanations are done in section 7.3.

5.1.6 Mode 6, Second Order Server per Node Pair Emulated M/M/∞ Dynamic Traffic:

In this case we try to consider a second order meaning that it could be possible to have 2 arrivals or 2 ends at the same time. The idea to create this generator has been to solve in some way the transitory regime of the previous generator (mode 5). As we are dealing with second order, this model has the ability to reach the desired mean load faster than in mode 5: it has the possibility to generate demands quicker (2 by 2 if necessary, instead of a fixed 1 by 1 as in mode 5) but he is capable as well to balance them quicker . With this model we also have a probability to be in a waiting state and we do not know in advance the theoretical mean load of the network for the given parameters.
To decide either we have 2 arrivals or only one, anyone, an end or 2 ends we use the probabilities theory to define thresholds. Being p the probability to get an arrival, q the probability to get and end and x the actual state the resulting probabilities are:

$$\begin{cases} \text{2 arrivals: } p^2(1-q)^x \\ \text{1 arrival: } p(1-q)^x \\ \text{waiting: } (1-q)^x + px(1-q)^{x-1} \\ \text{1 end: } x(1-q)^{x-1}q \\ \text{2 ends: } x(x-1)(1-q)^{x-2}q^2 \end{cases} \tag{5.8}$$

Of course in this case we have to normalize, register and refresh each node pair thresholds considering each new value divided by the sum of the rest of the new values.
The aim of this model is to be directly compared to the Mode 5 where the waiting state is considered. Indeed, the principle is the same, the more elements are in the system, the less probability exist to create new demands. We want to see which mode maintains the load the most balanced and how long is the transitory regime. Those questions are answered in section 7.3,

5.1.7 Mode 7, Server and Threshold per Node Pair Emulated M/M/1 Dynamic Traffic:

This mode has the same principle as Mode 3 but each node pair has his own defined threshold. Then, when the decision of what will happen in a node pair has to be made, we take from the given thresholds matrix the corresponding value and make the decision. This converts the network into something more heterogeneous.

5.1.8 Mode 8, Server and Threshold per Node Pair Emulated M/M/∞ Dynamic Traffic:

Here we find the principle of Mode 7 but with M/M/∞ mode. Thus, the threshold matrix has to be refreshed after each change.

In this section we have gone over all the implemented emulated traffic generators. We will take up them again to study their response in concrete OCS networks.

Chapter 6

Approach to an Energetic Study in Optical Networks

6.1 Virtual Topology Creator

The aim of this part is to provide Javanco of the possibility to create virtual topologies from a given agh. A virtual topology is a representation of the followed paths by wavelengths in a network. There is an illustrative example in Fig 6.1.

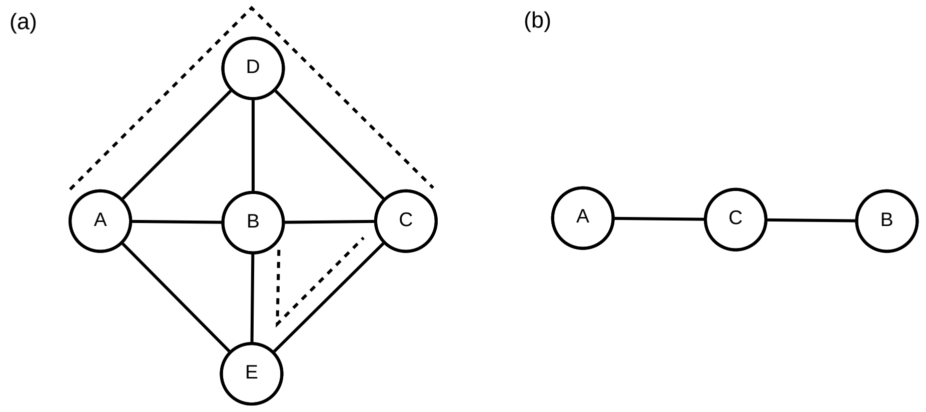


Figure 6.1: In (a) the solid lines represent the physical fiber optic links and the dotted lined represent the paths of two routed wavelengths. The two wavelengths paths create the virtual topology shown in (b), where the solid lines represent virtual links.

To create a virtual topology it is necessary to build a new layer, called "virtual", having exactly the same number of nodes but with the corresponding links. The developed creators of virtual topologies, inheriting from an `AbstractVirtualTopology` are:

- Full Virtual Topology: This virtual topology connects all the possible node pairs in the network. We obtain a fully connected virtual topology. This would mean that for each single node pair there is a dedicated wavelength, for that reason we can call it a *pure wavelength* topology.
- Over Layered Virtual Topology: This virtual topology is the exact copy of the topology on the physical layer. This architecture reminds the network level of the OSI because the whole routing is made by routing through intermediate routers, in our case, the nodes of the topology.
- Star Vitual Topology: In this virtual topology the center of the star has to be chosen. The highest node degree or node betweenness or other metrics can determine which node will be at the center. In this case we have chosen the highest node degree (obtained through the topology analyzer). Then all the rest of nodes will be connected to the central one. This means that all the lightpaths are impelled to pass through the central node (or router) to arrive to its destination.

Those are simple and basic virtual topologies just to have an overview of which would be the results of an energetic study based on them. Obviously, there are other performing virtual topologies but here we will not work on it.

6.2 Energy Cost Study

The operational costs and heat dissipation in a network comes from all the elements participating in the switching, routing, multiplexing, amplifying and transporting stages. Thus it would be interesting to study how many elements are needed in a network with a specific physical and virtual layer to calculate the total energetic cost. Based on [53] we will extract some metrics from the designed virtual topologies and calculate an approximative energetic cost. In Figure 6.2 it is shown the IP and physical layers of an IP over WDM network considered in this study. In the Table 6.1 there are the meanings and the values for some of the variables appearing in the figure and:
- f_{mn} is the number on fibers on physical link (m, n)
- D_{mn} is the physical distance of a physical link between nodes m and n

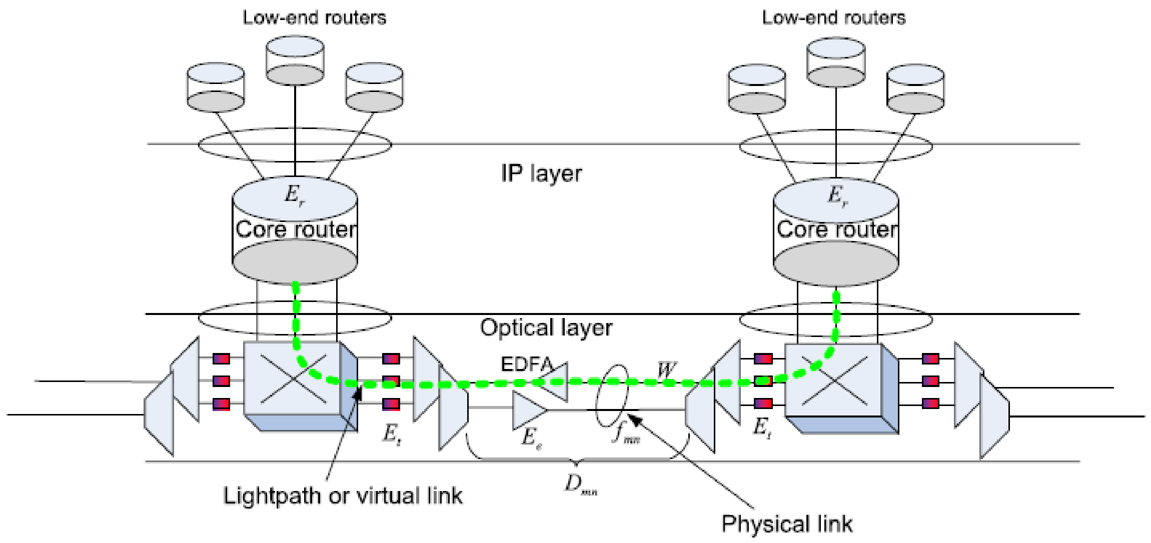


Figure 6.2: Architecture of an IP over WDM optical network extracted from [53]

6.2.1 Energetic Metrics

Those are the studied metrics obtained through the virtual topology analyze supposing uniform traffic matrix with value `val`. All metrics are obtained based on the shortest path routing algorithm.

- Number of Connections: Connections, meaning the number of lightpaths traveling through the network are obtained:
  - counting the number of demands traversing each link of the virtual layer when routing with shortest path
  - multiplying each value by `val` (the traffic rate of each demand) and obtaining like this the traffic passing through the links (if the traffic matrix would not be uniform we should route and add step by step each traffic to the concerned link)
  - getting the entire superior part for each link traffic divided by the total capacity of a wavelength
  - adding all results.

- Number of fibers: To know the number of needed fibers we count how many connections do we need in the physical layer. For this we get the number of connections per link in the virtual topology, we route those node pairs in the physical layer and we add in a register those connections related to physical links. We obtain like this the number of lightpaths passing through physical links. From now we have to count the needed fibers taking into account that each fiber can contain a given number of wavelengths and that each connection needs one.

- Routed Unities: To obtain the number of routed unities we get the node frequencies after routing in the virtual layer and we multiply each value by `val`. We add all the results. If the traffic matrix wouldn't be uniform we would have to consider the different values in traffic matrix and add them step by step.

- Router Ports: As described in [53] we calculate it as

$$R = \sum_{i \in N} (\Delta_i + \sum_{j \in N : i \neq j} C_{ij}) \tag{6.1}$$

  being N the set of nodes, Δ_i the number of ports that are used to aggregate the data traffic from the low-end routers at node i and C_{ij} the number of wavelength channels on the virtual link between node pair (i, j) (integer)

6.2.2 Cost Simulation

To simulate costs we have designed an object called `ConsumptionCounter` that, based on the equation and energy values from [53], give us a result. The whole equation is:

$$\begin{aligned} E &= \sum_{i \in N} E_r \cdot (\Delta_i + \sum_{j \in N : i \neq j} C_{ij}) + \sum_{m \in N} \sum_{n \in N_m} E_t \cdot w_{mn} && (6.2) \\ &+ \sum_{m \in N} \sum_{n \in N_m} E_e \cdot A_{mn} \cdot f_{mn} && (6.3) \end{aligned}$$

being w_{mn} the number of used channels on physical link (m, n) (integer), f_{mn} the number of fibers on physical link (m, n) (integer) and A_{mn} the number of EDFAs that should be deployed on each fiber of physical link, here we will consider a fixed number to simplify things, we choose 4.

The energy data for the study are in Table 6.1

Later we will obtain some simulation results in order to extract conclusions about energy consumption in an optical network (Cf. 7.4).

Input Data for the Study	
Number of wavelengths in each fiber (W)	8
Average power consumption per IP router port (E_r)	1000W
Power consumption per transponder (E_t)	73W
Power consumption per EDFA (E_e)	8W

Table 6.1: Energy data

Chapter 7

Simulations, Validations and Result Analysis

Since here we have develop a series of tools to analyze the performance of OCS networks. In the following sections we will validate models, compare results and extract conclusions about which are the most suitable and performing models and algorithms for those kind of optical networks.
As we have developed many tools obtaining a very flexible framework it is not possible to simulate and analyze all the combinations. That is why we will highlight the most important conclusions, launching questions and indicating other interesting case studies.

7.1 Topological Analyzer Simulation Results

The topology analyzer has been the very first developed tool.The goal is to be able to show graphically what characterizes some topologies.

7.1.1 Geographical and Topological Distances Analysis

In this analysis we study the geographical and topological distances for polygons with different number of nodes (from 2 to 19). The nodes are added to the network as if they were embedded in a fixed circumscribed circle. The simulation results permits as well to discover which values can be displayed in the graphical interface. In Figure 7.1 are represented the resulting polygons of 5, 6 and 7 nodes.

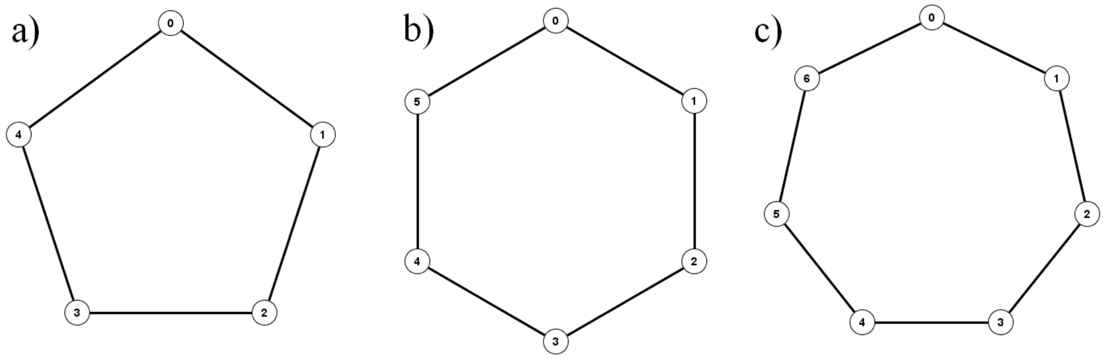


Figure 7.1: Generated polygons with: a) 5 nodes, b) 6 nodes and c) 7 nodes

Figures 7.2 and 7.3 show the results of geographical and topological distances based on the number of nodes for the generated polygons. The key of the graphic indicates:
- in red the maximum obtained value for each polygon; i.e. in a) the maximum value considering the shortest path routing would be from node 0 to 2 (or 1 to 3, etc), in b) it would be from 0 to 3 (or from 1 to 4, etc.)
- in pink the minimum obtained value (in those topologies, the distance between neighbors)
- in blue the mean of all results for each polygon (light blue means the same)
- in green the medium value (the value in the middle of the sorted results)

In the geographical distance results (Figure 7.2) we see how the maximum value fluctuates: as polygons have a pair or odd number of nodes the most separated nodes following the shortest are once diametrically opposite and once (when we add a new node) a little bit closer. The distances keep growing because we are getting closer to the arc of the circle but we always find this fluctuation. Indeed there is a limit in the maximum distance that would be half of the length of the circumscribed circumference. As well, the minimum value keeps decreasing as neighbors get closer each time we add a node. The limit is 0.

Topological distances can be studied analogically (Figure 7.3). As we are dealing with hops, now we get integer numbers. The maximum distance has the same value for two consecutive topologies and increases 1 each time. Based on the topologies in Figure 7.1 we have that for a) the maximum number of hops is 2, for b) it is 3 and

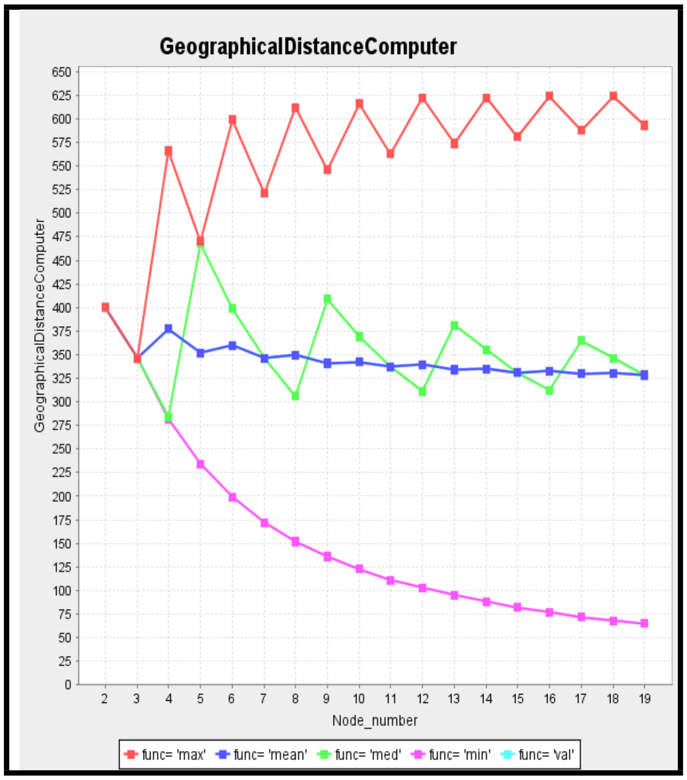


Figure 7.2: Results for the Geographical Distance

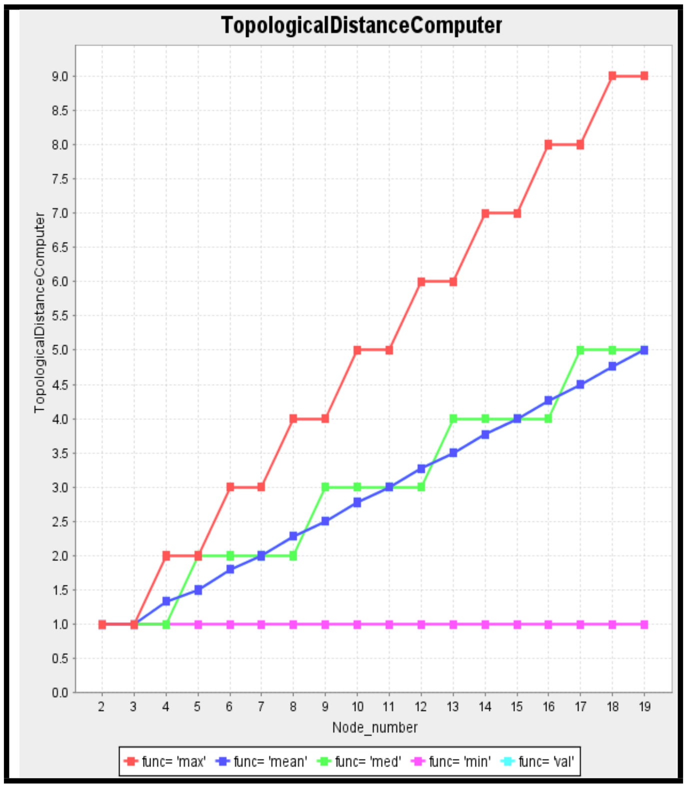


Figure 7.3: Results for the Topological Distance

for c) it is 3 as well (corresponding to the graphic values). Then the explanation is found again in the parity of the number of node of the polygons. In the odd nodes polygons the higher number of hops is $\frac{N-1}{2}$ meanwhile in the pair nodes polygons it is $\frac{N}{2}$ with N the number of nodes. In this case there is no limit and as we add nodes, the number of maximum hops grows. The minimum topological distance is always the hop to go to the next node. For that reason this value is constant and will always be 1 no matter the number of nodes.

7.1.2 Node Betweenness Analysis

From the developed metrics, we choose to analyze the node betweenness because is one of the most intuitive one. For that reason we have chosen for the case study a 5-nodes polygon and a 5-node star topologies. It comes naturally to say that in the 5-node start topology, the central node will have the higher node betweenness related to the others. On the other hand, for the 5-node polygon we guess that all nodes have the same importance and though, there will have equivalent node betweenness values.

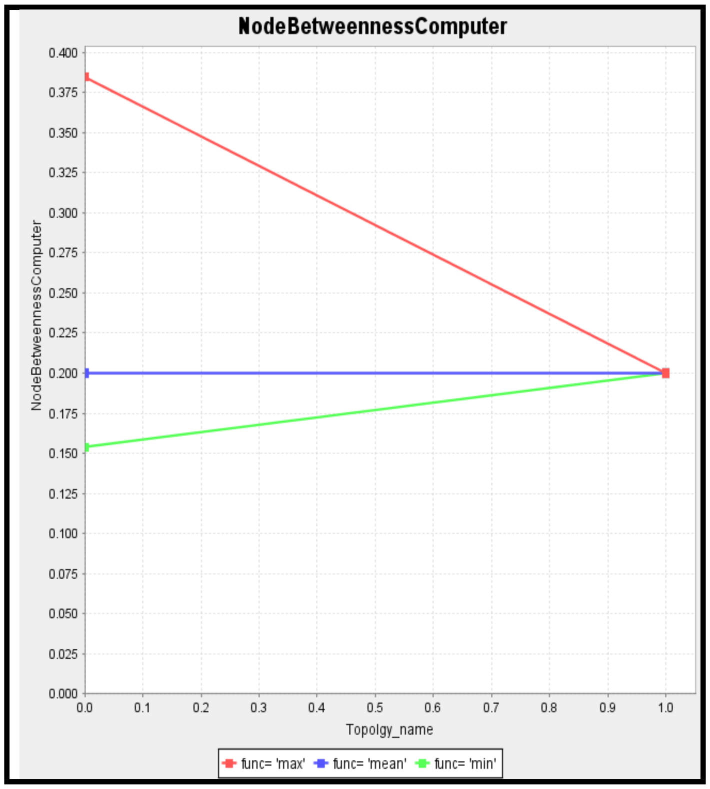


Figure 7.4: Node Betweenness results for a 5-nodes star and 5-nodes polygon topologies

Figure 7.4 shows the results for the node betweenness simulation. The x axis indicates which topology is studied: x = 0.0 is related to the star topology and x = 1.0 is related to the polygon.
As supposed, there is as maximum value for the star topology corresponding to the central node. This value is approximately 0.348. We know that node betweenness gives relative values and we also suppose that the rest of nodes (thus, the other 4) have the same weight. Indeed, the minimum value is 0.154 and 0.348 + 4*0.154 = 1. The polygon topology results confirm that all nodes have the same importance due to the fact that the maximum and minimum value are the same, 0.2.
The mean curve corroborates that the addition of each result per node is 1 and the division by the number of nodes is exactly 0.2.

The role of the topology analyzer would be to characterize topologies in order to classify them based on results. Then we could use them to study general behaviors as if topologies have nodes or links particularly used (in order to, for example, provide those components with higher capacities), if the represented topology is highly meshed or not or if the links' length are shorter enough to avoid regeneration.

7.2 OCS Networks Model Simulation Results

We have developed many tools to take into a account in an OCS network simulation. In summary there are different types of:
- topologies
- traffic matrix creators (uniform or based on different distributions)
- dynamic traffic generators
- ordering policies for incremental traffic generators
- routing algorithms
- wavelength continuity constraint
- wavelength assignment algorithms

If we would like to consider all the possibilities, we would find some cents of combinations. Therefore we will consider some aspects and fix some others to extract conclusions. Then the obtained conclusions will be based on particular conditions and those could vary under other ones.
Moreover other configurations, models and algorithms could be developed and added easily to this framework.

7.2.1 Validation of Real Dynamic Traffic

The first objective here is to validate the Real Dynamic Traffic generator model. We want to validate specifically that the duration of the demands in the model get the shape of the distribution we define in the OCS model to generate this time.Time in the real traffic generator in the most important parameter to take into account and that is why we want to corroborate that it works correctly.

We impose to the network an acceptance ratio of 1 (all demands are accepted) in order to take into account all the generated demands and their length in real time. The simulation is done for 5000 events in a 5-nodes polygon topology and for 7 seeds. First we choose a normal distribution to model the time between arrivals and the time of duration of the demands. In particular we choose normal distributions with mean 10 and variance 10. During the simulation we will count how many times the same demand length or duration of demands (rounded to the hundredth)appear during the simulation. The most important remark to do is that normal distribution can reach negative values. When this happens during the simulation, the value is ignored and we obtain a new value until it is positive, we are truncating the distribution. The direct consequence of this fact is an increase of the theoretical mean value of the results without truncation.Indeed, we do not know theoretically which should be the mean value of a truncated normal distribution with a given mean and variance.
In Figure 7.5 we have the obtained curve. In x axis there are the length of the demands (in microseconds) and in y axis there are the occurrence of each demand length. We recognize the truncated normal distribution as we represent the number of times demands have the same duration. The given mean to the simulator has been 10 but after the normal distribution being truncated, the calculated mean after the simulation is 12.49 and this value is higher than 10, as we had supposed. Indeed, in the curve, the majority of values are between 6 and 17.

We try the same simulation choosing an exponential distribution. The experiment has the same exact parameters, 5-nodes polygon topology, 5000 events and a mean of 10. The simulation results are in Figure 7.6. We recognize the exponential form, but does it corresponds to the theoretical one? Is there any limitation?

To plot the theoretical exponential we first normalize the number of occurrences by the highest value, like this we could consider an theoretical exponential distribution also normalized. So we represent at the same time the normalized results of the simulation and a curve representing $e^{-\frac{1}{10}x}$ where x are the duration of demands we are considering (from 0 to 100 by steps of 1). In Figure 7.7 there is the result of this superposition of curves. Looking at this figure we could say that they are quite

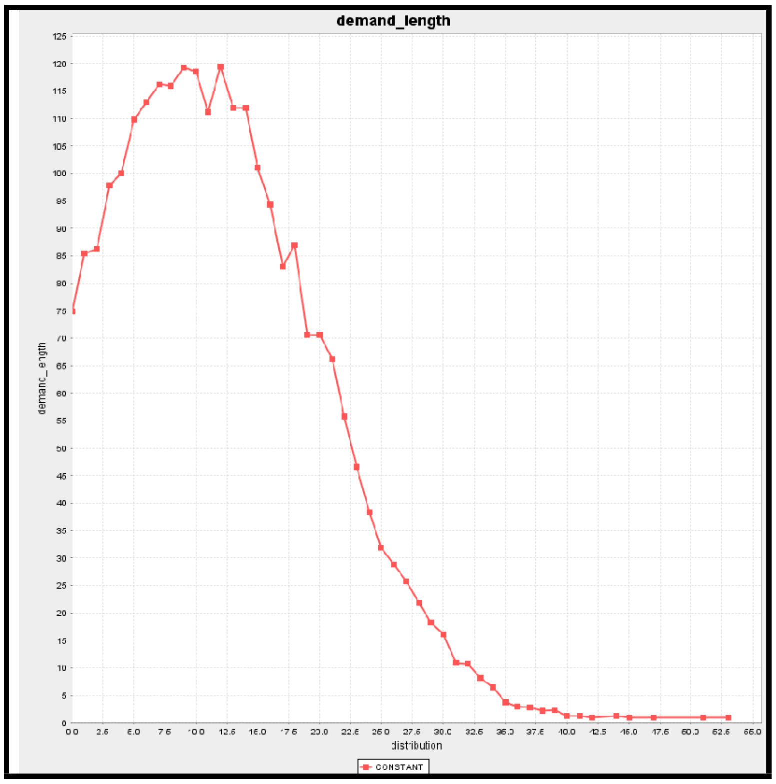


Figure 7.5: Real traffic generator results for normal distribution

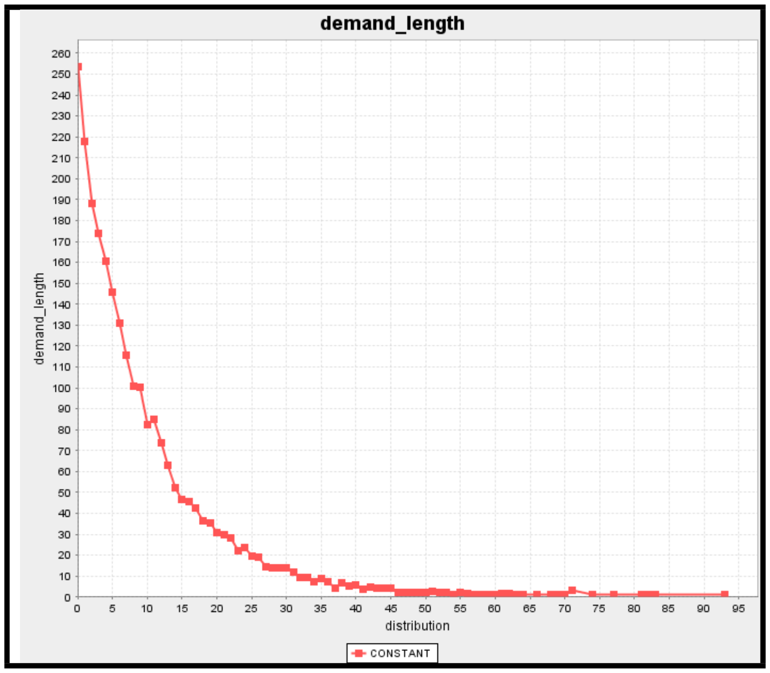


Figure 7.6: Real traffic generator results for exponential distribution

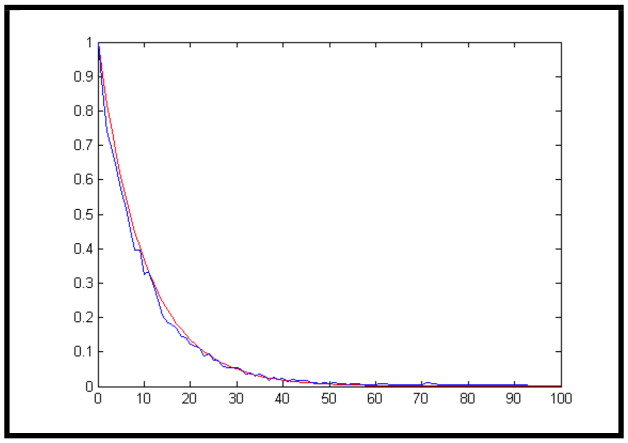


Figure 7.7: Theoretical and simulated results for exponential distribution

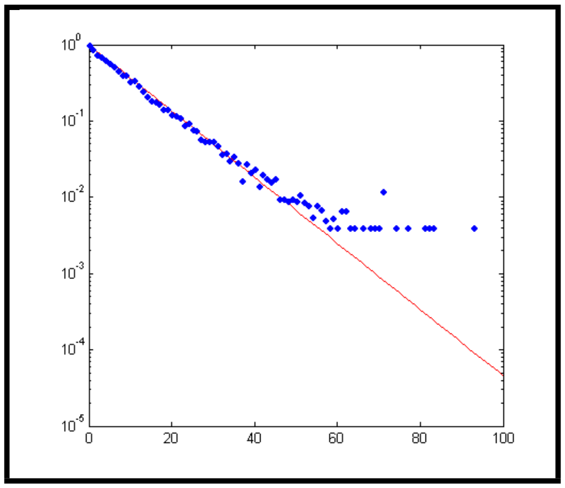


Figure 7.8: Theoretical and simulated results for exponential distribution in a logarithmic representation

similar but we can use another tool to detect if the simulation results are good enough or if there is any limitation. The main idea of the tool is to represent the logarithm of both curves. The resulting curves are in Figure 7.8. We observe that the blue dots, which correspond to the results of the simulation, present a limit. This limit is due to the fact that the minimum value we can obtain in the simulation is 1, and there are a lot of 1 for high x values (or for long duration demands). Hence, when this minimum and repeated value of one is normalized and then we represent the logarithms we see that the curve do not pass this minimum normalized value. Moreover we can observe that the simulation tends to have longer demands than the theoretical ones (blue dots over the solid red line). The model seems to be correct enough for short demands. Considering that we are not able to overcome this limitation (the only way would be to have infinite demands) we can validate the simulation model despite this restriction.

So even if the results of the simulation seem to be right at first sight, both have limitations:
- the normal distribution is truncated then the mean is greater than the theoretical one
- the exponential distribution has a threshold that cannot be overcame in any situation generating longer demands (not expected in the theoretical model)
All in all, we observe that the consequence of those restrictions is the increase of the mean duration of demands. This fact participates negatively to the congestion and load of the network: the network is loaded more time than predicted. Resources will be used during longer times resulting in a higher blocking probability as new demands expecting to use already free channels will get refused. And this means that we have a general loss of performance in the OCS networks using this Real Dynamic Traffic generator.
It would be interesting to keep studying other distributions in order to depict other limitations or if, on the contrary, they work according to the theory.

7.2.2 Wavelength Assignment Algorithms Simulation Results

The study of the wavelength assignment algorithms, the incremental traffic orderings and the wavelength continuity constraint will be done sequentially, using each time the results of the previous analysis to continue. As wavelength assignment is the first studied block, we detail here the conditions of the simulations.

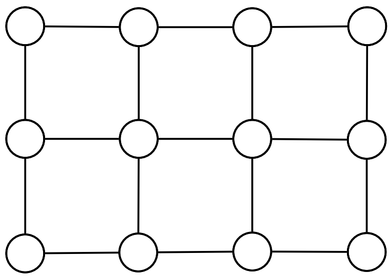


Figure 7.9: Square tessellation topology

To study the performance of the implemented wavelength assignment algorithms we fix the following conditions:
- an Incremental Traffic generator with random ordering in order to know the % of accepted demands
- a traffic matrix generator based on a normal distribution with mean 5 and variance 4

- two topologies: a 6-nodes polygon and a square tessellation as represented in Figure 7.9 (3 rows, 4 columns)
- fiber links configurations with 12, 18, 24 and 30 as total capacity and organized by fibers and wavelengths as in Table 7.1

capacity \ number of fibers	1	2	3
12	12	6	4
18	18	9	6
24	24	12	8
30	30	15	10

Table 7.1: Number of wavelengths per fiber

The simulations are done applying First Fit, Most Used and Least Used algorithms.
First we focus in what happens for both topologies when we fix a capacity per link of 12 and we consider:
- 1 single fiber of 12 wavelengths
- 2 fibers of 6 wavelengths
- 3 fibers of 4 wavelengths

The results for the tessellation are in Figure 7.10 and for the polygon in Figure 7.11. The y axis represent the % of accepted demands and the x axis the wavelength assignment algorithms corresponding to:
- First Fit: x = 0.0
- Most Used: x = 1.0
- Least Used: x = 2.0
The solid colored lines represent the number of fibers per link. The most remarkable observation coming out of those results is that in any topology or link configuration the wavelength assignment with higher acceptance ratio is the Least Used. If we compare the values based on the number of fibers, the Least Used WA is more performing as the number of fibers is lower. The gain for the LU algorithm in the square tessellation is 2.5% if we use 1 fiber instead of 3, in the polygon is 0.7%. Otherwise, no particular conclusions can be extracted from the other two WA algorithms.

A particular situation is found when we observe what happens for different capacity values, thus 12, 18, 24 and 30. Results are represented for the square tessellation and for the polygon in Figure 7.12 and Figure 7.13 respectively. This time the x axis represents the number of fibers considered and the y axis still is the acceptance ratio.

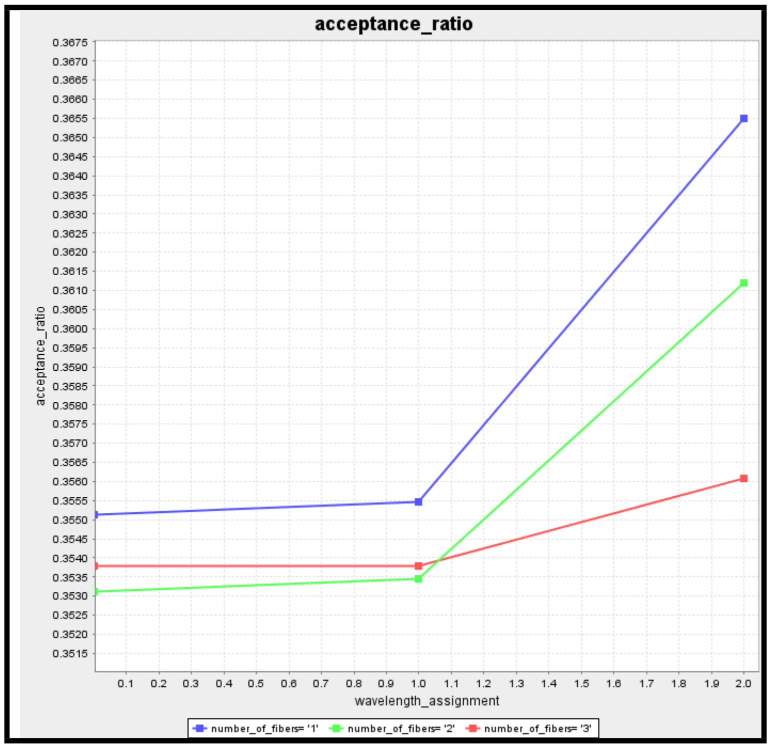


Figure 7.10: Accepted demands for a square tessellation and 12 channels per link

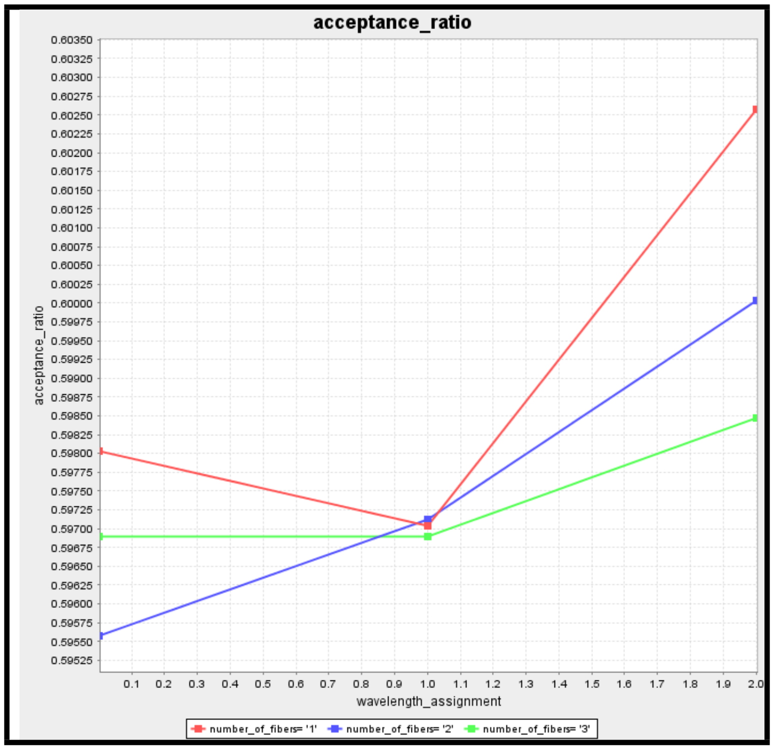


Figure 7.11: Accepted demands for a 6-node polygon and 12 channels per link

Solid lines represents the wavelength assignment algorithms and the shapes are the capacities. We find the key at the bottom of each figure.

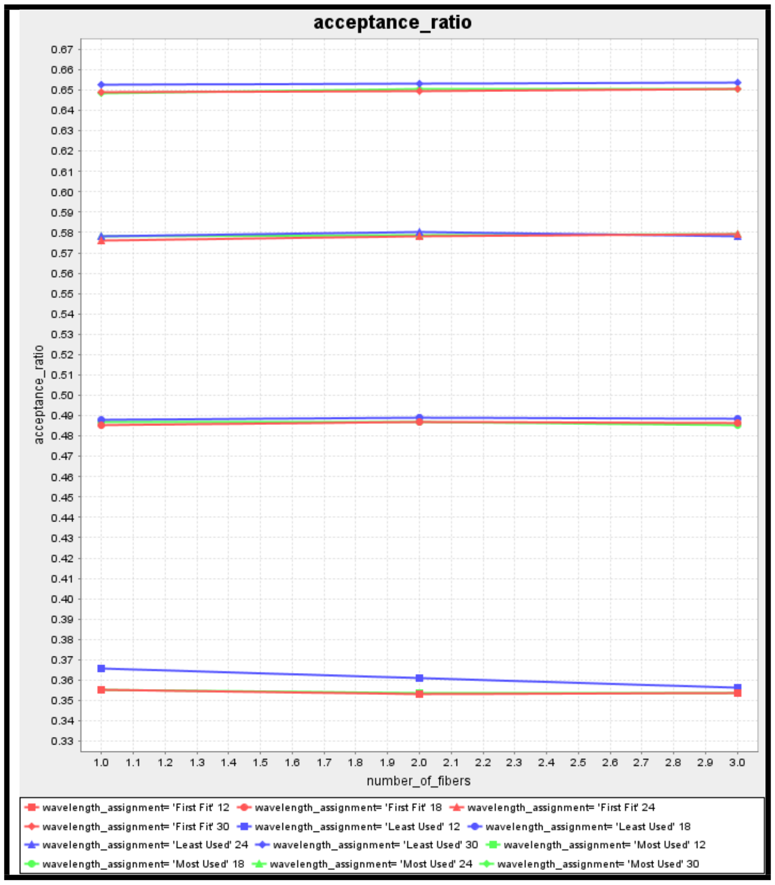


Figure 7.12: Accepted demands for a square tessellation and 12, 18, 24 and 30 channels per link

In the tessellation topology results we observe that in almost all cases the same order in acceptance ratio is conserved. Then it would seem that, in general, LU is the more performing wavelength assignment algorithm, followed by the MU and FF in last position (even if MU and FF do not show big differences).
If now we take a look to the polygon topology results, we realize that for a capacity of 12, LU is the most performing, but since the capacity is 18 or higher, MU turns to be the most performing in any situation.
The most perceptible difference between the studied topologies is how meshed they are, then the consequently question is: does the performance of the wavelength assignment algorithms have any dependence in how much the physical topology is meshed?

We conclude through this section that there is no one single performing wavelength

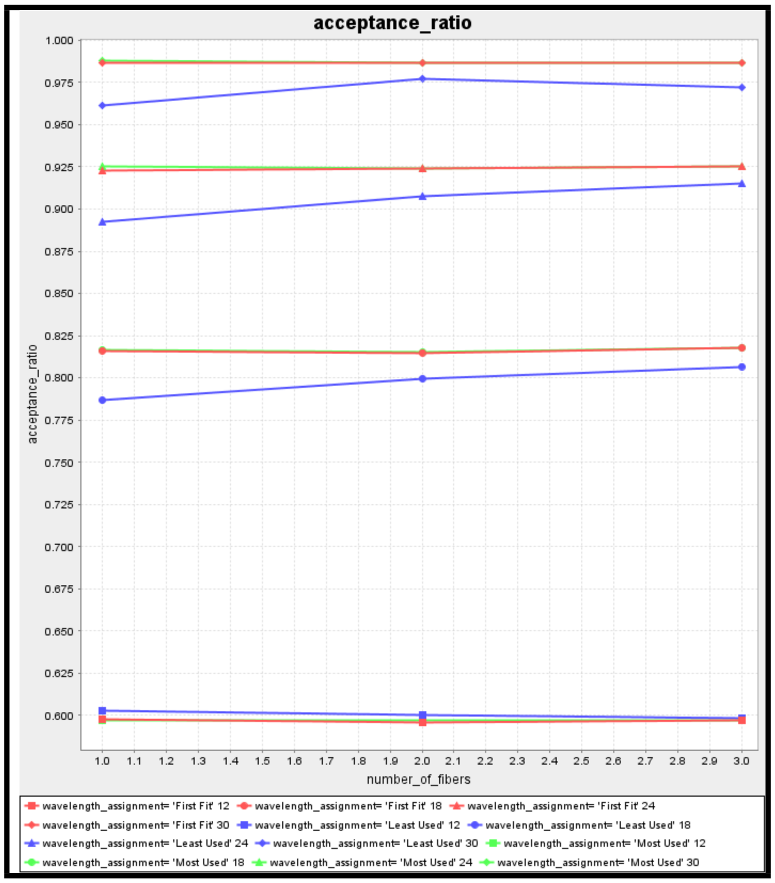


Figure 7.13: Accepted demands for a 6-node polygon and 12, 18, 24 and 30 channels per link

assignment but that everything depends on the inputs conditions.

7.2.3 Types of Incremental Traffic Simulation Results

We choose again the same conditions (topologies, distribution for the traffic matrix) but this time we fix for each wavelength assignment that link have 2 fibers with 10 wavelengths each. In this situation we study which are the most suitable orderings for an incremental traffic to obtain a higher acceptance rate. For the square tessellation topology results are in Figure 7.14 and for the polygon topology in Figure 7.15.

In the x axis we have the wavelength assignment algorithms as:
- First Fit: x = 0.0
- Most Used: x = 1.0
- Least Used: x = 2.0

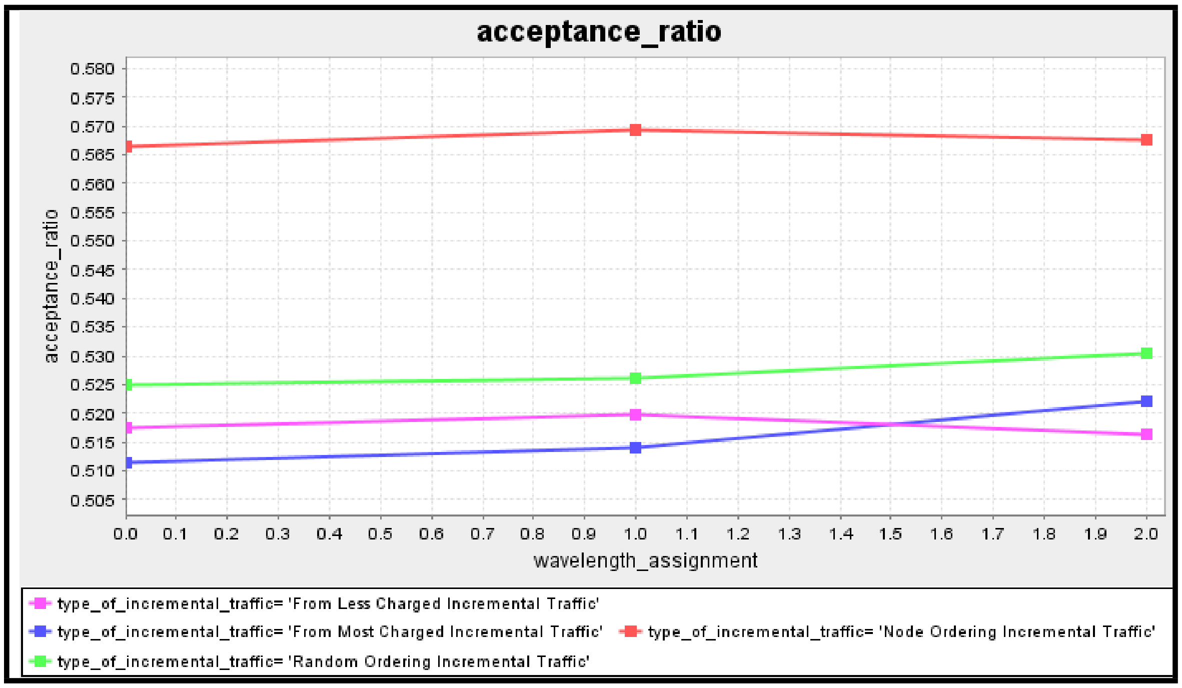


Figure 7.14: Accepted demand rates for a square tessellation and different types of incremental ordering

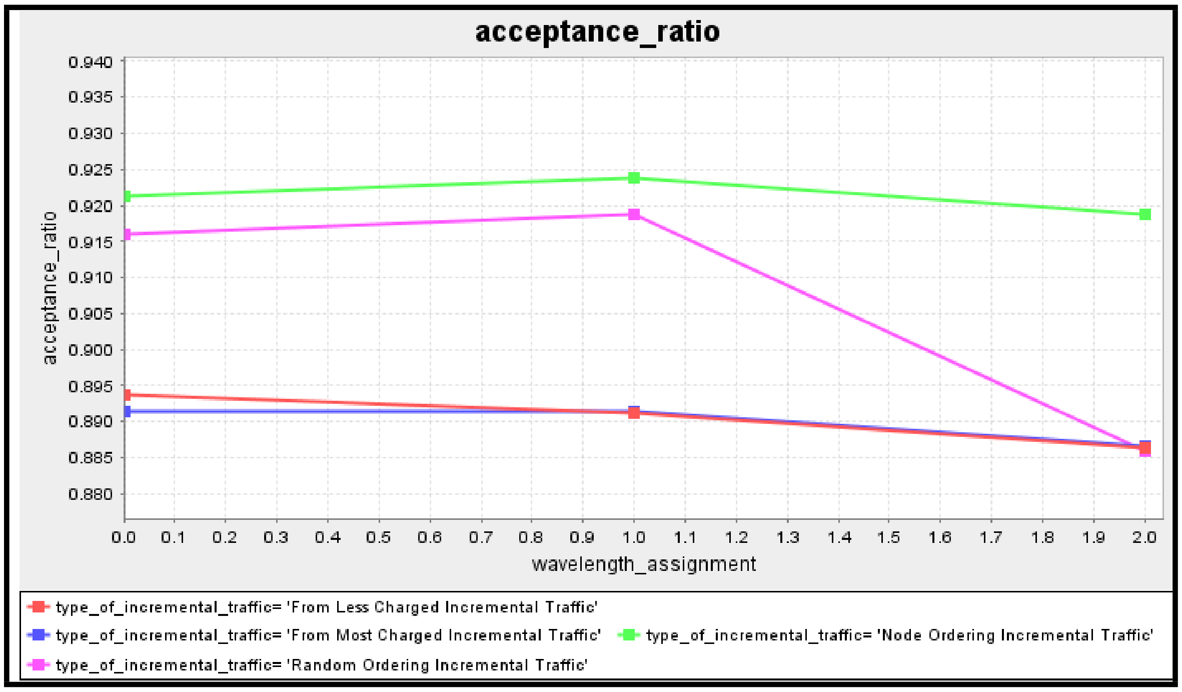


Figure 7.15: Accepted demand rates for a 6-node polygon and different types of incremental ordering

The solid colored lines represent the 4 ways to order the demands in the traffic matrix developed.
In both cases, the Node Ordering seems to be the most performing independently of the wavelength assignment algorithm. It could be due to the fact that wavelength channels are getting filled progressively following a logic order and resources are used little by little. The ordering beginning from the most charged use quickly a lot of resources in each link and the one beginning by the less charged, occupies a lot of link in a sparse way and finally there are not enough resources for demands much more charged. The random ordering is the second more favorable case.
From this particular situations we extract that the Node Ordering for an Incremental Traffic would be the most performing. Anyway, we can not affirm that this will be always true.

7.2.4 Wavelength Continuity Constraint Simulation Results

We have implemented in this work the possibility to make optical networks using or not wavelength continuity constraint. We have just seen that, in most of the cases it is not possible to predict which one would be the best solution for a given inputs. It is known that the ability to enable wavelength conversion in all nodes should improve the acceptance ratio (although being less performing energetic and economically because component allowing it are expensive and use extra energy). We want to observe what happens in our system if we make simulations considering the previous topologies, fixing again the same distribution for the traffic matrix, considering all the WA algorithms and also all the ordering policies for the incremental traffic. Results for the tessellation are found in Figure 7.16 and for the 6-nodes polygon in 7.17. The x axis represents the WA as previously but this time there is an extra element in x = 3.0 for the reference system thus the system with wavelength conversion.
The results for the tessellation does not seem to fit with the prediction above. The values in blue which are the reference system are in almost all cases worst than the most performing WA (for the node ordering it is higher but nothing significant). On the other hand the results for the polygon are more performing for most and least charged incremental ordering using reference system (2.43% and 1.46% respectively) while the difference in the other cases are insignificant.

In front of these results we would tend to say the improvement of the wavelength conversion compared to wavelength continuity constraint is not enough to establish it in network. And we are confronted again to the doubt that if wavelength conversion does not apport any improvement or is it a special situation for those cases? would a sparse wavelength conversion would be better to ameliorate the performance?

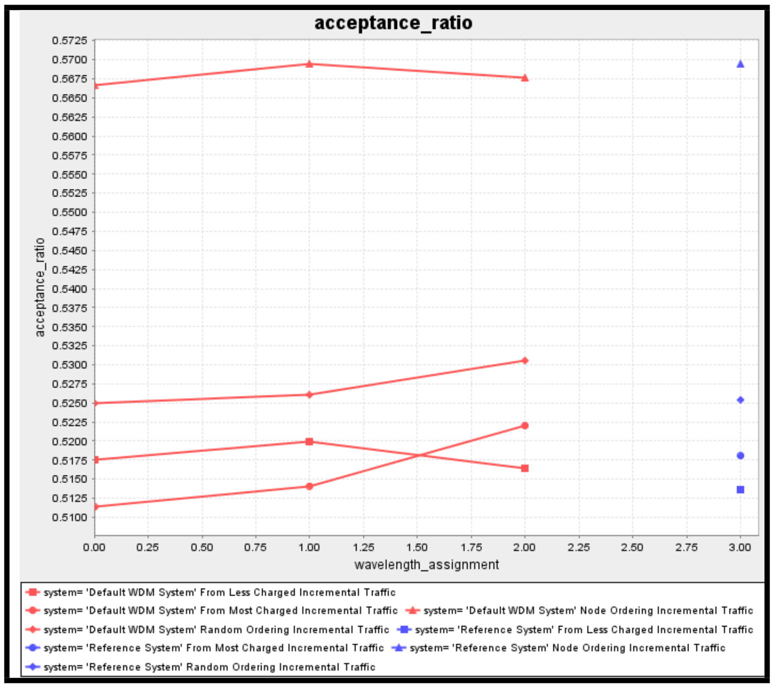


Figure 7.16: Accepted demand rates for a square tessellation and OCS systems

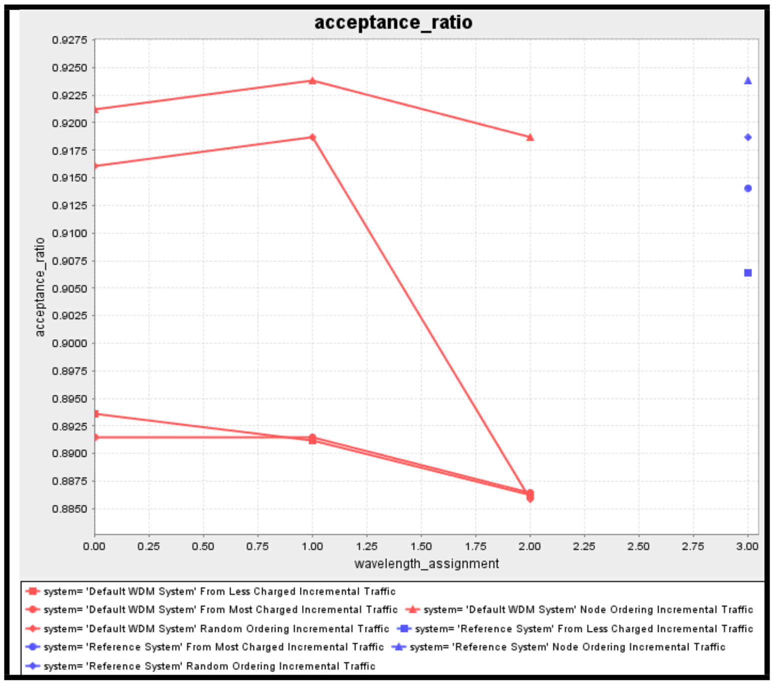


Figure 7.17: Accepted demand rates for a 6-node polygon and OCS systems

7.2.5 Occupancy Depending Routing Simulation Results

In this section we want to validate the occupancy depending routing. For this we choose a 6 nodes square tessellation (2 rows and 3 columns) and an incremental traffic with a traffic matrix filled using a normal distribution. We fix a capacity of 100 wavelengths for each link. The goal through results is to corroborate that the maximum number of used channels in a link does not surpasses the maximum occupancy percentage. We simulate using an occupancy depending routing algorithm with maximum occupations of 10%, 30%, 50%, 70% and 90%. As link capacity per link is 10, we expect a maximum number of occupied channels of 1, 3, 5, 7 and 9 for each considered maximum occupancy %.

Results are shown in Figure 7.18. Each integer of the x axis corresponds to a physical link in the topology. In y axis we have the maximum number of channels used during the simulations (results are not integers because the simulation is done for various seeds and the represented value is the mean).
We observe that the maximum values per link never surpasses the predicted threshold. Indeed, and occupancy of 10% is very easy to reach and all links receive at least one demand. The rest of maximum occupancies never reach the maximum value but they are close anyway. This routing algorithm is finally validated.

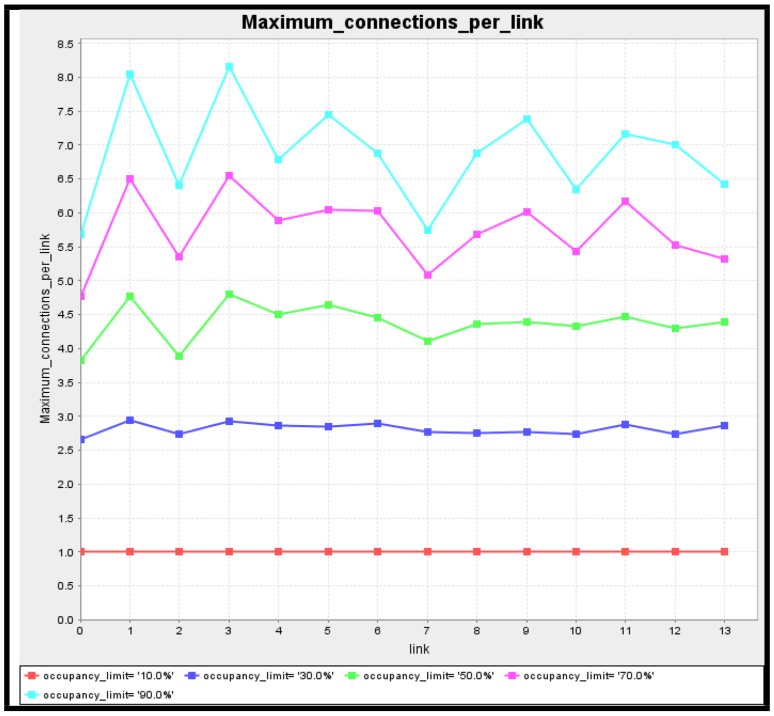


Figure 7.18: Maximum occupancy per link using occupancy depending routing

7.3 Emulated Traffic Generator Models Simulation Results

We proceed here to study the different types of emulated traffic generators explained in section 5.1. It is interesting to prove that our models are valid and that we are able to play with the traffic offered considering the network as a single server, as an infinity of servers or even considering that each physical link has his own server (or more).
Some validations have already been done in 5.1 for the M/M/1 and the M/M/∞ models.
As always, the ultimate objective is to , knowing the behavior of each generator, decide whether would be the most performing one: which would accept the highest number of demands, which would do a good use of the resources or which one would allow a high offered traffic maintaining the load of the network.

The first simulation we do is the comparison between the M/M/1 and the M/M/∞ systems in order to corroborate the conclusions explained before and see what happens with the traffic offered.
We start simulating what would happen for a 10000 simulation in a 2 nodes and 1 link topology, taking 11 different seeds for the PRNG and with an arrival rate from 0,05 to 0,45 by steps of 0,05. We consider a maximum simulation arrival rate of 0.45 because 0.5 it is the maximum theoretical value that we can consider in this simulation.
In Figure 7.19 there are the mean number of elements for the two system based on the removal probability (which corresponds to μ or to $1-\lambda$). We find that, as said before (Cf. section 5.1), the number of elements in the system are lower in the M/M/∞ for higher arrival rates thus it is translated in a more "controlled" traffic offered for big arrival rates (in Figure 7.20).

Even if theoretically it is not permitted we want to know what happens when the arrival rate is 0,5 or higher. In Figure 7.21 there are the results. The offered traffic scale has no importance since we note that from 0,5 the offered traffic increases as the removal probability decreases. On the other hand, the M/M/∞ controls it and keep it low (in comparison to the M/M/1).

Theoretically, if a network has m servers we should obtain a mean number of elements m times than the mean number of this same network with a single server .

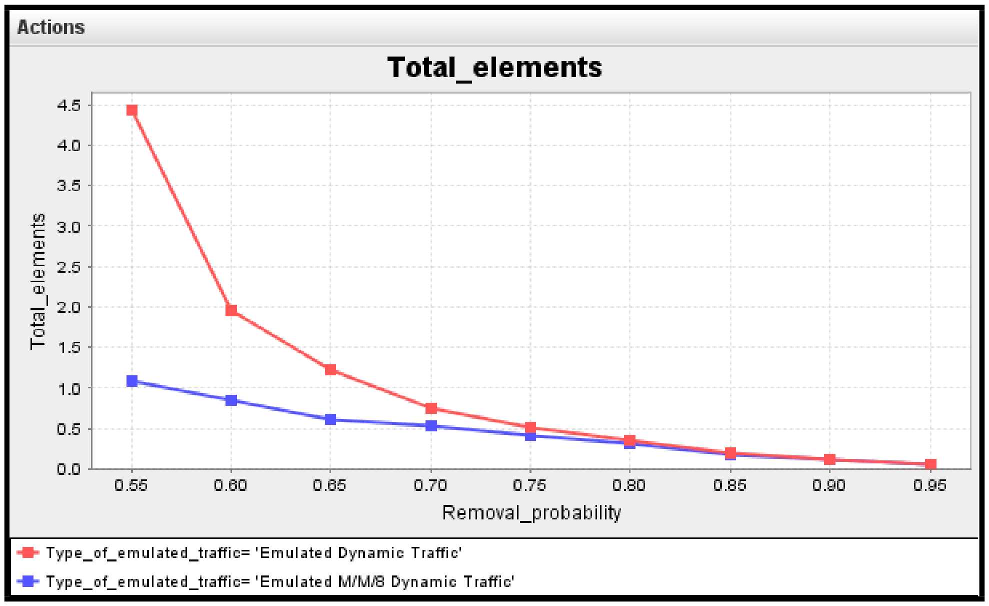


Figure 7.19: Total elements in M/M/1 and M/M/∞ systems

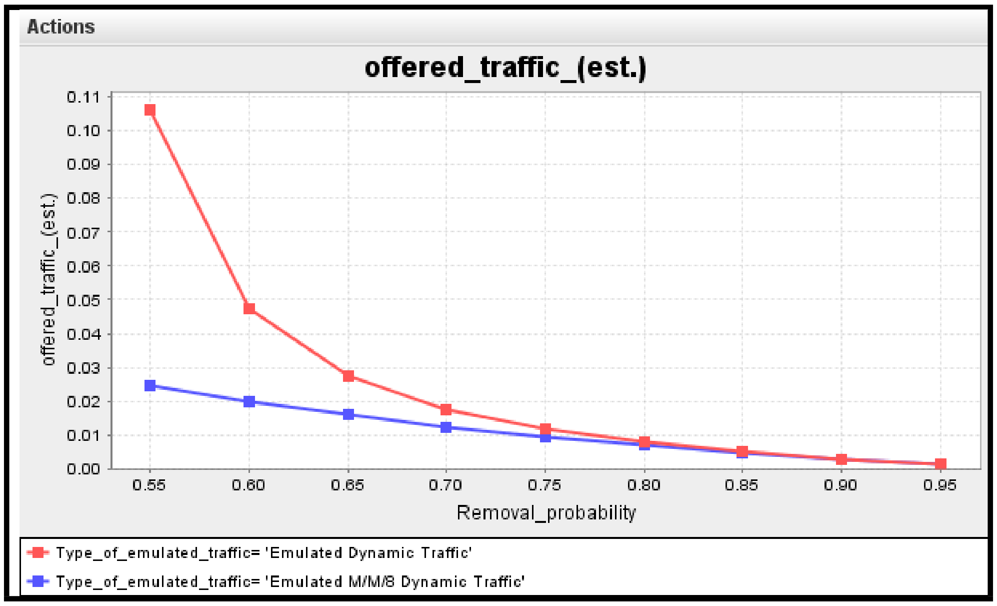


Figure 7.20: Traffic offered in M/M/1 and M/M/∞ systems

To prove this graphically we take the M/M/1 system and the system that gives an M/M/1 structure to each node pair (Mode 3). We simulate 10000 events, for a 2 nodes and 1 link network (then, 2 node pairs so the equivalent to 2 servers in the network), for an arrival probability from 0,05 to 0,45 and we obtain the Figure 7.22. This figure not only represents the mean number of elements but also the offered traffic that, as supposed, it has to double as well. Indeed, the two curves for the different systems have the same shape. In the table 7.2 we have transcribed the obtained values and

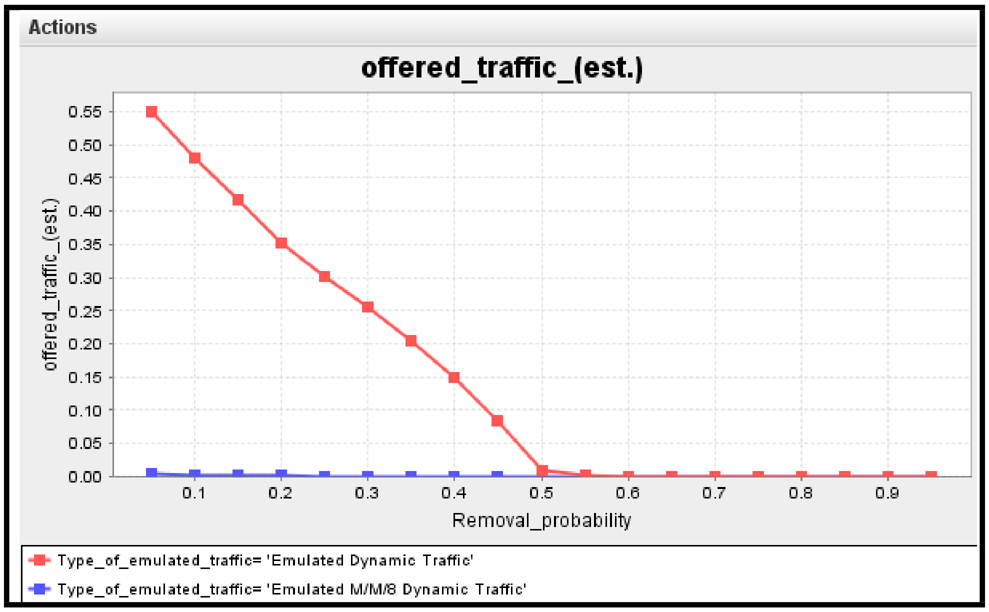


Figure 7.21: Traffic offered in M/M/1 and M/M/∞ systems for high arrival rates

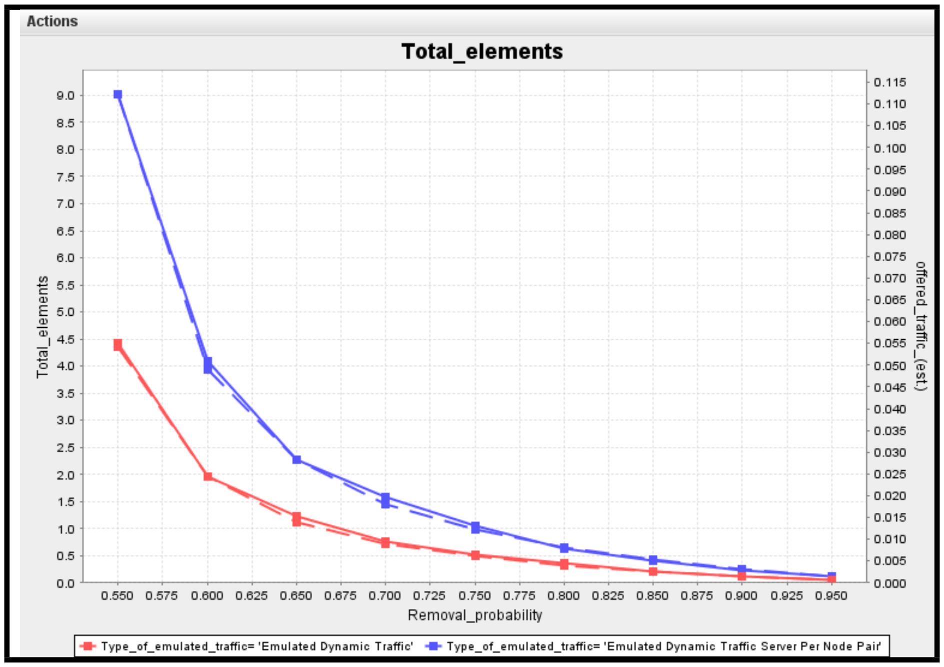


Figure 7.22: Total Elements and Traffic offered in M/M/1 and m times M/M/1

the corresponding confidence interval of 95% and the theoretical ones. We observe that all the theoretical values are ini the confidence interval and then we conclude that the model can be valid and allows the multiplication of the offered traffic by the number of node pairs in the network. The same simulation has been made for other topologies with different number of nodes n and we obtain that the traffic offered is

λ	μ	theoretical N value	theoretical $2N$ value	obtained $2N$ value	confidence interval 95%
0.05	0.95	0.055	0.110	0.122	[0.093, 0.150]
0.10	0.90	0.125	0.250	0.227	[0.187, 0.267]
0.20	0.80	0.30	0.600	0.633	[0.562, 0.705]
0.30	0.70	0.75	1.500	1.612	[1.476, 1.747]
0.40	0.60	2.00	4.000	4.247	[3.963, 4.531]
0.45	0.55	4.50	9.000	9.313	[8.702, 9.925]

Table 7.2: Theoretical and simulated values for the mean number of elements in a 2 servers with M/M/1 system

$n(n-1)$ times the traffic offered of one single server.

Once the mode 3 is validated as a model, we proceed to validate the mode 7 that gives to each node pair a specific threshold. The mode 7 would be validated if for an homogeneous threshold matrix with value p everywhere (except in the diagonal) would correspond to the mode 3 results with a fixed p as arrival threshold. The simulation gives exact results between the two modes for all the metrics, for different topologies and thresholds. Then the model of the mode 7 is validated. Even if we will not study it in this project, it could be interesting to see how the offered traffic changes and which is the network behavior when we give to node pairs different thresholds to control the traffic. The choice of thresholds could be based on the traffic matrix, on the length of lightpaths between node pairs or other metrics.

If now we consider mode 2 and mode 4 (that is an M/M/∞ system for the whole network and an M/M/∞ system for each node pair in the system) we should find the same relationship between results as for the M/M/1 cases. With an acceptance ratio of 1, simulating an arrival probability from 0.05 to 0.95 , 10000 events and considering a polygon of 3 nodes (thus, a 6 node pair network) we obtain the traffic offered displayed in Figure 7.23 and the values in Table 7.3. The obtained rate is again very close to the theoretical value and then we can validate this model.

Mode 8 is validated through mode 4 in the same way mode 3 validates mode 7. We have for mode 8 a matrix of thresholds which gives to each node pair, working with the M/M/∞ system, his own threshold. We obtain the same exact results than mode 4 with the same initial threshold.

The last validation involves the mode 5 which is the same as mode 4 (an M/M/∞ server for each node pair) but giving the possibility to stay in a waiting state. If we impose to the mode 5 a waiting state threshold equal to 0 (disabling then the waiting

λ	μ	traffic offered 1 server	traffic offered 6 servers	rate
0.05	0.95	0.0002	0.0012	6.065
0.10	0.90	0.0005	0.0030	6.022
0.20	0.80	0.0011	0.0070	6.045
0.30	0.70	0.0020	0.0121	6.040
0.40	0.60	0.0033	0.0200	6.023
0.50	0.50	0.0049	0.0299	5.993
0.60	0.40	0.007	0.045	6.044
0.70	0.30	0.011	0.067	5.928
0.80	0.20	0.018	0.109	5.956
0.90	0.10	0.038	0.231	5.986
0.95	0.05	0.079	0.475	5.985

Table 7.3: Comparison between the simulated values for the traffic offered in a 6 servers with M/M/∞ system and with a single one

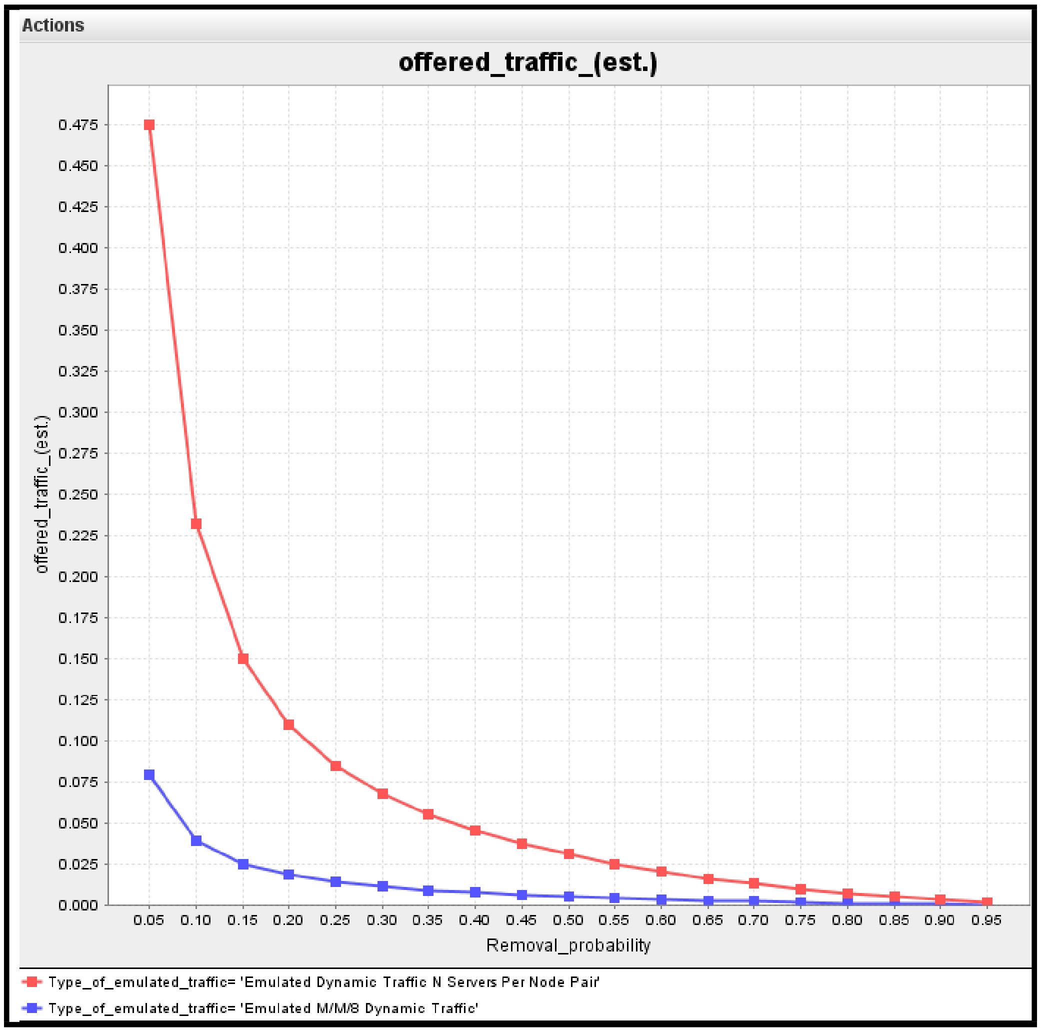


Figure 7.23: Traffic offered in M/M/∞ and m times M/M/∞

state) we should obtain the same results as mode 4. Indeed, the simulation results are exactly the same and then the validation is done.

The next step to follow is to launch some simulations to study this mode 5 allowing a waiting state. First we want to prove how for different probabilities of waiting state influence in the load of the network. For this we choose:
- a topology with 2 nodes and 1 link
- we generate 1000 events with Mode5 traffic generator
- as we know that for M/M/∞ systems the mean number of elements in the network (or load) is defined as $N = \frac{\lambda}{\mu}$ we fix $N = 16$ and $N = 4$ and we modulate λ and μ to obtain those values based as well on the desired w probability of waiting state
- we chose waiting states $w = [0.4, 0.8, 0.9, 0.95]$ in order to compare what happens in each case
The restrictions taken into account to calculate the corresponding values are:

$$\lambda + \mu + w = 1 \tag{7.1}$$

and

$$N = \rho = \frac{\lambda}{\mu} \tag{7.2}$$

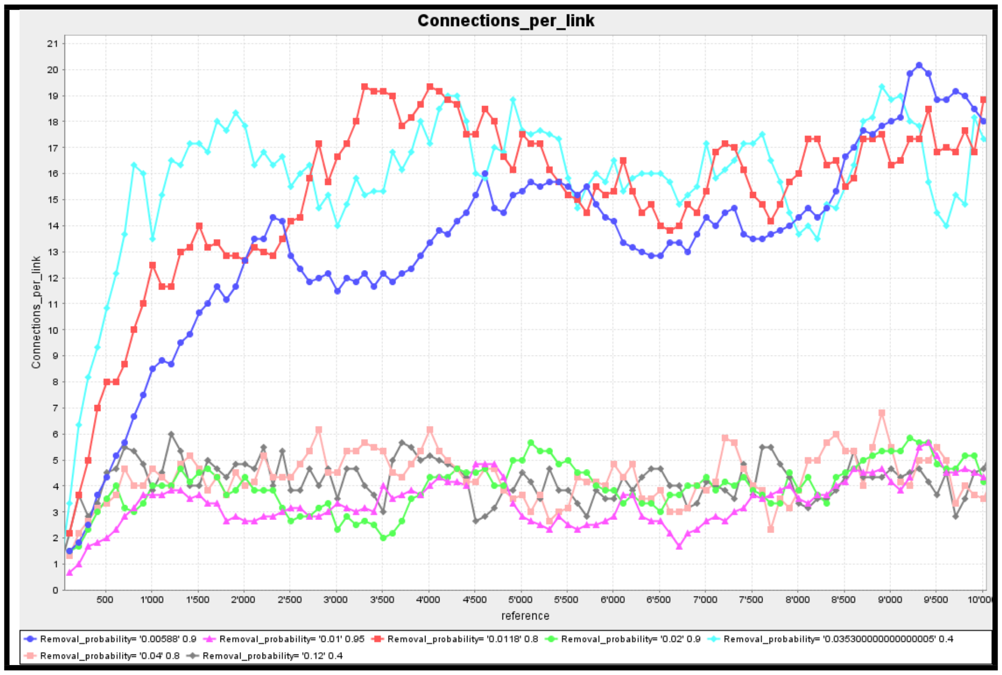


Figure 7.24: Load in networks using Mode5 for different waiting state

We include in Figure 7.24 the results of the simulation with those values. In x axis we have the time reference (given by the step interval) and in y axis there are

the mean connections per link all along the simulation. The key specifies the removal probability as well as the waiting state probability fixed for this concrete removal probability.
The first clear observation it that there are transitory regimes in all cases before reaching the right average load in the network. The second observation is the confirmation that, once the load is reached, it that the value is maintained. In fact, those loads corresponds to the defined N load, 4 and 16. Obviously, to reach a mean of four takes less time than reaching 16. That is why in general, the transitory regime for a load of 4 is shorter than for 16. But we also detect differences among the speed of the growth of the load in the network.
If we focus on the blue dotted line we read in the key that it corresponds to a waiting state of 0.9 and a little removal probability. We want to compare it to the light blue lozenged line which corresponds to a waiting probability of 0.4 and a little bit higher than the previous removal probability (which is normal because they have a fixed load of 16 in both cases and as w decreases, μ increases). We observe that the one with the highest waiting state takes more time to grow. This is due to the fact that there are more possibilities to be in a waiting state and then, the arrival of demands, and in consequence the gradual increase of the load in the network, is a less frequent event. Then we have to wait a higher amount of time to obtain the same result as another model with no waiting state or a lower waiting state, for example, 0.4 . The curve with $w = 0.4$ grows more rapidly and reaches an stationary regime, that is to say, when the majority of the results turn around the load, is reached at 1500ms while for $w = 0.9$ it comes at 4000ms. For lower charges, for example 4, represented in the graphic, the difference is not that clear but we still see that for $w = 0.95$ we get the mean at 1000ms and for $w = 0.4$ it is at 500ms.
And from those observations we understand why there is an error in the calculation of the mean load in the OCS network: the transitory regime add an error, the higher the waiting state probability is, the higher the bias with the theory result is.

The objective now is to try to correct this error using another traffic generator that could optimize the previous situation. For this, we have created the Mode 6 that, as explained in section 5.1, can generate 2 or 1 arrival and demands or just nothing. The background of the simulation is the same as in the previous one but this time we will fix $N = 32$ because the differences can be easily observed as the network takes very long to get charged (as commented before). We will fix as well $w = 0.4$ and $w = 0.9$ to distinguish clearly both cases. We simulate for 2000 events.

The results are shown in Figure 7.25. The graphic representation here has the same axis as in the Mode5 simulation. We distinguish in blue the results for the mode 6 and in red, those for the mode 5.We find again a transitory regime for both models even if there is an essential difference: the growth of the mode 6 with second order

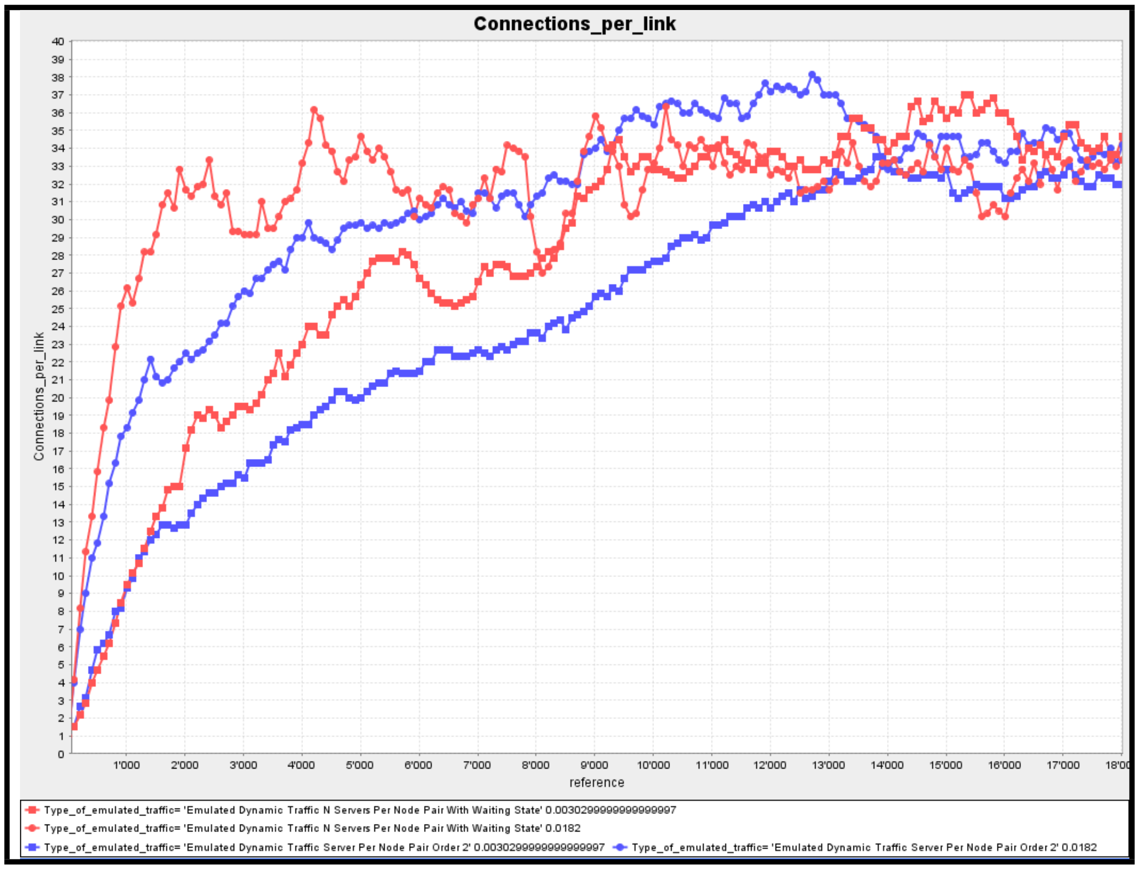


Figure 7.25: Load in networks using Mode5 and Mode 6 for different waiting state

is done more slowly and we get a more ”soft” curve than for the mode 5. From here we can say that even if 6 is more slow in the load race, it maintains a rhythm and we do not observe abrupt changes (as there are for the mode 5). This implies a more controlled load of the network, arriving to reach the charge progressively. Furthermore, once the permanent regime begins, mode 6 tends to maintain it whereas mode 5 experiments more fluctuations around the mean rate. On the other hand mode 5, the one with waiting state, seems to reach more quickly the permanent regime (even if t fluctuates more around it). To resume, we would say that even if mode 6 has a lower response, the control of the load in the network and the progressive advances could be positive in the network experiment. Moreover, we have not studied yet which would be the bias of the transitory stage in a whole simulation.
To complete this comparison we develop another experiment.

We make another experiment to judge again the behavior of mode 5 and mode 6. Both methods have the objective to maintain the load in the network stable. The results representing mean the number of connections per link for removal probabilities μ of 0.1, 0.3 and 0.6 with a respective arrival probabilities λ of 0.7, 0.5 and 0.2 and a fixed probability to wait w of 0.2 are in Figure 7.26. We observe that as we have

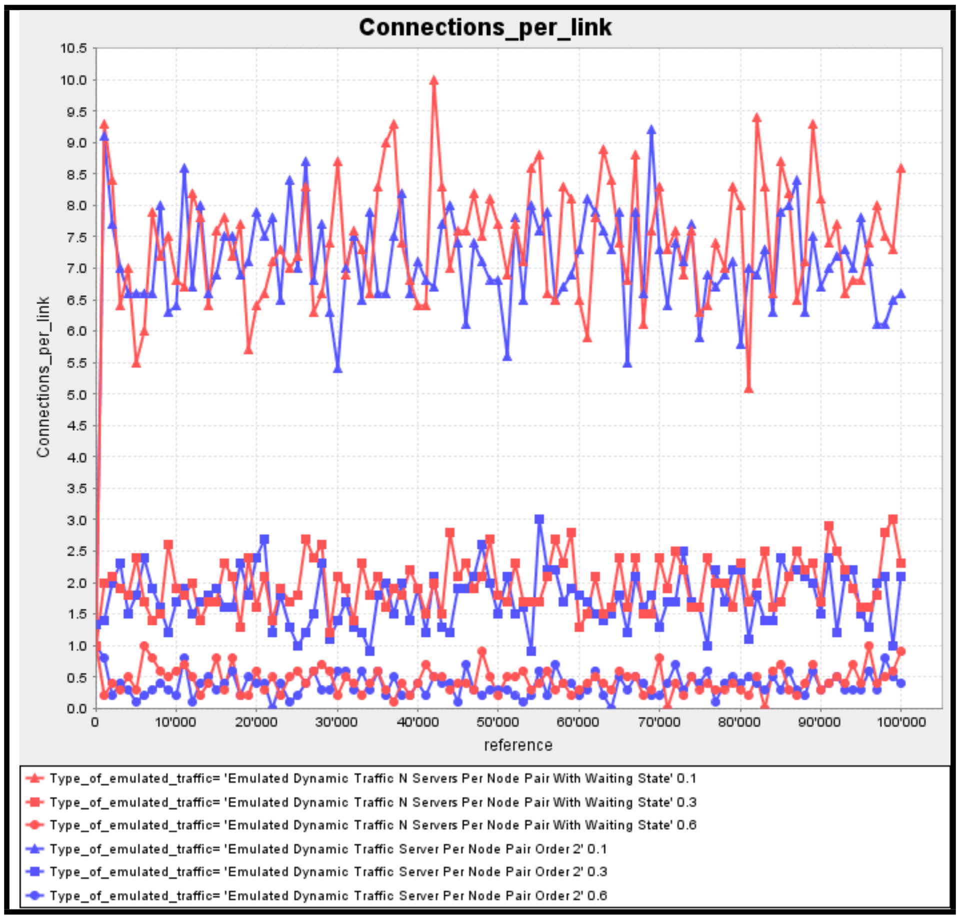


Figure 7.26: Mean connections per link for mode 5 and 6

done a simulation with a lot of events (10000), we cannot appreciate the transitory regime (except for the highest load where we can recognize the passage from a low value to highest one). In the Table 7.4 we have calculated the mean values of the obtained results for each mode. For the mode 5, as each link is a M/M/∞ system, we know that the theoretical mean number of elements per link should correspond to $\frac{\text{arrival probability}}{\text{removal probability}}$. Anyway, as we have seen, whether for mode 5 and 6 the transitory regime introduces a bias in the mean obtained with the simulation results. The results on the table reflects those differences but we observe that anyway, mode 6 results keep closer to the theory than mode 5. The results of the simulation seem to be a little bit more accurate and close to what we expect to for the system allowing second order.

We can also take a look to the traffic offered (estimated) for the two modes in Figure 7.27. As we expect a higher control of the demands using the model with order 2, it has sense that the offered traffic is lower in this case (or at least, more adapted to the theory and not disturbed by possible transitory stages). The higher

Removal probability / mode	0.1	0.3	0.6
5, waiting state	7.46	1.99	0.42
6, 2nd order	7.15	1.76	0.38
Theoretic $\frac{arrival_prob}{removal_prob}$	7.00	1.66	0.33

Table 7.4: Mean connections per link results for mode 5 and 6

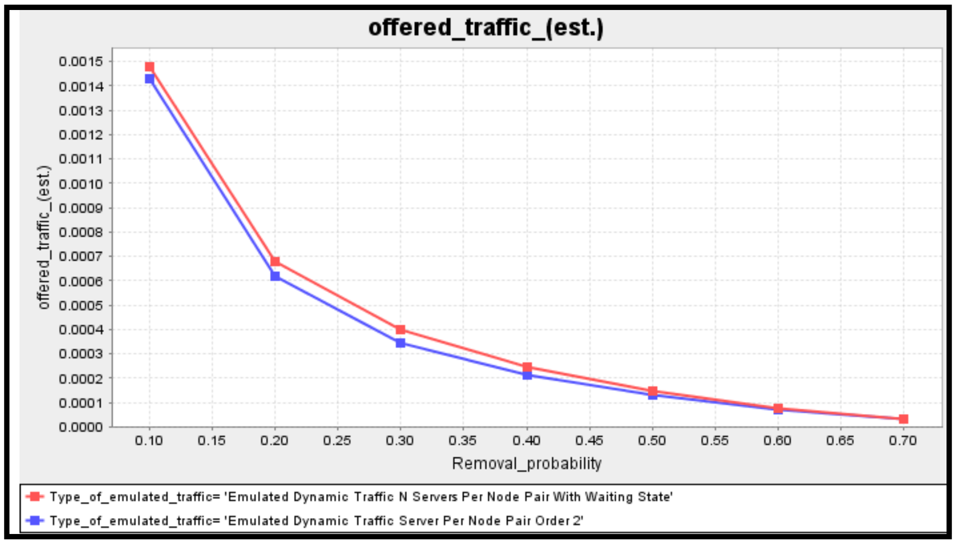


Figure 7.27: Mean estimated offered traffic for mode 5 and 6

distance between both modes is 15.8% when the removal probability is 0.3 (so the arrival probability is 0.5 and the waiting state 0.2).

The final objective of this study of emulated traffic generators is to furnish to the network a control in the mean number of elements as well as in the offered traffic accomplishing specific requirements. For example, if we know we have limited resources, thus m wavelength channels in each link, we will be able to adapt the traffic generator in order to get a mean of $m - x$ to be sure that all demands are coped. Another interesting feature is that are able to control as well the charge in each link if we configure correctly the matrix of thresholds in systems that permit it. To conclude, we have created a tool in charge of manipulate the occupation in the network.
We cannot decide which one is the best or most performing traffic generator because each one could be useful in particular situations.

7.4 Energetic Consumption Simulation Results

Since we had develop the possibility to obtain three types of virtual topologies, get some related metrics and obtain energetic results (Cf. chapter 6) we are able to obtain and contrast:
- metric and energetic results for a given physical topology using different virtual topologies
- metric and energetic results for different physical and virtual topologies
Those metrics and results will allow us to calculate energetic consumption and conclude in which situation the energetic cost is the lowest hence the most interesting to the users of the studied OCS networks.

7.4.1 Virtual Topologies Metric Comparison

To get results of metrics of a physical topology for different virtual topologies we will fix a physical topology which is called *test.xml* in the Figure 7.28. We have chosen this 6-node topology because it is partially meshed, there is a clear "central" node (the node 4) which is necessarily used many times at least to the get to node 5. Moreover, the link $(4, 5)$ will also be highly used. We remind that "metrics" represent the needed elements in the network that will contribute to the energy consumption balance.

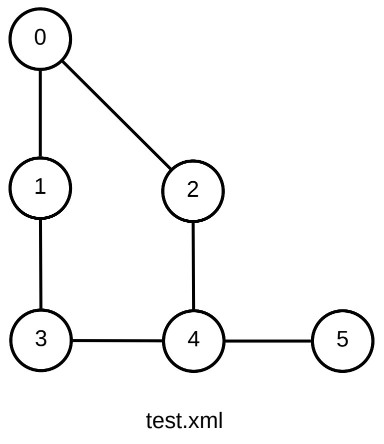


Figure 7.28: `test.xml` physical topology

To run the simulation we provide the network of a uniform traffic matrix containing `Traffic_value` called `val` as traffic rate for each node pair. At the beginning we only consider traffic rates from 0.1 to 1.5 by steps of 0.1 because we find a particular situation. Later we will study general tendencies for higher traffic rates. The metrics

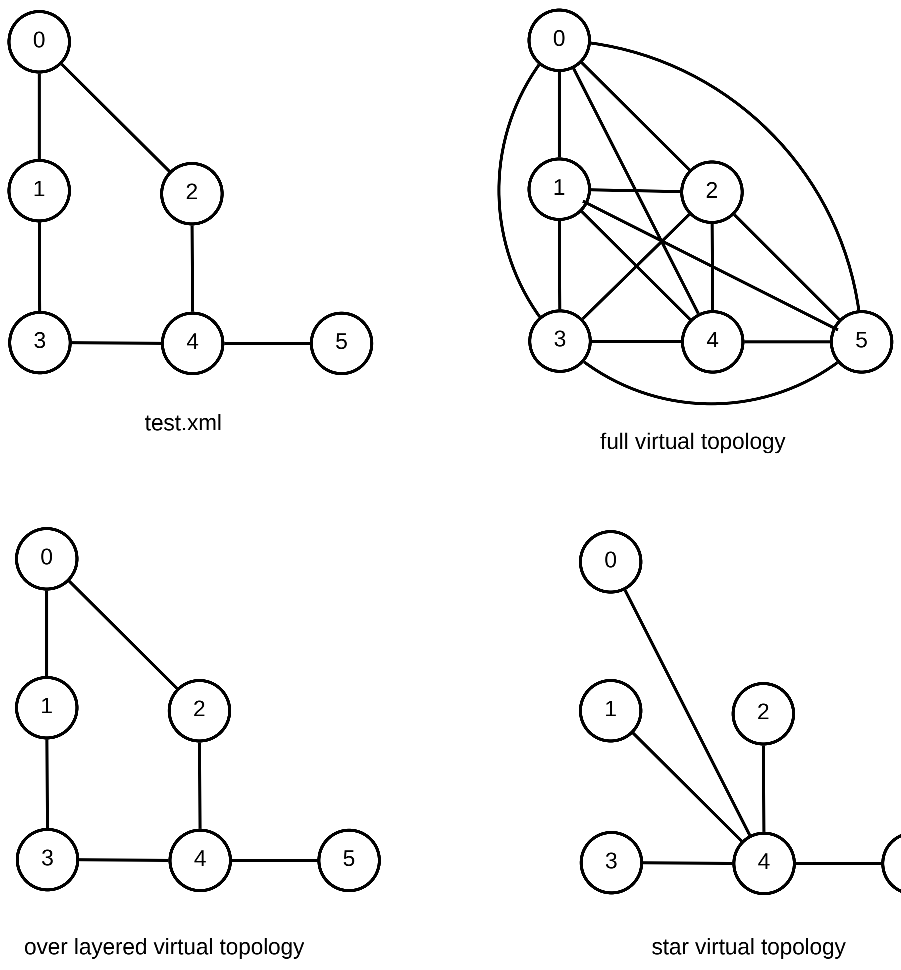


Figure 7.29: Selected physical topologies for the energy consumption study

for the studied physical topology in Figure 7.28 are calculated for the virtual topologies presented in section 6.1 and following the way the are explained in section 6.2. The resulting virtual topologies of the virtual topology `test.xml` are represented in Figure 7.29.
The resulting metrics for those topologies are:

- Connections:

  In Figure 7.30 we observe that for traffic values inferiors to 0.5 the highest number of connections corresponds to the full virtual topology that needs to give a connection for each node pair ($n(n-1)$ node pairs). After this stage, the number of connections for the two other virtual topologies (over layered and star) overcomes the full one because the full virtual topology has a direct path for each node pair while in the star and over layered topologies paths have to

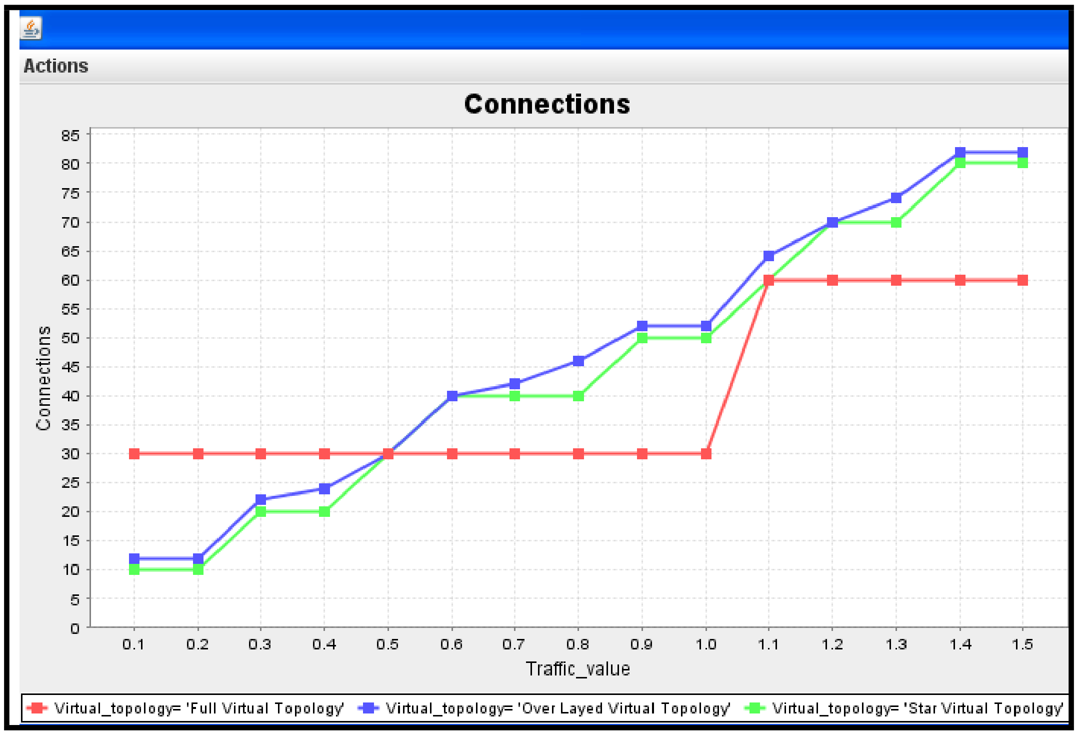


Figure 7.30: Number of connections for `test.xml`

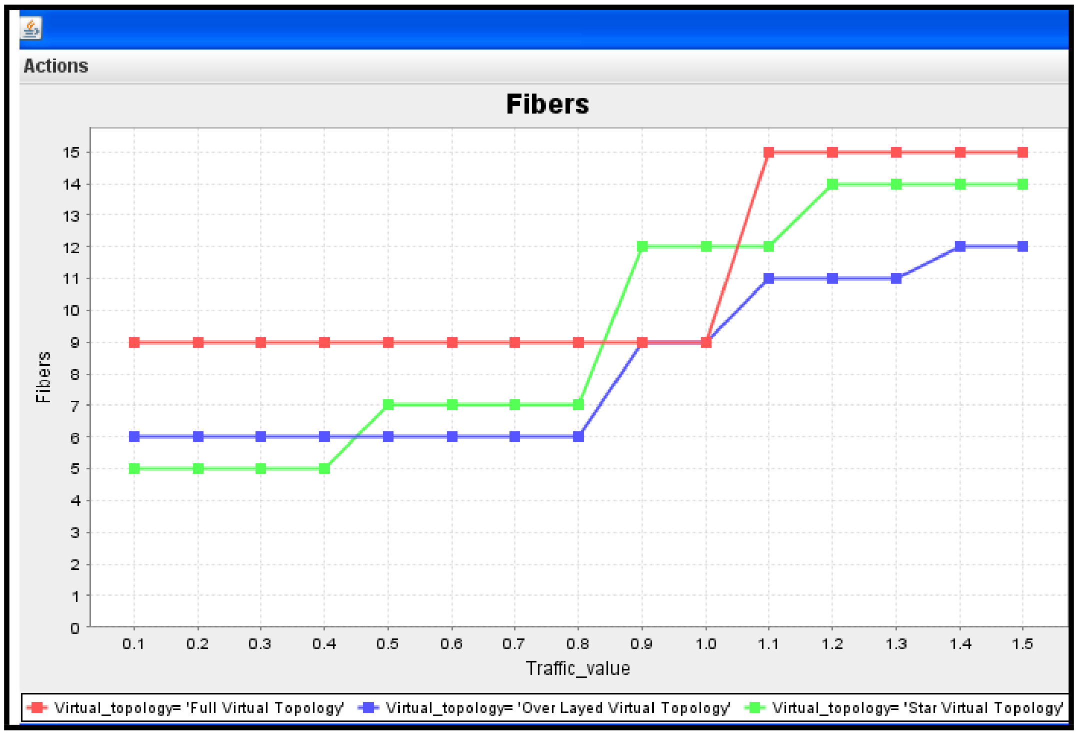


Figure 7.31: Number of needed fibers for `test.xml`

occupy more than one link per node pair increasing the links capacity usage. We see as well that the full virtual topology doubles the number of connections as the traffic value passes 1: since the capacity of a lightpath link (1 in this case) is surpassed, another one is needed to transmit all the traffic. The two others follow their progression without abrupt changes.

- Fibers:

  Figure 7.31 shows that in almost all cases, the higher number of needed fibers are requested by the full virtual topology because, even if the number of connections is lower, when routing the virtual node pairs in the physical layer, each connection takes up, at least, a whole wavelength causing the need of a high number of fibers. And here we face the first tradeoff between number of connections / number of fibers : the virtual topology requiring in general the less number of connections, requires the highest number of fibers.

- Router Ports:

  We find again the fact that until a traffic value of 0.5 the full virtual topology would need more router ports but since then it is the opposite. This situation is due to the fact that in the full virtual topology, if the traffic demand is lower than half of the maximum capacity of a wavelength then the rest of the capacity is useless and we are wasting resources. On the other hand, when the traffic rate is 0.5 or more, star and over layered virtual topologies need more and more resources to route and transmit the traffic.
  Another remarkable thing is that the star and the over layered virtual topologies do not show big differences among them.

- Routed Unities:

  The routed unities in Figure 7.33 maintain a linear progression because they are independent of the components needed in the network (fibers or router ports) and only depend on the traffic value and on the routing in the virtual topology (that is why they are different).

- Comparison between the number of connections and the number of wavelength channels needed in physical links:

  Using the same scale (y axis) we represent at the same time the number of connections (calculated in the virtual topologies) and the resulting number of wavelength channels needed in all the physical links. The results are shown in 7.34: the solid lines represent the connections and the dotted ones the wavelength channels. Obviously we need more channels in the physical links than connections except in the over layered virtual topology because this one is the same exact topology as the physical one and that for that reason there the solid and the dotted line are the same.

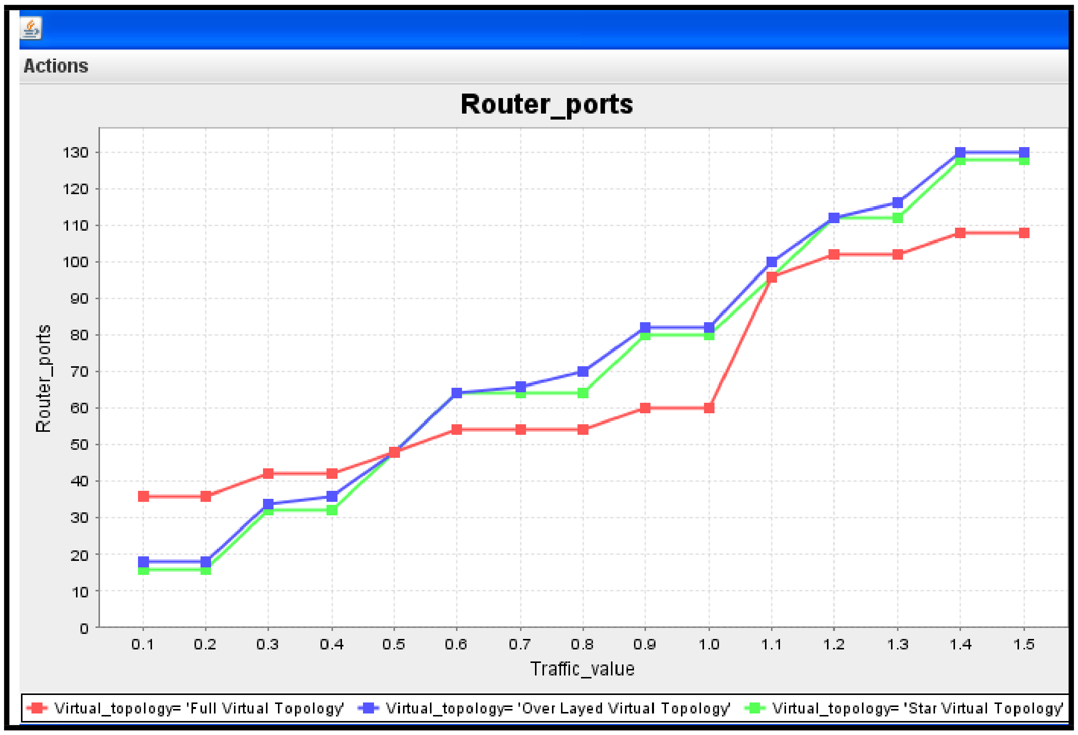


Figure 7.32: Number of needed router ports for `test.xml`

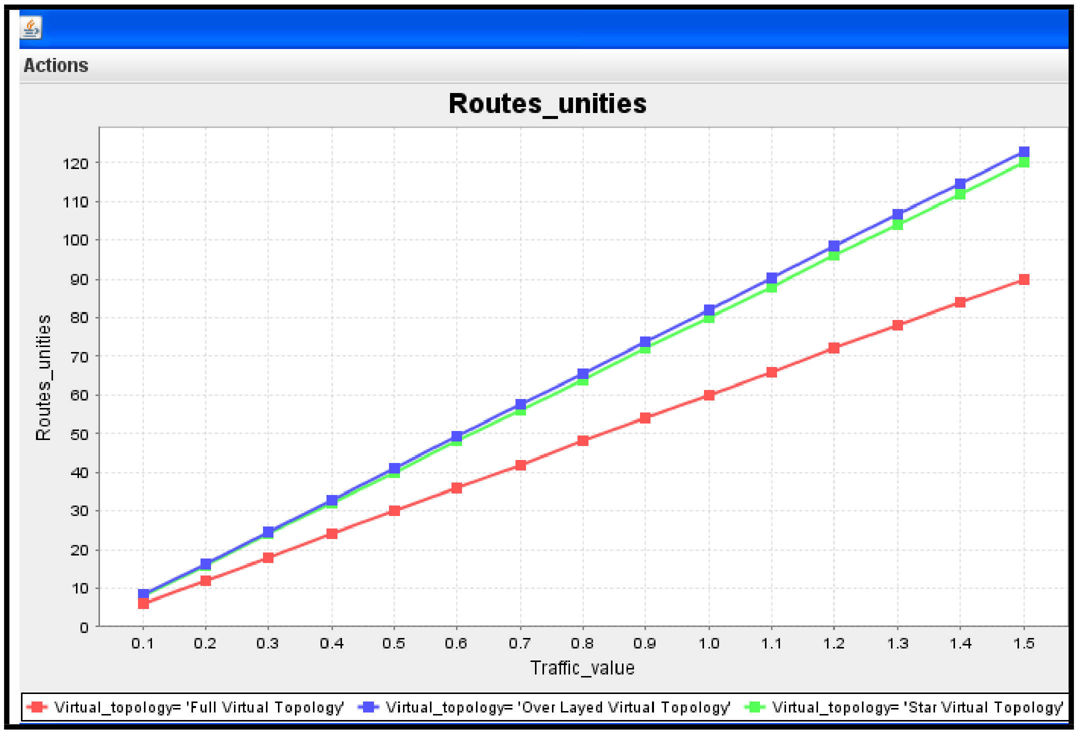


Figure 7.33: Number of routed unities for `test.xml`

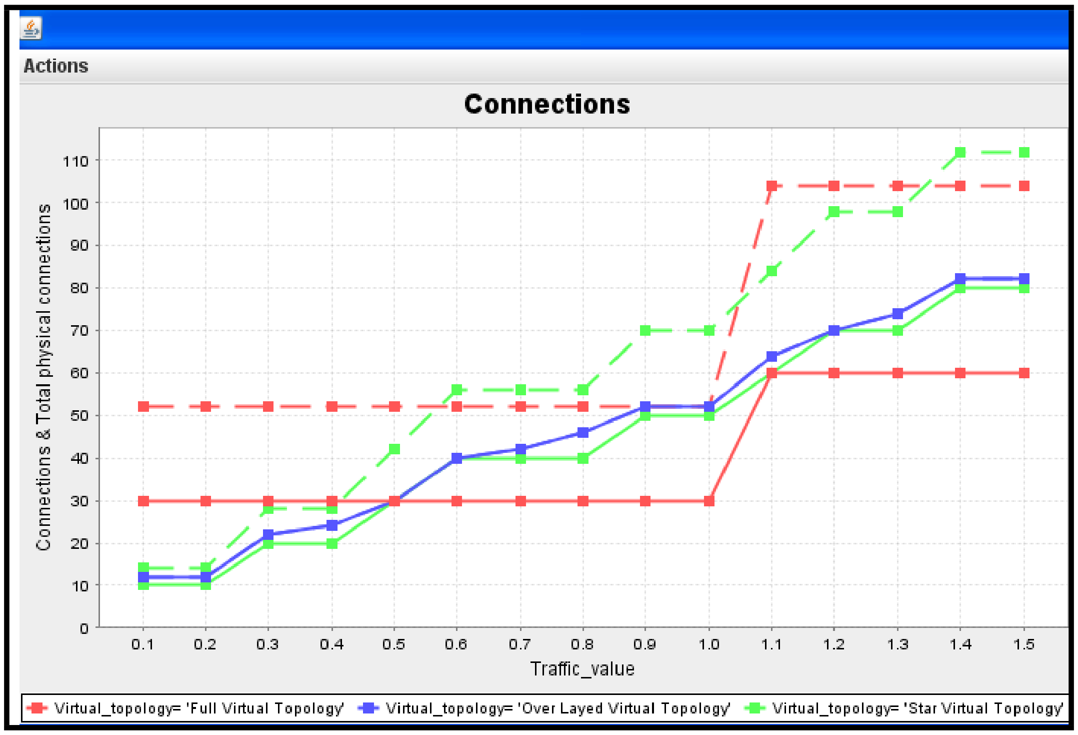


Figure 7.34: Comparison between number of connections and wavelength channels for `test.xml`

- Comparison between the number of router ports and wavelength channels in physical links:

  In Figure 7.35 are represented in solid lines the router ports and in dotted lines the wavelength channels needed. The most important remark is that even if the over layered is the one that requires the least number of wavelength channels it is the one that requires the higher number of router ports.

- Energy results and comparison with the number of router ports:

  In this case, we get in Figure 7.36 the results of the superposition, even if they are not using the same axis, of the consumed energy (in solid lines) and the number of needed router ports (in dotted lines). Here it is to corroborate that router ports are those who consume the higher energy in optical networks. Indeed, the energy scale is approximately 1100 times the router ports scale while each router port consumes 1000W (Cf. 6.1). Then, the resulting energy consumption value has almost the same shape as the router ports curve except a scale factor so we see that they clearly have the main role in the energetic consumption.
  The comparison between the three virtual topologies conducts to conclude that for this specific physical topology and for a traffic demand lower than 0.5 for all the node pairs, the best solution would be the star virtual topology, before

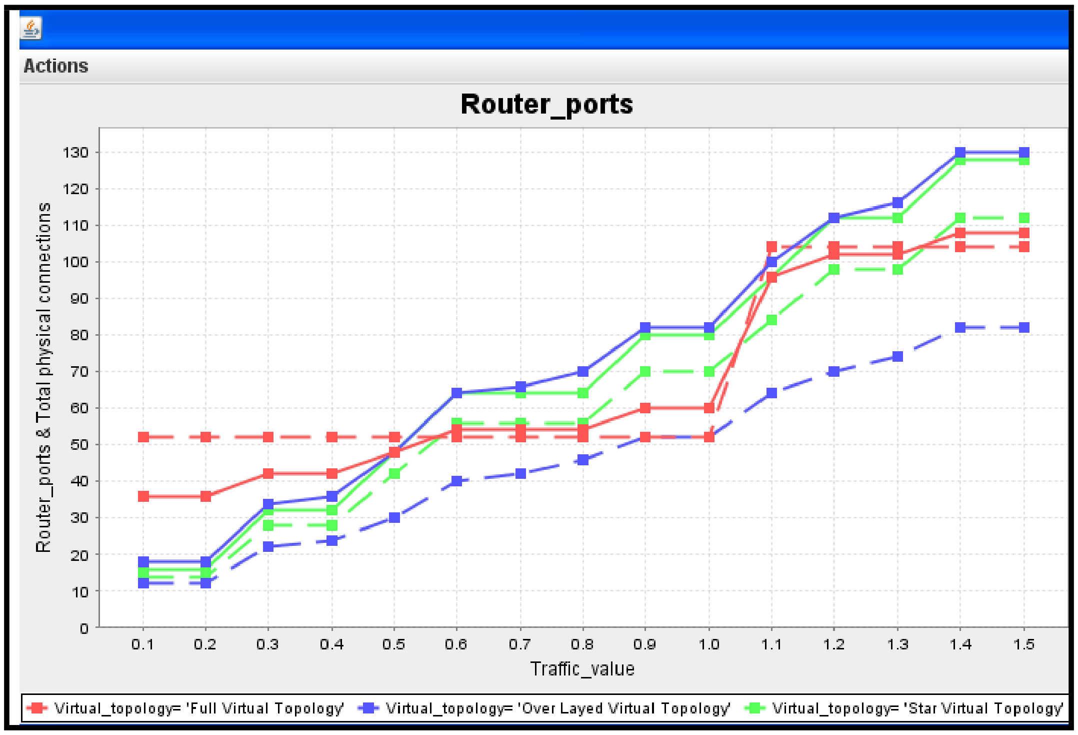


Figure 7.35: Comparison between number of router ports and wavelength channels for `test.xml`

that value, the full virtual topology would be recommended.

If we pay attention at what happens when the traffic reaches a traffic rate of 10 for each node pair we get a general idea of which virtual topology is suitable to consider to save energy.

Figure 7.37 represents the evolution of how many fibers are needed for the `test.xml` physical topology based on the three analyzed virtual layers. As explained before, for very low traffic rates the star virtual topology seems to be the best solution but since we increase little by little the traffic rate, the number of fibers grows until surpassing the two others since the traffic rate is higher than 1.7. On the other hand, the over layered topology is the one the need the lowest number of fibers in almost any case (unless for very low traffic rates as said before and because sometimes the full meshed virtual topology has the same results as the star).
The number of wavelengths channels has almost the same interpretation.

Nevertheless, if we take a look to 7.38 which represents the energy consumption we conclude that from a traffic demand higher than 0.5, the more performing vir-

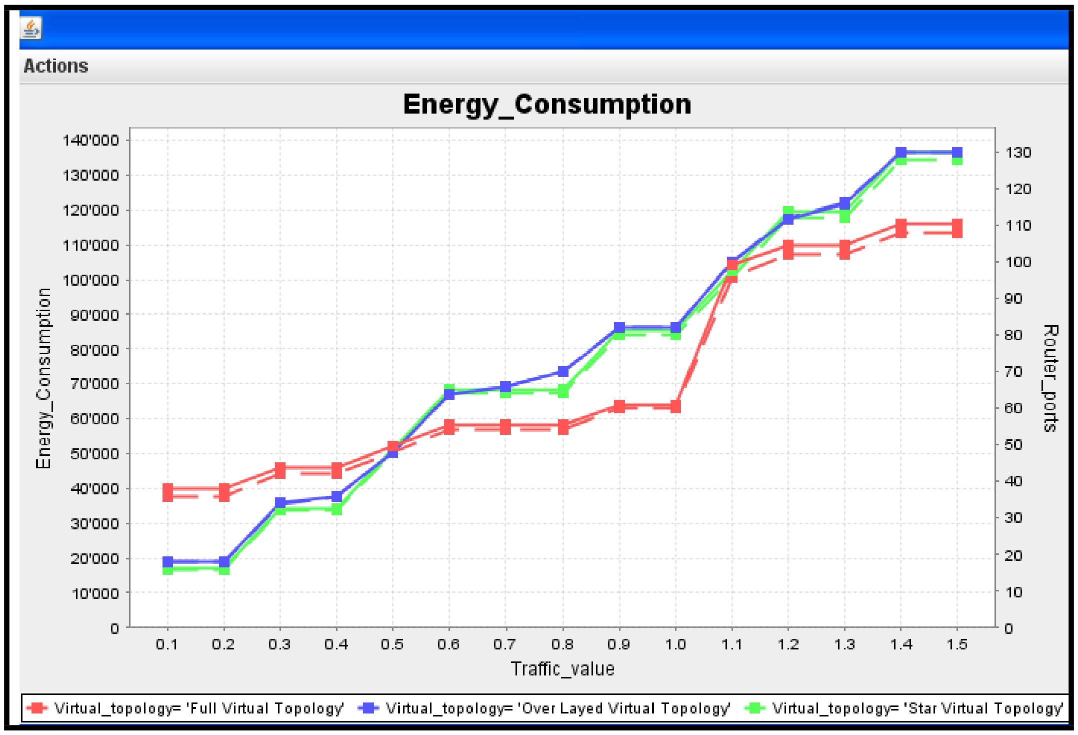


Figure 7.36: Comparison between the energy consumption and the number of router ports for `test.xml`

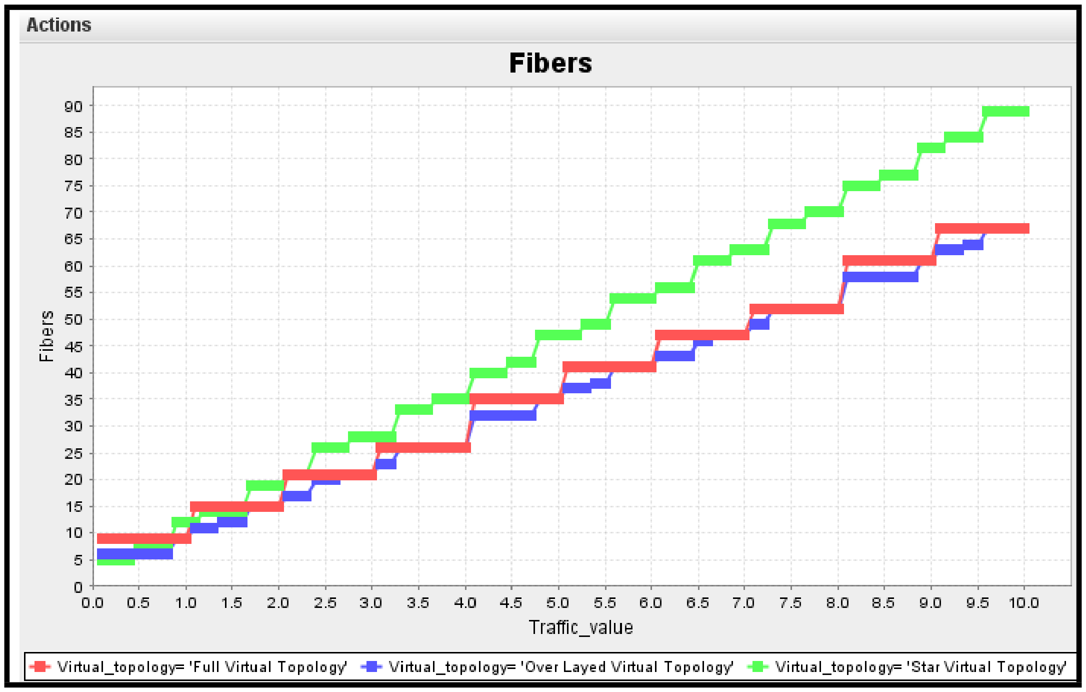


Figure 7.37: Number of needed fibers for `test.xml` with high traffic rates

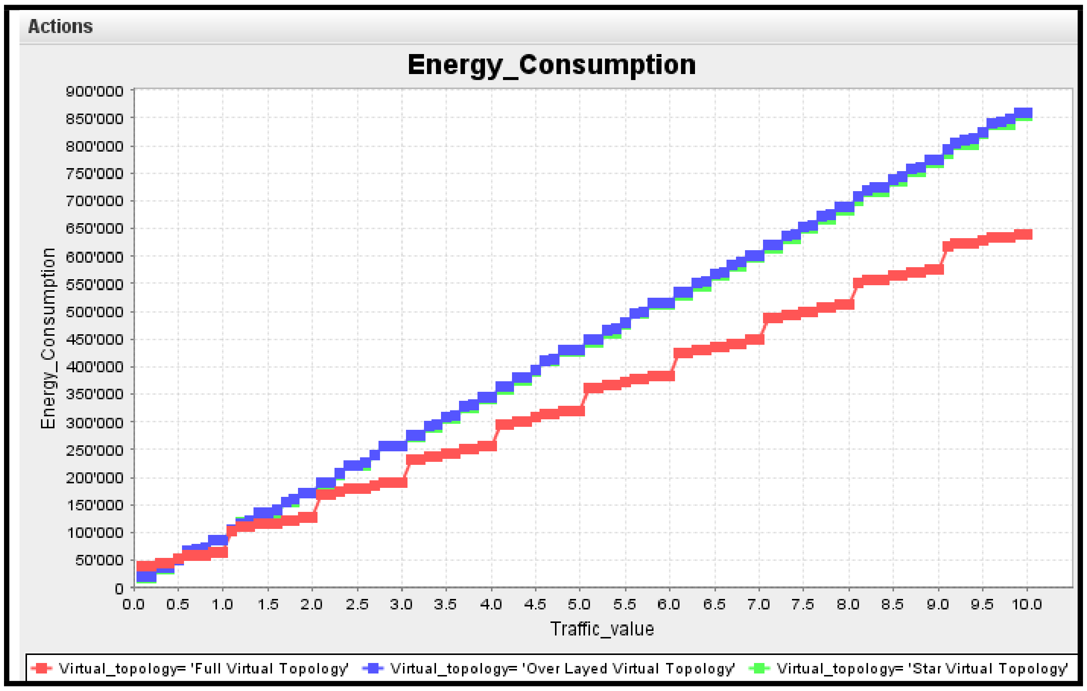


Figure 7.38: Energy consumption for `test.xml` with high traffic rates

tual topology would be the full meshed one. We corroborate again that even if the number of needed fibers and wavelength channels participate to the energy balance, the number of router ports is the heaviest component, energetically speaking. Then, looking at this Figure we can guess that the router ports graphic has almost the same shape as this one and, in consequence the number of router ports for the over layered topology is close to the star topology and both are quite separated from the full virtual topology.

7.4.2 Physical and Virtual Topologies Metrics Comparison

Six different physical topologies have been chosen (Figure 7.39) to be studied. They have the particularity that they all have six nodes but they are meshed in different degrees to compare their performance. This time we study traffic rates from 0.1 to 5 because from 1 onward, we obtain the same pattern. For each topology the over layered virtual topology will have the exact same shape as the physical one and the full virtual topology is the same for all the topologies. The special parameter we have to consider is the central node of the physical topology when constructing the star virtual topology (Table 7.5). We have said before that in our implementation it is the node with highest degree (even if other criterions could be considered in future research). If various nodes in the topology have the same degree, the lower index is the chosen one.

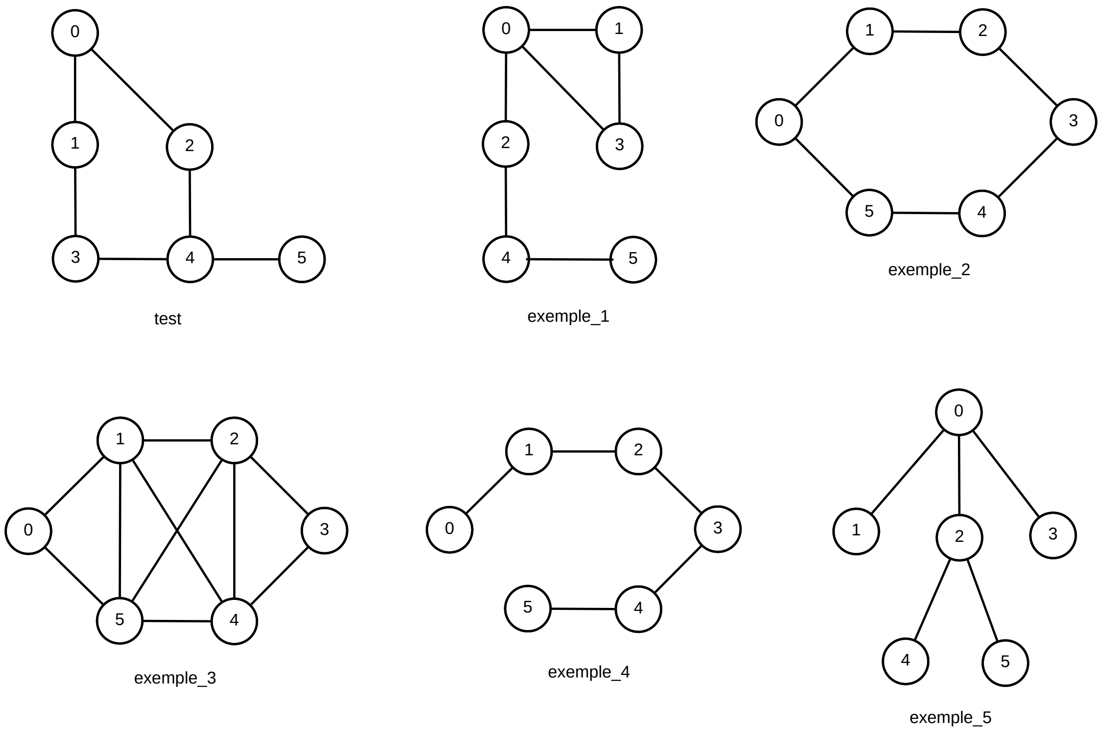


Figure 7.39: Selected physical topologies for the energy consumption study

physical topology	central node
test	4
exemple_1	0
exemple_2	0
exemple_3	1
exemple_4	1
exemple_5	0

Table 7.5: Central nodes of star virtual topologies

In Figure 7.40 there is graphically represented the number of connections for the chosen physical topologies. We observe that the red and green solid lines representing full and star topologies are the same for all the physical studied topologies. This is normal because connections is a metric from the virtual representation and all the 6 physical topologies have exactly the same resulting star and full topology. On the other hand the over layered virtual topology let us see some differences. In general, the number of connections for this specific virtual topology increases as the physical topology (and then, the virtual) is less meshed. When a demand is routed, the more links the topology has, the shorter is the resulting lightpath, the less charged links are

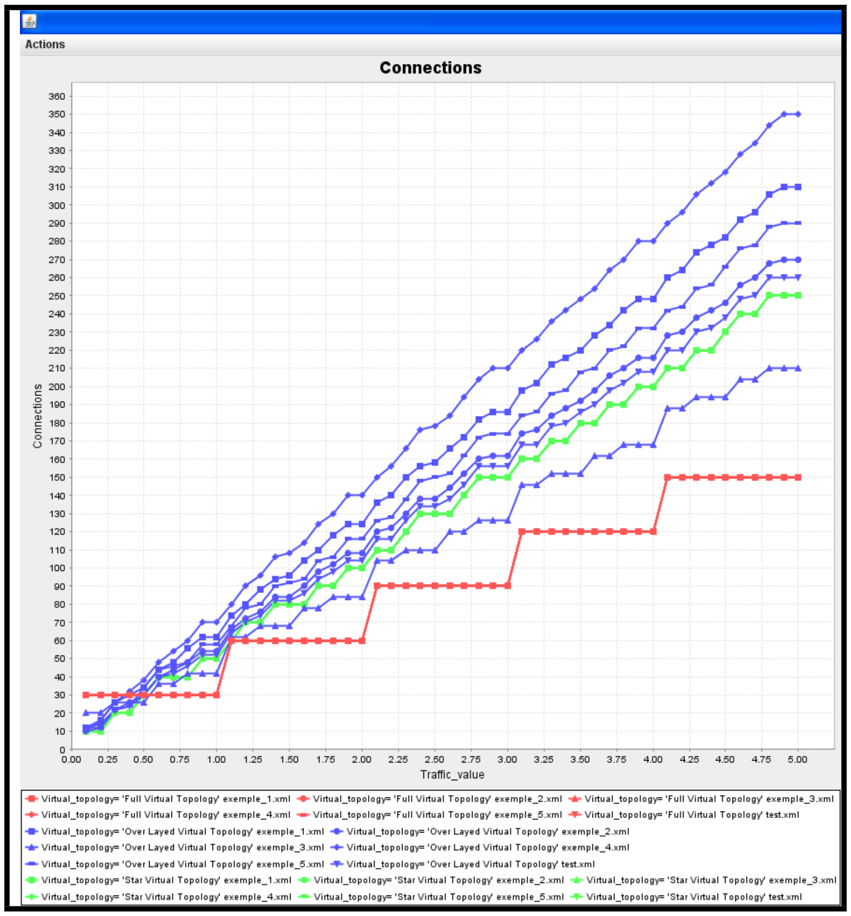


Figure 7.40: Connections for different physical topologies with high traffic rates

and in consequence, the less connections are needed. Furthermore, the green solid line is between of the blue ones which allows us to think that for highly meshed topologies it is suitable to choose the over layered scheme rather than the star one (dealing with connections).

Anyway, we find out that the general tendency is broken for little traffic rates. We represent in Figure 7.41 a zoom in the conflictive zone. As said before, full and star topologies crosses at a rate of 0.5 as well as two over layered cases. The rest of cases cross and surpasses full and star topologies sooner (already from 0.35 for the lowest meshed topology) except in the case where the topology (`example_3`) has more links than in the test one(`test`).

The Figure 7.42 shows the needed fibers for `exemple_3` and `example_4` physical and virtual topologies. We have chosen those two graphics because they are the most and least meshed. In this case we get a more heterogeneous graphic where each physical topology has different values for each virtual topology. The lines with dots represent the `example_4` thus, the line topolgy, and we observe that all those lines

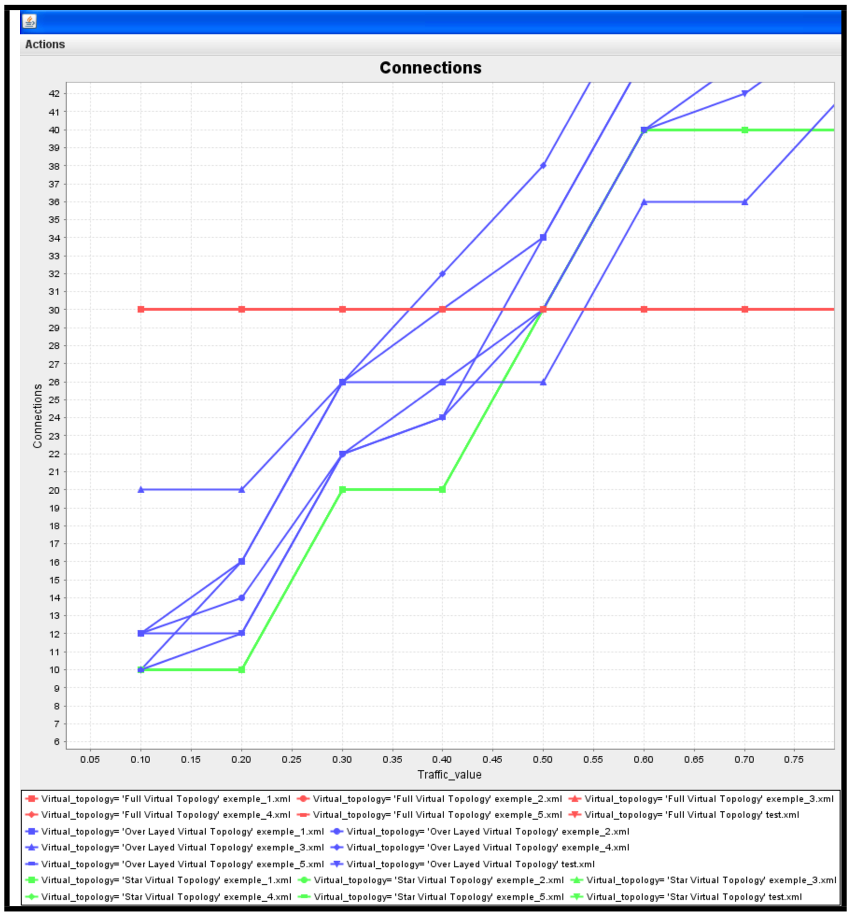


Figure 7.41: Connections for different physical topologies with low traffic rates

surpass the squared ones (which represents `exemple_3`, the almost full meshed topology). It is reasonable to say that we will need more fibers in a line topology than in an almost full meshed one because to reach nodes the same links are used more times.
The blue squared line is hidden behind the red squared one thus, the full and over layered virtual topology are indistinct for this almost full meshed topology. Is it that for a certain % of link connections it is the same to consider over layered or full virtual topology? We will not discuss this in this part.
In both cases (except at the beginning) the over layered virtual topology requires the lowest number of fibers.

If now we take a look to the Figure 7.43 where the energy consumption is represented for the whole physical topologies studied we observe that:
- no matter the type of physical topology, from a traffic rate higher than 0.5 (in almost all cases except for the `exemple_3`), the full virtual topology seems to be the most performing energetically speaking

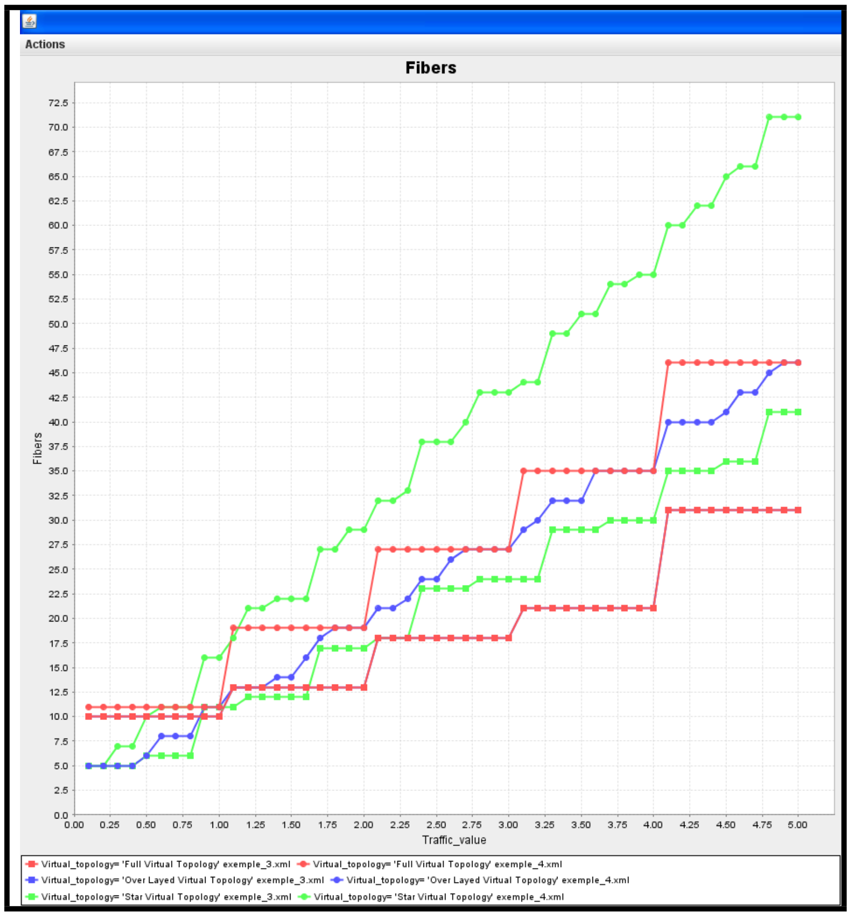


Figure 7.42: Number of fibers for `exemple_3` and `example_4` physical topologies with high traffic rates

- for traffic rates lower than 0.5 (in almost all cases except for the `exemple_3`), the star virtual topology is the most suitable
- the lower a physical topology is meshed, the higher his energy consumption is
- the energy consumption for different virtual topologies can be more or less performing depending on the physical topology (i.e. for a line in the physical layer, the star virtual topology is more performing than in the over layered one meanwhile for an almost full meshed topology, it is the contrary).

Moreover, this figure shows that the virtual topology which has the most different results for some given physical topologies is the over layered where the difference between the highest and the lowest value for a traffic value of 5 is more than 25% whereas for the full topology is 3.2% and for the star 4.4%. In addition, the difference between values gets bigger as the traffic grows even if for full and star layers the increase is slow.

We conclude once again that the obtained results vary depending on the chosen inputs. Despite observing some tendencies for some physical and virtual topologies we cannot be sure that the will find the same patterns in other situations.

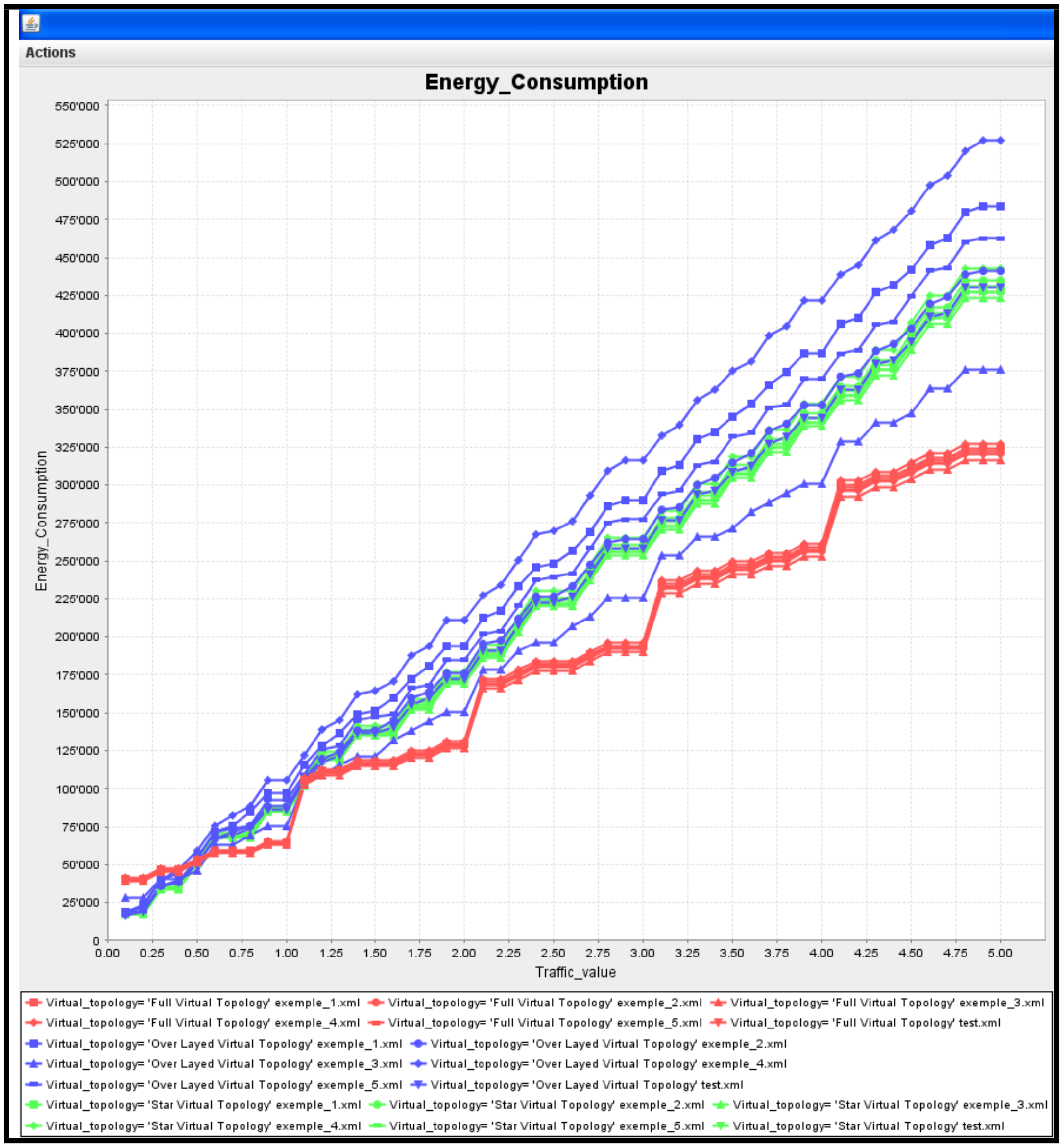


Figure 7.43: Energy consumption for different physical topologies with high traffic rates

Chapter 8

Conclusions and Further Research

8.1 Summary and Conclusions

Optical networks are constantly growing and asking for higher bandwidths in order to cope with all the IP-over-WDM applications and data traffic. Increase the transmission capacity or find solutions to take advantage of the already set up systems are the main objective to receive the higher number of demands. What is significant is that we are still far away from hitting the fundamental limits of capacity in optical fiber. While there are several obstacles and limits in the deployment of this technology, we will no doubt see the invention of new techniques that enable progressively higher and higher capacities, and the development of optical networks with increasing functionality.
Meanwhile, already existing techniques allow optical networks to make the most of the resources. In this case, we have dealt with Optical Circuit Switching technology and we have tried to analyze in which way the possibilities offered by this technique contributes to the performance of the network.
Moreover, worried by the negative effects of the massive usage of components and the increasing waste of energy because of the growth of networks, we have outlined a study about energy consumption in order to propose possibilities to find the most suitable way to save some energy.

The most feasible way to conduct those studies is the simulation. And that is why we have implemented a series of elements creating a modular framework to carry out the performance study. This idea was born because nowadays it does not exist a framework for this specific technology and because it is a perfect complement for Javanco already including an Optical Burst Switching framework. In fact, OCS and OBS technologies are struggling to be the technology for future networks. That is

why it is important to highlight the advantages and disadvantages of each technique. A big number of inputs, configurations and algorithms have been developed and further validated to go on with simulations. The graphical interface has supported our studies representing the output data in a clear and understandable way.
Through all the studied cases, we have understood that there is no single optimal combination for all scenarios. The right algorithms to apply to networks need to be chosen based on the factors participating on its operation from the inputs to the configurations and algorithms. If we translate this words to the real life situations we could say that there are too many tradeoffs to take into account in the communication between users and for the moment, we are not able to give to the population the absolute soltuion to permit to all users at any place, transmitting any things and at any time to be satisfied in terms of success, speed, reliability and cost and energetic economy.

But of course, the study of OCS switching technology could go further and further until find, maybe one day, the ideal combination of elements to guarantee a maximum of performance and satisfaction. Thanks to the developed tool in this work we are able to continue exploring what would happen in several situations.

8.2 Further Research

Optical networks still being a large domain to explore and develop. As the needs are changing lots of possibilities can be considered to continue researching and creating new solutions and optimizations to cover them.
The particular switching method developed in this work could be extensively completed in many ways. First, we could go further with the already built elements and find new case studies. Secondly, we could answer to questions found during the development of tools in this project. But we could add other configurations or consider more algorithms, many of them explained in the background chapter, involving protection or grooming methods. The most important thing to take into account is that the created framework allows the easy adding of new elements without interfering the already existing ones and highly contributing to the performance study in optical networks.
In the literature we find other ideas to continue developing: furnish the AON with sparse wavelength conversion [66], create routing and wavelength assignment algorithms jointly and adaptive [67] or even make dynamic wavelength routing based on path and neighborhood link congestion [68]. Other innovative ideas as to set up lightpaths with different bit rates [69] or create a time-driven optical switch based on an universal time clock to facilitate frame switching of a given wavelength without requiring frame content to be processed [70] could be also considered as guidelines for

the implementation of new ideas.

In the energetic domain we have dealt with things to get a general idea but, of course, the pursue of the energetic savings has still a lot of work ahead. For example, in this case we have considered only three virtual topologies although other topologies maybe could be more performing. Many new ways to green the Internet are being developed because we are not only dealing with saving energy but with saving humanity from human damages.

Appendix A

Diagrams' key

Below there is the key of the diagrams representing program structures. We remark that the objects are placed in a coherent way: the objects extending other objects are placed below and so on.

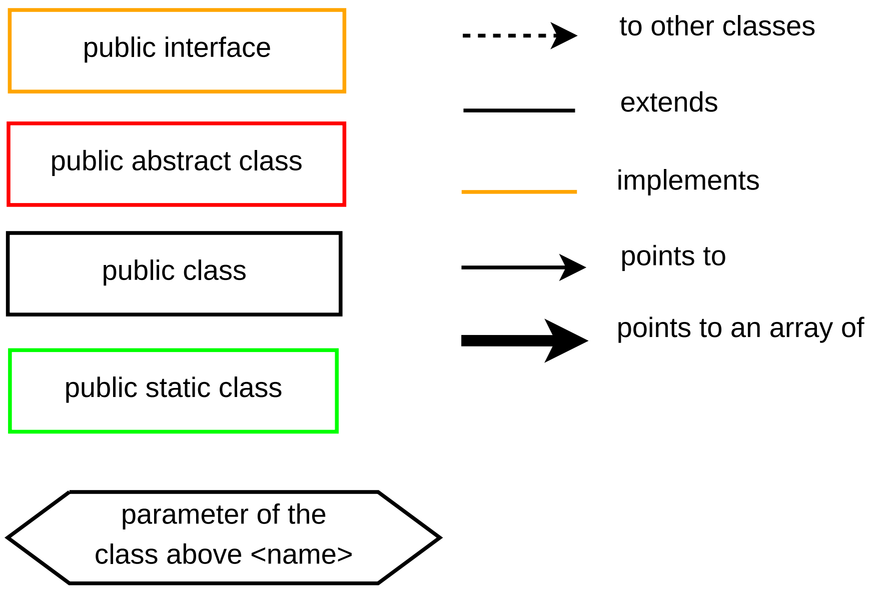


Figure A.1: Diagrams' key

Appendix B

Erlang B

The Erlang B formula is written:

$$B = \frac{\frac{A^n}{n!}}{\sum_{i=0}^{n} \frac{A^i}{i!}} = E_n(A) \tag{B.1}$$

and the graphical representation for an offered traffic $A = [1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}, \frac{1}{7}]$ and a number of channels $n = [1, 2, 3, 4]$ is represented in Figure B.1 with numerical data transcribed in Table B.1

Channels	P_b $(T = 1)$	P_b $(T = 0.5)$	P_b $(T = 0.33)$	P_b $(T = 0.25)$	P_b $(T = 0.2)$	P_b $(T = 0.17)$	P_b $(T = 0.14)$
1	0.500	0.333	0.250	0.200	0.166	0.143	0.125
2	0.200	0.076	0.040	0.024	0.016	0.011	0.008
3	0.062	0.012	0.004	0.002	0.001	0	0
4	0.015	0.001	0	0	0	0	0

Table B.1: **Theoretical** blocking probabilities

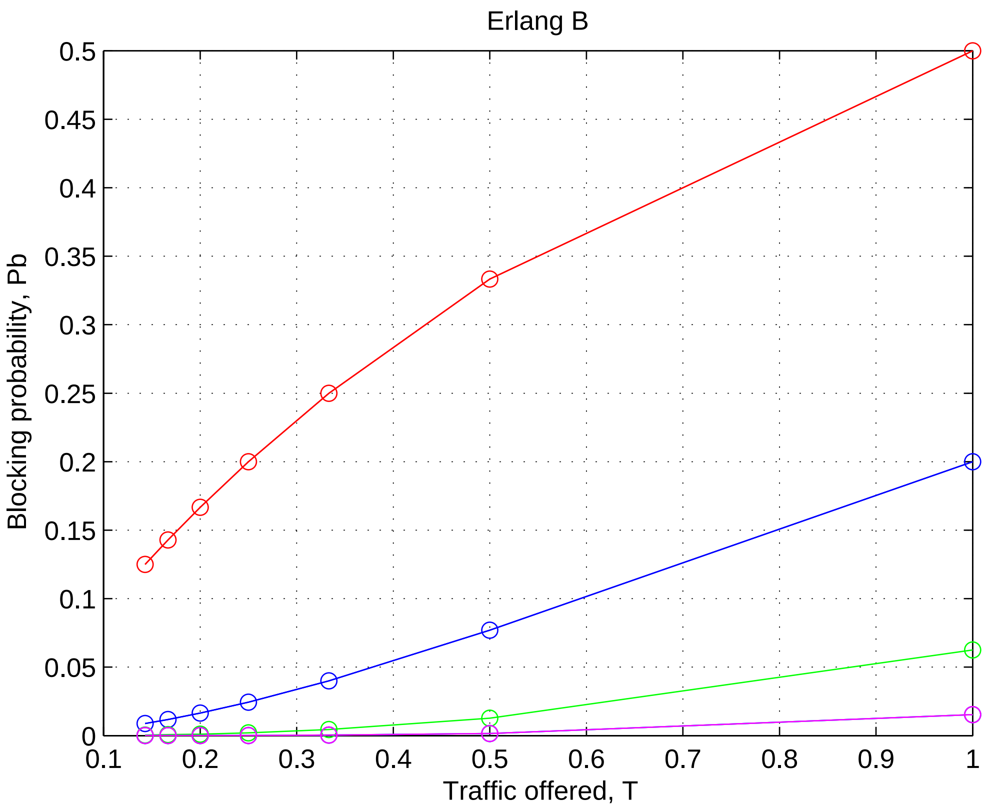


Figure B.1: Erlang B representation for the studied values

Erlang B Traffic Table

Maximum Offered Load Versus B and N
B is in %

N/B	0.01	0.05	0.1	0.5	1.0	2	5	10	15	20	30	40
1	.0001	.0005	.0010	.0050	.0101	.0204	.0526	.1111	.1765	.2500	.4286	.6667
2	.0142	.0321	.0458	.1054	.1526	.2235	.3813	.5954	.7962	1.000	1.449	2.000
3	.0868	.1517	.1938	.3490	.4555	.6022	.8994	1.271	1.603	1.930	2.633	3.480
4	.2347	.3624	.4393	.7012	.8694	1.092	1.525	2.045	2.501	2.945	3.891	5.021
5	.4520	.6486	.7621	1.132	1.361	1.657	2.219	2.881	3.454	4.010	5.189	6.596
6	.7282	.9957	1.146	1.622	1.909	2.276	2.960	3.758	4.445	5.109	6.514	8.191
7	1.054	1.392	1.579	2.158	2.501	2.935	3.738	4.666	5.461	6.230	7.856	9.800
8	1.422	1.830	2.051	2.730	3.128	3.627	4.543	5.597	6.498	7.369	9.213	11.42
9	1.826	2.302	2.558	3.333	3.783	4.345	5.370	6.546	7.551	8.522	10.58	13.05
10	2.260	2.803	3.092	3.961	4.461	5.084	6.216	7.511	8.616	9.685	11.95	14.68
11	2.722	3.329	3.651	4.610	5.160	5.842	7.076	8.487	9.691	10.86	13.33	16.31
12	3.207	3.878	4.231	5.279	5.876	6.615	7.950	9.474	10.78	12.04	14.72	17.95
13	3.713	4.447	4.831	5.964	6.607	7.402	8.835	10.47	11.87	13.22	16.11	19.60
14	4.239	5.032	5.446	6.663	7.352	8.200	9.730	11.47	12.97	14.41	17.50	21.24
15	4.781	5.634	6.077	7.376	8.108	9.010	10.63	12.48	14.07	15.61	18.90	22.89
16	5.339	6.250	6.722	8.100	8.875	9.828	11.54	13.50	15.18	16.81	20.30	24.54
17	5.911	6.878	7.378	8.834	9.652	10.66	12.46	14.52	16.29	18.01	21.70	26.19
18	6.496	7.519	8.046	9.578	10.44	11.49	13.39	15.55	17.41	19.22	23.10	27.84
19	7.093	8.170	8.724	10.33	11.23	12.33	14.32	16.58	18.53	20.42	24.51	29.50
20	7.701	8.831	9.412	11.09	12.03	13.18	15.25	17.61	19.65	21.64	25.92	31.15
21	8.319	9.501	10.11	11.86	12.84	14.04	16.19	18.65	20.77	22.85	27.33	32.81
22	8.946	10.18	10.81	12.64	13.65	14.90	17.13	19.69	21.90	24.06	28.74	34.46
23	9.583	10.87	11.52	13.42	14.47	15.76	18.08	20.74	23.03	25.28	30.15	36.12
24	10.23	11.56	12.24	14.20	15.30	16.63	19.03	21.78	24.16	26.50	31.56	37.78
25	10.88	12.26	12.97	15.00	16.13	17.51	19.99	22.83	25.30	27.72	32.97	39.44
26	11.54	12.97	13.70	15.80	16.96	18.38	20.94	23.89	26.43	28.94	34.39	41.10
27	12.21	13.69	14.44	16.60	17.80	19.27	21.90	24.94	27.57	30.16	35.80	42.76
28	12.88	14.41	15.18	17.41	18.64	20.15	22.87	26.00	28.71	31.39	37.21	44.41
29	13.56	15.13	15.93	18.22	19.49	21.04	23.83	27.05	29.85	32.61	38.63	46.07
30	14.25	15.86	16.68	19.03	20.34	21.93	24.80	28.11	31.00	33.84	40.05	47.74
31	14.94	16.60	17.44	19.85	21.19	22.83	25.77	29.17	32.14	35.07	41.46	49.40
32	15.63	17.34	18.21	20.68	22.05	23.73	26.75	30.24	33.28	36.30	42.88	51.06
33	16.34	18.09	18.97	21.51	22.91	24.63	27.72	31.30	34.43	37.52	44.30	52.72
34	17.04	18.84	19.74	22.34	23.77	25.53	28.70	32.37	35.58	38.75	45.72	54.38
35	17.75	19.59	20.52	23.17	24.64	26.44	29.68	33.43	36.72	39.99	47.14	56.04
36	18.47	20.35	21.30	24.01	25.51	27.34	30.66	34.50	37.87	41.22	48.56	57.70
37	19.19	21.11	22.08	24.85	26.38	28.25	31.64	35.57	39.02	42.45	49.98	59.37
38	19.91	21.87	22.86	25.69	27.25	29.17	32.62	36.64	40.17	43.68	51.40	61.03
39	20.64	22.64	23.65	26.53	28.13	30.08	33.61	37.72	41.32	44.91	52.82	62.69
40	21.37	23.41	24.44	27.38	29.01	31.00	34.60	38.79	42.48	46.15	54.24	64.35
41	22.11	24.19	25.24	28.23	29.89	31.92	35.58	39.86	43.63	47.38	55.66	66.02
42	22.85	24.97	26.04	29.09	30.77	32.84	36.57	40.94	44.78	48.62	57.08	67.68
43	23.59	25.75	26.84	29.94	31.66	33.76	37.57	42.01	45.94	49.85	58.50	69.34

44	24.33	26.53	27.64	30.80	32.54	34.68	38.56	43.09	47.09	51.09	59.92	71.01
45	25.08	27.32	28.45	31.66	33.43	35.61	39.55	44.17	48.25	52.32	61.35	72.67
46	25.83	28.11	29.26	32.52	34.32	36.53	40.55	45.24	49.40	53.56	62.77	74.33
47	26.59	28.90	30.07	33.38	35.22	37.46	41.54	46.32	50.56	54.80	64.19	76.00
48	27.34	29.70	30.88	34.25	36.11	38.39	42.54	47.40	51.71	56.03	65.61	77.66
49	28.10	30.49	31.69	35.11	37.00	39.32	43.53	48.48	52.87	57.27	67.04	79.32
50	28.87	31.29	32.51	35.98	37.90	40.26	44.53	49.56	54.03	58.51	68.46	80.99
51	29.63	32.09	33.33	36.85	38.80	41.19	45.53	50.64	55.19	59.75	69.88	82.65
52	30.40	32.90	34.15	37.72	39.70	42.12	46.53	51.73	56.35	60.99	71.31	84.32
53	31.17	33.70	34.98	38.60	40.60	43.06	47.53	52.81	57.50	62.22	72.73	85.98
54	31.94	34.51	35.80	39.47	41.51	44.00	48.54	53.89	58.66	63.46	74.15	87.65
55	32.72	35.32	36.63	40.35	42.41	44.94	49.54	54.98	59.82	64.70	75.58	89.31
56	33.49	36.13	37.46	41.23	43.32	45.88	50.54	56.06	60.98	65.94	77.00	90.97
57	34.27	36.95	38.29	42.11	44.22	46.82	51.55	57.14	62.14	67.18	78.43	92.64
58	35.05	37.76	39.12	42.99	45.13	47.76	52.55	58.23	63.31	68.42	79.85	94.30
59	35.84	38.58	39.96	43.87	46.04	48.70	53.56	59.32	64.47	69.66	81.27	95.97
60	36.62	39.40	40.80	44.76	46.95	49.64	54.57	60.40	65.63	70.90	82.70	97.63
61	37.41	40.22	41.63	45.64	47.86	50.59	55.57	61.49	66.79	72.14	84.12	99.30
62	38.20	41.05	42.47	46.53	48.77	51.53	56.58	62.58	67.95	73.38	85.55	101.0
63	38.99	41.87	43.31	47.42	49.69	52.48	57.59	63.66	69.11	74.63	86.97	102.6
64	39.78	42.70	44.16	48.31	50.60	53.43	58.60	64.75	70.28	75.87	88.40	104.3
65	40.58	43.52	45.00	49.20	51.52	54.38	59.61	65.84	71.44	77.11	89.82	106.0
66	41.38	44.35	45.85	50.09	52.44	55.33	60.62	66.93	72.60	78.35	91.25	107.6
67	42.17	45.18	46.69	50.98	53.35	56.28	61.63	68.02	73.77	79.59	92.67	109.3
68	42.97	46.02	47.54	51.87	54.27	57.23	62.64	69.11	74.93	80.83	94.10	111.0
69	43.77	46.85	48.39	52.77	55.19	58.18	63.65	70.20	76.09	82.08	95.52	112.6
70	44.58	47.68	49.24	53.66	56.11	59.13	64.67	71.29	77.26	83.32	96.95	114.3
71	45.38	48.52	50.09	54.56	57.03	60.08	65.68	72.38	78.42	84.56	98.37	116.0
72	46.19	49.36	50.94	55.46	57.96	61.04	66.69	73.47	79.59	85.80	99.80	117.6
73	47.00	50.20	51.80	56.35	58.88	61.99	67.71	74.56	80.75	87.05	101.2	119.3
74	47.81	51.04	52.65	57.25	59.80	62.95	68.72	75.65	81.92	88.29	102.7	120.9
75	48.62	51.88	53.51	58.15	60.73	63.90	69.74	76.74	83.08	89.53	104.1	122.6
76	49.43	52.72	54.37	59.05	61.65	64.86	70.75	77.83	84.25	90.78	105.5	124.3
77	50.24	53.56	55.23	59.96	62.58	65.81	71.77	78.93	85.41	92.02	106.9	125.9
78	51.05	54.41	56.09	60.86	63.51	66.77	72.79	80.02	86.58	93.26	108.4	127.6
79	51.87	55.25	56.95	61.76	64.43	67.73	73.80	81.11	87.74	94.51	109.8	129.3
80	52.69	56.10	57.81	62.67	65.36	68.69	74.82	82.20	88.91	95.75	111.2	130.9
81	53.51	56.95	58.67	63.57	66.29	69.65	75.84	83.30	90.08	96.99	112.6	132.6
82	54.33	57.80	59.54	64.48	67.22	70.61	76.86	84.39	91.24	98.24	114.1	134.3
83	55.15	58.65	60.40	65.39	68.15	71.57	77.87	85.48	92.41	99.48	115.5	135.9
84	55.97	59.50	61.27	66.29	69.08	72.53	78.89	86.58	93.58	100.7	116.9	137.6
85	56.79	60.35	62.14	67.20	70.02	73.49	79.91	87.67	94.74	102.0	118.3	139.3
86	57.62	61.21	63.00	68.11	70.95	74.45	80.93	88.77	95.91	103.2	119.8	140.9
87	58.44	62.06	63.87	69.02	71.88	75.42	81.95	89.86	97.08	104.5	121.2	142.6
88	59.27	62.92	64.74	69.93	72.82	76.38	82.97	90.96	98.25	105.7	122.6	144.3
89	60.10	63.77	65.61	70.84	73.75	77.34	83.99	92.05	99.41	107.0	124.0	145.9
90	60.92	64.63	66.48	71.76	74.68	78.31	85.01	93.15	100.6	108.2	125.5	147.6

91	61.75	65.49	67.36	72.67	75.62	79.27	86.04	94.24	101.8	109.4	126.9	149.3
92	62.58	66.35	68.23	73.58	76.56	80.24	87.06	95.34	102.9	110.7	128.3	150.9
93	63.42	67.21	69.10	74.50	77.49	81.20	88.08	96.43	104.1	111.9	129.8	152.6
94	64.25	68.07	69.98	75.41	78.43	82.17	89.10	97.53	105.3	113.2	131.2	154.3
95	65.08	68.93	70.85	76.33	79.37	83.13	90.12	98.63	106.4	114.4	132.6	155.9
96	65.92	69.79	71.73	77.24	80.31	84.10	91.15	99.72	107.6	115.7	134.0	157.6
97	66.75	70.65	72.61	78.16	81.25	85.07	92.17	100.8	108.8	116.9	135.5	159.3
98	67.59	71.52	73.48	79.07	82.18	86.04	93.19	101.9	109.9	118.2	136.9	160.9
99	68.43	72.38	74.36	79.99	83.12	87.00	94.22	103.0	111.1	119.4	138.3	162.6
100	69.27	7~.25	75.24	80.91	84.06	87.97	95.24	104.1	112.3	120.6	139.7	164.3

N is the number of servers. The numerical column headings indicate blocking probability B in %. Table generated by Dan Dexter

Appendix C

Demonstrations: Mean Number of Elements in Network Systems

To define how a system works we will use Kendall notation written as A/B/C/K/N/D meaning:
- A: The arrival process (we use M to mark that it is a Poisson process for arrival process)
- B: Service time distribution (we use M for an exponential service time)
- C: Number of servers
- K: Number of place in the system
- N: Calling population
- D: Queue's discipline
If a factor does not appear it is because it is infinite.

C.1 M/M/1 system

We assume that the probability to be in the state k is

$$P_k = \frac{\lambda^k}{\mu^k} P_0 \tag{C.1}$$

Considering λ the arrival rate, μ the service rate and assuming that $\rho = \frac{\lambda}{\mu}$ the mean number of elements in the system N is:

$$\begin{aligned}
N &= \sum_{k=0}^{\infty} k p_k \\
&= \sum_{k=0}^{\infty} k(1-\rho)\rho^k \\
&= (1-\rho)\rho \sum_{k=0}^{\infty} k\rho^{k-1} \\
&= (1-\rho)\rho \sum_{k=0}^{\infty} \frac{\delta}{\delta\rho}(\rho^k) \\
&= (1-\rho)\rho \ \frac{\delta}{\delta\rho}[\sum_{k=0}^{\infty} \rho^k] \\
&= (1-\rho)\rho \ \frac{\delta}{\delta\rho}[\frac{1}{1-\rho}] \\
&= \frac{\rho}{1-\rho} \qquad \text{(C.2)}
\end{aligned}$$

The the mean elements in a M/M/1 system is $N = \frac{\rho}{1-\rho}$

C.2 M/M/∞ system

We assume that the probability to be in the state k is:

$$P_k = \frac{(\frac{\lambda}{\mu})^k}{k!} e^{\frac{-\lambda}{\mu}} \qquad \text{(C.3)}$$

Considering λ the arrival rate, μ the service rate and assuming that $\rho = \frac{\lambda}{\mu}$ the mean number of elements in the system N:

$$
\begin{aligned}
N &= \sum_{k=0}^{\infty} k p_k && \text{(C.4)}\\
&= \sum_{k=0}^{\infty} k \frac{\rho^k}{k!} e^{-\rho} && \text{(C.5)}\\
&= e^{-\rho} \sum_{k=0}^{\infty} k \frac{\rho^k}{k!} && \text{(C.6)}\\
&= e^{-\rho} (0 + \sum_{k=1}^{\infty} k \frac{\rho^k}{k!}) && \text{(C.7)}\\
&= \rho e^{-\rho} \sum_{k=1}^{\infty} \frac{\rho^{k-1}}{(k-1)!} && \text{(C.8)}\\
&= \rho e^{-\rho} \underbrace{\sum_{k=0}^{\infty} \frac{\rho^k}{k!}}_{e^{\rho}} && \text{(C.9)}\\
&= \rho e^{-\rho} e^{\rho} && \text{(C.10)}\\
&= \rho \underbrace{e^{-\rho+\rho}}_{1} && \text{(C.11)}\\
&= \rho && \text{(C.12)}
\end{aligned}
$$

In the equation C.8 we use the property that $exp(x) = \sum_{k=0}^{\infty} \frac{x^n}{n!}$

Bibliography

[1] World Internet Usage Statistics, Web-Site:
`http://www.internetworldstats.com/stats.htm`

[2] "Cisco Visual Networking Index: Forecast and Methodology, 2009-2014", Cisco VNI, June 2, 2010.

[3] K. G. Coffman, A. M. Odlyzko, "Internet growth: Is there a "Moore's Law" for data traffic?", AT&T Labs - Research, June 4, 2001.

[4] Jane M.Simmons, "Optical Network Design and Planning", Springer, 2008.

[5] C. Siva Ram Murthy and Mohan Gurusamy, "WDM Optical Networks, Concepts, Design and Algorithms", Prentice Hall, 2002.

[6] Fei Xue, S. J. Ben Yoo, "Performance Comparison of Optical Burst and Circuit Switched Networks", 2005, Optical Society of America.

[7] Xin Liu, "A Fair Packet-Level Performances Comparison of OBS and OCS".

[8] P-G. Fontolliet, "Teletrafic", October 2000.

[9] Yoo, S.J.B., "Wavelength conversion technologies for WDM network applications", in Journal of Lightwave Technology", 1998.

[10] E. Karasan, E. Ayanoglu, "Effects of wavelength Routing and Selection Algorithms on Wavelength Conversion Gain in WDM Optical Networks", IEEE Transactions on Networking, vol, 6, no 2, pp. 186-196, April 1998.

[11] M.N. Sysak, J.W. Raring, L. Johansonn, "Optical 2R and 3R Signal Regeneration in Combination with Dynamic Wavelength Switching Using a Monolithically Integrates, Widely Tunable Photocurrent Driven Wavelength Converter", from University of California Santa Barbara.

[12] S. J. Ben Yoo, "All-Optical Regeneration for Ultra-Long Fiber Links and its Prospects for Future Applications with New Modulation Formats", OSA/OFC/NFOEC 2009.

[13] Georgios I. Papadimitriou, Chrisoula Papazoglou, Andreas S. Pomportsis, "Optical Switching: Switch Fabrics, Techniques, and Architectures", in Journal of Lightwave Technology, vol. 21, n^o 2, February 2003.

[14] Todimala A., Ramamurthy B., "Algorithms for Intermediate Waveband Switching in Optical WDM Mesh Networks", High-Speed Networks Workshop, 2007.

[15] Xiaojun Cao, Vishal Anand, Chunming Qiao, "Waveband switching for dynamic traffic demands in multigranular optical networks", IEEE/ACM Transactions on Networking (TON), 2007.

[16] Richard S. Barr, M. Scott Kingsley, Raymond A. Patterson, "Grooming Telecommunications Networks: Optimization Models and Methods", technical report 05-emis-03, June 2005.

[17] K. Zhu, H. Zang, B. Mukherjee, "A Comprehensive Study on Next-Generation Optical Grooming Switches", IEEE Journal On Selected Areas in Communications, vol. 21, no. 7, pp. 1173-1186, September 2003.

[18] HEPnet Canada, Web-Site: `http://www.hepnetcanada.ca/lightpath`

[19] Martin Maier, "Optical Switching Networks", Cambridge, May 2008.

[20] M. Shiva Kumar and P. Sreenivasa Kumar, "Static lightpath establishment in WDM networks - New ILP formulations and heuristic algorithms", Computer Communications, vol. 25, Issue 1, pp. 109-114, January 2002.

[21] R. Ramaswami, K. N. Sivarajan, "Routing and Wavelength Assignment in All-Optical Networks", IEEE/ACM Transactions on Networking, vol. 3, no. 5, pp. 489-500, Oct. 1995.

[22] Hui Zang, Jason P. Jue, Laxman Sahasrabuddhe, Ramu Ramamurthy, Biswanath Mukherjee, "Dynamic Lightpath Establishment in Wavelength-Routed WDM Networks", IEEE Communications, September 2001.

[23] Yonghua Zhu, Rujian Lin, "Algorithms for lightpath establishment in wavelength-routed networks", Optical transmission, switching, and subsystems. Conference Wuhan, 2004, vol. 5281, pp. 334-341.

[24] Hui Zang, Jason P. Jue, Biswanath Mukherjee, "A review of Routing an Wavelength Assignment Approaches for Wavelength-Routed optical WDM Networks", in Optical Networks Magazine, January 2000.

[25] Sun X., Li Y., Lambadaris I., Zhao Y.Q., "Performance analysis of first-fit wavelength assignment algorithm in optical networks", Conference on Telecommunications, 2003. ConTEL 2003.

[26] Bsiwanath Mukherjee, "Optical WDM Networks", Springer, 2006.

[27] Josue Kuri Nicolas, Puech Maurice Gagnaire, Emmanuel Dotaro, "Routing And Wavelength Assignment Of Scheduled Lightpath Demands In A WDM Optical Transport Network".

[28] A. Jaekel, Y. Chen, "Resource provisioning for survivable WDM networks under a sliding scheduled traffic model", in Optical Switching and Networking, May 2008.

[29] Malabika S., Swapan K. M., Debashis S., "A protocol for piggy-backing on Markov based wavelength reservation in WDM optical networks", in Optical Switching and Networking, September 2009.

[30] Avishek Nag, Massimo Tornatore, "Optical Network Design with Mixed Line Rates", in Optical Switching and Networking, September 2009.

[31] Bo Wen and Krishna M. Sivalingam, "Routing, Wavelength and Time-Slot Assignment in Time Division Multiplexed Wavelength-Routed Optical WDM Networks", 2002 IEEE.

[32] Bose S.K., Singh Y.N., Raju A.B., Popat B., "Sparse converter placement in WDM networks and their dynamic operation using path-metric based algorithms", Communications, 2002.ICC 2002.

[33] Mahesh Sivakumar, Suresh Subramaniam, "Wavelength Conversion Placement and Wavelength Assignment in WDM Optical Networks", in High Performance Computing, pp. 351-360, 2001.

[34] S. Subramaniam, M. Azizoglu, A.K. Somani, "All-Optical Networks with Sparse Wavelength Conversion", IEEE Transactions on Networking, vol, 4, no 4, pp. 544-557, August 1996.

[35] Sébastien Rumley, Christian Gaumier, "Cost Aware Design of Translucent WDM Transport Networks", IEEE 2009.

[36] Emre Yetginer, Ezhan Karasan, "Regenerator Placement and Traffic Engineering with Restoration in GMLPS Networks", Photonic Networks Communications, 6:2, pp. 139-149, 2003.

[37] M. Flammini, A. Marchetti, G. Monaco, L. Moscardelli, S. Zaks, "On the Complexity of the Regenerator Placement Problem in Optical Networks", SPAA 2009.

[38] Si Chen, S. Raghanvan, "The Regenerator Location Problem".

[39] S. Pachnicke, T. Paschenda, P. M. Krummrich, "Physical Impairment Based Regenerator Placement and Routing in Translucent Optical Networks", OFC/NFOEC 2008.

[40] Wang Yao, Gokhan Sahin, Mengke Li, Byrav Ramamurthy, "Analysis of multi-hop traffic grooming in WDM mesh networks", in Optical Switching and Networking 6, 2009.

[41] Chunsheng Xin, Chunming Qiao, Sudhir Dixit, "Traffic Grooming in Mesh WDM Optical Networks-Performance Analysis", IEEE Journal on Selected Areas in Communications, vol. 22, no. 9, November 2004.

[42] Keyao Zhu, Hongyue Zhu, Biswanath Mukherjee, "Traffic Engineering in Multi-granularity Heterogeneous Optical WDM Mesh Networks through Dyanmic Traffic Grooming", IEEE 2003.

[43] Srinivasan Ramasubramanian, Arun K. Somani, "Analysis Of Optical Networks With Heterogeneous Grooming Architectures", Ieee/Acm Transactions On Networking, Vol. 12, No. 5, October 2004.

[44] Rajendran Parthiban, Rodney S. Tucker, "Waveband Grooming and IP Aggregation in Optical Networks", IEEE 2003.

[45] G. Maier, "Optical Network Survivability: Protection Techniques in the WDM Layer" in Photonic Network Communications, 4:3/4, pp. 251-269, 2002.

[46] Muntz, Gary S., "Multi-port line card redundancy technique for an intermediate network node", Cisco Technology, Inc., 2008.

[47] O. Crochat, J. Le Boudec, "Design Protetion for WDM Optical Networks", IEEE Journal on Selected Areas in Communications, Vol. 16, n° 7, pp. 1158-1165, September 1998.

[48] G. Maier, "Optical Network Survivability: Protection Techniques in the WDM Layer" in Photonic Network Communications, 4:3/4, pp. 251-269, 2002.

[49] O. Gerstel, G. Sasaki, "Quality of Protection (QoP): A Quantitative Unifying Paradigm to Protection Service Grades", Nortel Networks and the University of Hawaii.

[50] A. Fumagalli, M. Tacca, "Differentiated reliability (DiR) in wavelength division multiplexing rings", IEEE/ACM Transactions on Networking (TON), Volume 14 , Issue 1 (February 2006), pp. 159 - 168.

[51] J. Baliga, K. Hinton and R.S. Tucke, "Energy Consumption of the Internet",in *Proc. COIN/ACOFT*, Melbourne, Australia, 2007.

[52] Maruti Gupta, Suresh Singh, "Greening of the Internet", SIGCOMM '03, August.

[53] Gangxiang Shen, Rodney S. Tucker, "Energy-Minimized Design for IP Over WDM Networks", J. OPT. COMMUN. NETW./VOL. 1, NO. 1/ JUNE 2009.

[54] Francesco Palmieri Ugo Fiore and Sergio Ricciardi, "SimulNet: a wavelength-routed optical network simulation framework", 2009 IEEE.

[55] A. Nejat Ince, Arnold Bragg, "Recent Advances in Modelling and Simulation Tools for Communication Networks and Services", Springer, 2007.

[56] A Java library for a stochastic simulation API specification, documentation Web-Site:
http://www.iro.umontreal.ca/~simardr/ssj/doc/html/overview-summary.html

[57] Laith al Jazi (CIO), "Steps in a Simulation Study", 11 September 2007.

[58] Bo Wen, Nilesh M. Bhide, Ramakrishna K. Shenai, and Krishna M. Sivalingam, "Optical Wavelength Division Multiplexing (WDM) Network Simulator (OWns): Architecture and Performance Studies", SPIE Optical Networks Magazine Special Issue, March 2001.

[59] NS-2, Web-Site:
http://nsnam.isi.edu/nsnam/index.php/Main_Page

[60] Hegons project, Web-Site:
http://linux.softpedia.com/get/Science/Hegons-24746.shtml

[61] Oscar Pedrola, Miroslaw Klinkowski, Davide Careglio, Josep Solé-Pareta, Sébastien Rumley, Christian Gaumier, "JAVOBS: A Flexible Simulator for OBS Network Architectures", Journal of Networks, Vol 5, No 2 (2010), 256-264, Feb 2010.

[62] Martin Hasler, "Dynamical Networks", EPFL, March 2009.

[63] S. J. Aries, "Dictionary of Telecomunications", Butterworths, 1981.

[64] Eytan Modiano, Philip J. Lin, "Traffic Grooming in WDM Network", IEEE Communications Magazine, July 2001.

[65] JCreator Web-Site: http://www.jcreator.com/

[66] Suresh Subramaniam, Murat Azizoglu and Arun K. Somani, "All-Optical Networks with Sparse Wavelength Conversion", IEE/ACM Transactions on Networking, Vol. 4, NO. 4, August 1996.

[67] Ahmed Mokhtar, Murat Azizoglu, "Adaptive Wavelength Routing in All-Optical Networks", IEE/ACM Transactions on Networking, Vol. 6, NO. 2, April 1998.

[68] Ling Li, Arun K. Somani, "Dynamic Wavelength Routing Using Congestion and Neigborhood Information", IEE/ACM Transactions on Networking, Vol. 7, NO. 5, October 1999.

[69] Avishek Nag, Massimo Tornatore, "Optical network design with mixed line rates", Optical Switching and Networking 6, 2009.

[70] Zvi Rosberg and Diethelm Ostry, ”A Generic Time Driven Fractional Wavelength OCS”, ICTON 2009.