Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL FINAL DE GRAU

TÍTOL DEL TFG: Design and implementation of a graphical interface for the Orchestrator in SDN-enabled Data Centres

TITULACIÓ: Grau en Enginyeria de Sistemes de Telecomunicació

AUTOR: Jaume Soler Guerrero

DIRECTOR: Salvatore Spadaro i Albert Pagès

DATA: 19 de Juliol 2016

Título: Diseño e implementación de una interfaz gráfica para el Orquestrador de Data centers basados en el paradigma de SDN

Autor: Jaume Soler Guerrero

Director: Salvatore Spadaro y Albert Pagès

Fecha: 19 de Julio de 2016

Resumen

El aumento del uso de internet en estos últimos años debido a servicios en la nube, como redes sociales, aplicaciones, páginas web… ha obligado a la generación de centros de datos que nos permitan soportar esta gran cantidad de tráfico a diario. Un centro de datos actual para almacenar la gran cantidad de información que se quiere ofrecer, puede llegar a estar formado por cientos de miles de servidores y enlaces que permiten a los usuarios disfrutar de su contenido en cualquier momento, en casi cualquier lugar y con una buena calidad de servicio. Esta masificación de los centro de datos ha supuesto un gran reto para la comunidad científica en términos de cómo gestionar y controlar la gran cantidad de elementos que los forman. Por ello, se están desarrollando nuevas técnicas tanto de control como de gestión con el fin de disminuir los gastos de mantenimiento, facilitar el despliegue de nuevas aplicaciones, y en definitiva, sacarle el máximo partido a los centro de datos. Por este motivo, se ha querido desarrollar una interfaz gráfica útil y fácil de usar, capaz de facilitar el uso de las capas de gestión y control de un centro de datos el cual ofrece el servicio de virtualización de sus recursos.

En resumen, este proyecto desarrollado en Java, permite al usuario poder observar cómo están siendo usados los distintos recursos que forman su centro de datos de interés.

Title: Design and implementation of a graphical interface for the Orchestrator in SDN-enabled Data Centres

Author: Jaume Soler Guerrero

Advisor: Salvatore Spadaro and Albert Pagès

Date: 19th July 2016

Overview

The increased use of Internet in recent years due to cloud services, such as social networks, applications, web pages… has forced the necessity of data centres be able to support this large amount of daily traffic. A current data centre, in order to store this amount of information, can be made up of hundreds of thousands of servers and links that allow users to enjoy their content anytime, almost anywhere and with a good quality of service. Due to this mass of data centres, the scientific community has to face hard challenges of how to manage and control the large number of elements that form them. Therefore, new techniques are being developed both to control and manage with the purpose of reducing maintenance costs, facilitating the deployment of new applications, and definitively, making the most of the data centres. For this reason, it has been developed a useful graphical interface which is able to facilitate the use of management and control layers of a data centre.

To sum up, this project designed and developed in Java, allows the user to observe how the data centre's resources are being used in a visually and friendly way.

ACKNOWLEDGEMENTS:

Foremost, I would like to express my gratitude for my project advisors Salvatore Spadaro and Albert Pagès. Thanks for giving me the opportunity to make my project and for all the corrections and assistances necessary to go forward.

Besides my advisors, thanks to all the Grup de Comunicaciones Òptiques (GCO) from the UPC for their opinions and ideas in the development of the project.

And last but not least, I would like to thank the support received by my family, girlfriend and friends. Thanks for everything.

INDEX

1. INTRODUCTION

Data centres are expecting an exponential increase of the traffic that they have to sustain due to the cloud computing and a lot of emerging web applications in the last years. To support the billions of daily requests from users to the data centres, technological community has to confront the hard challenges of how to sustain the high traffic through the data centre network offering a good quality of service (QoS) and above all, how to control and manage their hundreds of thousands of servers, links and switches.

Nowadays, in order to confront these troubles, a three-layered architecture is being implemented on data centres. As it can see in Fig. 1, this architecture is formed by an orchestration layer able to give a whole network view of data centre with the purpose of efficiently manage data centre's resources. Also that, a control layer is used to offer an easily way to configure the different network elements of a data centre. And finally, the third layer is made up for the physical devices of data centre.

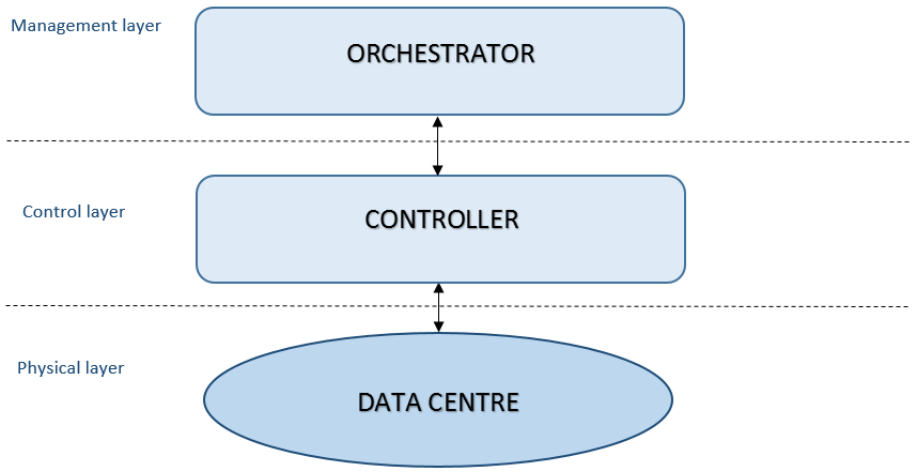


Fig. 1 Layered architecture in actual data centres.

In the same way, in recent years the virtualization of data centres' network is being developed as a promising service able to save costs and to reduce the complexity of the data centre's deployment in new applications and enterprises. In the following sections these terms will be explained in detail as well as how much can beneficiate to the data centre efficiency both in terms of use and management.

1.1. Objectives of the final project

The objective of this final project has been the development of a graphical tool which allows an easily work with the management plane of a data centre network based on SDN paradigm. The graphical interface will offer a friendly and intuitive tool in order to interact with the data centre's resources, providing a graphical view of how the servers, links, and switches are being used.

The design and development of the interface entails the following aspects:

- *Development of the Graphical User Interface:* the user's interface will offer the possibility to see and interact with the topology of data centre with the purpose to provision virtual data centres and to check the current status of the usage of the DC resources.

- *Establishment of the communication interfaces with the management plane:* these communications will be able to know how the orchestrator server is managing the data centre's resources.

- *Network Physical Parameters visualization:* the graphical interface will allow the possibility to show the different physical characteristics of each element on data centre and virtual data centre topology.

1.2. Scenario

Over the last few years, the exponential increase of the Internet traffic, mainly driven from emerging applications like video streaming, social networking and cloud computing has created the need of more powerful data centres, both in terms of computing resources and network capacities. Enterprise and government organizations are moving from test environments to placing their workloads into the cloud. On the other hand, consumers use cloud services to access to content and services, on multiple devices, in almost any place where they are located. Consequently, nowadays data centres' topologies are formed by hundreds of thousands of servers and links creating the necessity to use some methods to control and manage the big amount of network elements and their resources.

Firstly, in order to increase the data centre's (DC) efficiency by using its resources in the most optimal possible way, an orchestration layer responsible of the management of the DC is introduced. Moreover, regarding the configuration of the intra-DC network (DCN) resources, a Software Defined Networking (SDN)-based control infrastructure is considered. As it will be discussed in the following sections, the SDN-based controller facilitates the network control and configuration by decoupling the forwarding plane from the control plane.

Besides, with the purpose of minimize costs and get the greatest performance of data centres, the virtualization of them is a service progressively more and more used. In this project, in particular we focus on Virtual Data Centre (VDC), which is a service allowing to share the same physical infrastructure among multiple tenants. As it will be explained in more detail below, a VDC is a full-virtual network based on the creation of Virtual Machines (VM), inter-connected by virtual links, containing the necessary resources with the purpose of a tenant can install its applications which will be provided to third parties.

Fig. 2 shows the layered structure in which this scenario is based including the developed project. It can be understood as a dashboard of the orchestrator capable of show how data centre's resources are given to the multi VDC requests sent to the data centre.

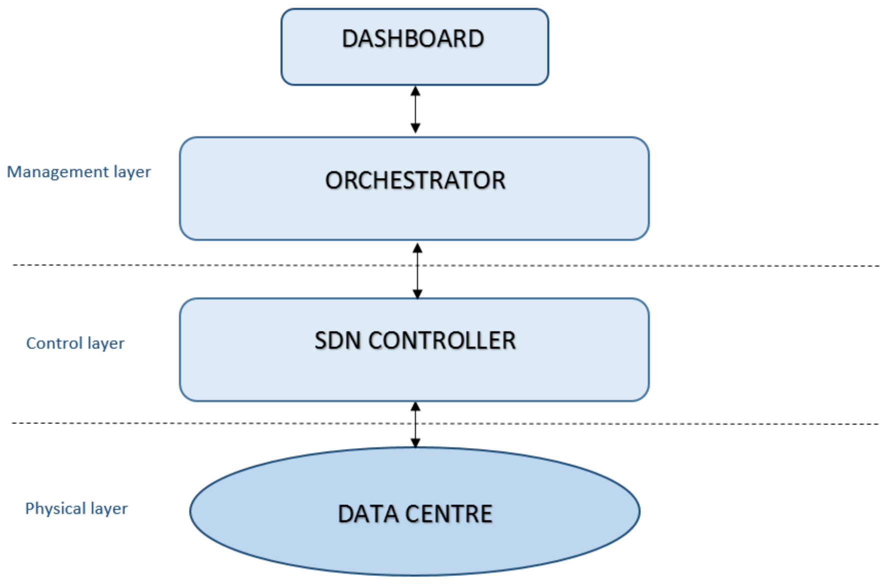


Fig. 2 This scenario layered structure of a data centre with the developed project.

In conclusion, by using this GUI, an administrator of a data centre which offers the virtualization of its resources, can observe how the data centre's resources are actually used or can trigger the provisioning of new VDCs.

1.2.1. Data Centres network topology

A Data Centre (DC) is a dedicated space where companies can keep and operate most of their ICT operations that support their business. Typically, a DC is a facility used to house computer systems and associated components, such as telecommunications and storage systems. Also that, sometimes there are backup power supplies, redundant data communication connections, environmental controls and security devices. With an estimated 100 billion web pages over 100 million websites and with almost two billion users accessing all these websites, including a growing amount of high bandwidth video, it's easy to understand but hard to comprehend how much data is being uploaded and downloaded every second on the Internet.

The data centre infrastructure is central to the IT architecture from which all content is sourced through. The stored data in servers' data centre is not static but it is in constant movement interrelated with each other resulting in new data. For this reason, a proper planning of the data centre infrastructure design is critical. The data centre network design is based on a layered approach that has been improved over the past few years. This layered structure seeks the improvement of performance, resiliency, flexibility and scalability in a data centre design.

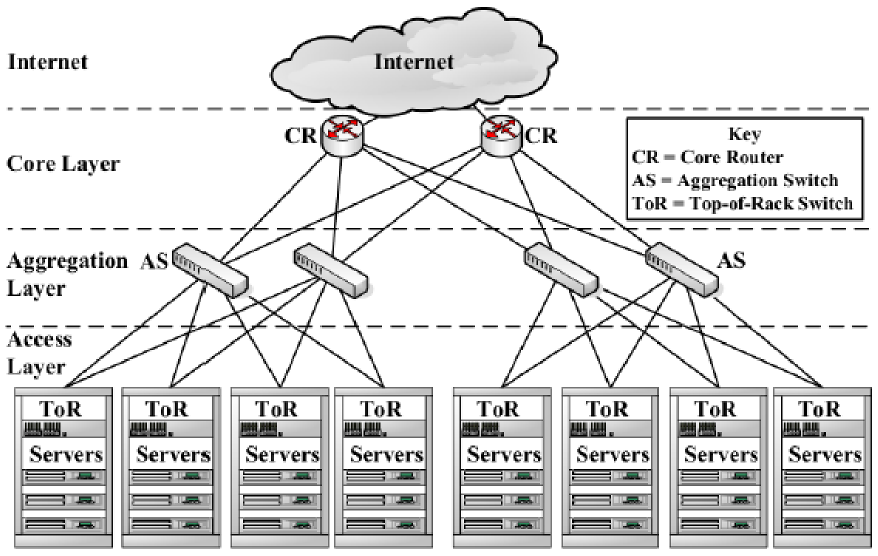


Fig. 3 Common DC Topology

As it can see in Fig. 3, [1], the layers of the data centre design are the: core, aggregation and access layers.

- *Core layer*: Provides the high-speed packet switching for flows in and off the data centre. This layer provides connectivity to multiple aggregation modules.

- *Aggregation layer*: Provides important functions, such as service module integration. These modules provide services, like content switching, firewall, intrusion detection, network analysis, and more.

- *Access layer*: Where the servers are physically attached to the network. Servers are stacked up in racks which are interconnected by ToRs (Top of Rack).

This is an example of a basic data centre topology but the high traffic of Internet has caused the huge complexity of them as it can see in the example of Fig. 4 [11].

Fig. 4 An overhead view of rows of servers inside the Google data centre in Council Bluffs.

1.2.2. Data Centres and Cloud Computing

The first data centres were designed following the classical architectures of the network informatics, in which the electronic devices were stacked up in racks. These DCs either were built to supply the current bandwidth and capacities requirements. Nowadays, the exponential increase of Internet as with Cloud Computing has produced the fast necessity of increment in size and in capacities the until now data centres known.

Cloud Computing is an informatics model in which the data and applications are divided in many data centres, each one containing hundreds of thousands of servers. That makes sense if we think about the big number of applications that every day have to support millions of requests, such as Hotmail, Gmail, Google, among many others. According to Cisco’s report [15] about the global cloud, Fig. 5, the amount of annual global data centre traffic in 2014 was estimated to be 3.5 Zettabytes, and by 2019, it will be almost triple, 10.4 ZB per year.

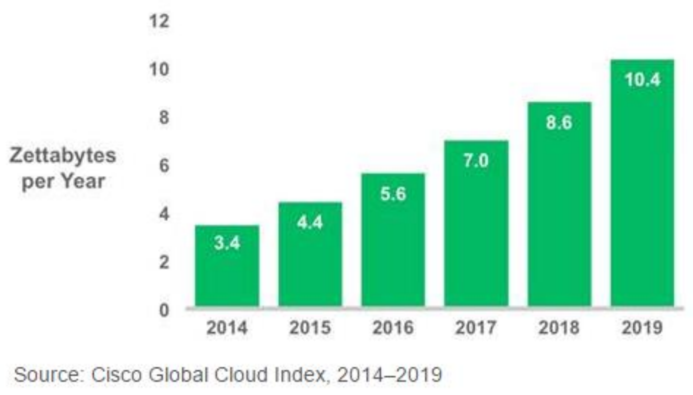


Fig. 5 Global Data Centre IP Traffic Growth

In the same way, in order to reply every request of users around the world, there are needed communications between data centres and above all communications between servers inside a data centre. Fig. 6 shows that the 75% of the global data centre traffic is inside the data centres, between its servers. In order to sustain this amount of traffic while offering a good quality of service, the servers of a data centre must experience low latency and high throughput with each other even if their number continues to increase. Due to this increment of servers (e.g. It is estimated that Google uses 900.000 servers approximately), the total power consumption inside the racks is increasing significantly. According to *Greenpeace's Make IT Green* report, the global demand for electricity from data centres was 330 billion kWh in 2007, and by 2020, it will be more than 1000 billion kWh.

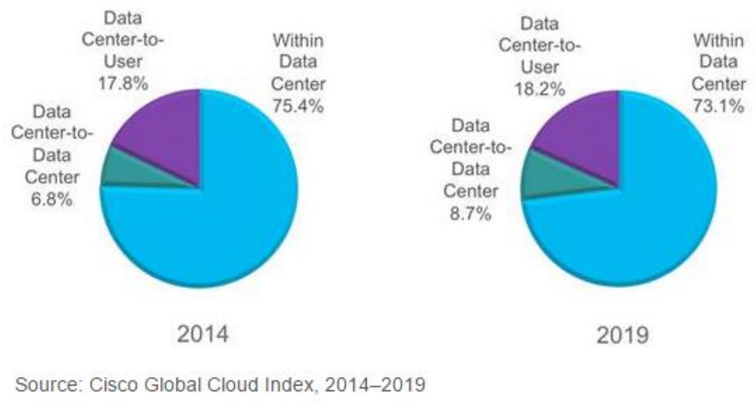


Fig. 6 Global Data Centre Traffic by Destination

To sum up, the impact of cloud computing on data centre traffic is clear and because of this, data centres will continue to dominate the Internet traffic and for this reason they have undergone an evolution.

1.2.3. From Electrical to Optical data centres

A data centre consists of thousands and thousands of racks hosting the servers (e.g. web, applications…) connected via the data centre interconnection network. When a request is sent by a user, a packet is forwarded through the Internet to the front end of the data centre. Then, the content switches are used to route the request to the appropriate server. A request may require the communication of this server with many others, for example, in a simple web search may be required the communication and synchronization between the web, the application and the database servers.

Because of this high demand of fast synchronization and capacities, current data centres based on electronic packet switches present a very important problem: it is difficult to guarantee a good quality of service because of their high latency and the high throughput demand. This is caused by, among others, high losses in large distances, problems with electrical conversion and with packets processing, and electromagnetic interferences. In order to confront the main problems of electric data centres, new interconnection schemes, such as optical networking, which can reduce power consumption, latency and increase the bandwidth demand have been developed in the last years.

1.2.4. Optical networking in data centres

Optical networking is a communication that uses signals encoded onto light to transmit information among various nodes of a telecommunication’s network. Component of an optical networking system can include: fibre, laser or LED light source, multiplexers, de-multiplexers, optical switches to direct light between ports without using an optical-electrical-optical conversion, optical splitters to send a signal down different fibre paths and optical amplifier. Nowadays, these networks based on optical fibre are being used more frequently thanks to their high throughput due to the use of WDM (see section 1.2.4.1.), low latency and low power consumption. While the physical limitations of electrical cable are speeds of 10 Gigabits per seconds, the limits of fibre optics have not reached yet.

Currently, the optical technology is used in data centres only for point-to-point links which are based on low cost multi-mode fibres (MMF). These links can offer throughputs of 10, 40 and 100 Gbps thanks to the using of WDM. In general, current data centres are based on commodity switches for the interconnection network. The network is usually a fat-tree 2-Tier or 3-Tier architecture. As it is represented in Fig. 7 [3], the servers are accommodated into racks and are connected through a ToR switch. The connections between racks and ToRs are by copper, so that convert the connection into an optical, ToRs are transceivers electrical-to-optical (E/O) and optical-to-electrical (O/E). For this reason, the main drawback of this architectures is the high consumption of the ToRs and because of the high number of required links. As well as, these ToR switches are interconnected through aggregate switches in a tree topology. In topologies Tier-3, such as shown in Fig. 7, it is used one more level in which aggregate switches are connected using core switches in order to get a better scalability.

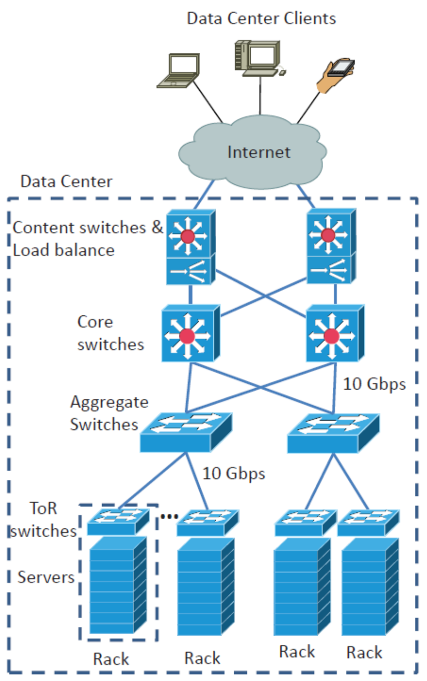


Fig. 7 Example of a current optical data centre

As shown before, these days optical data centres are increasingly used because of their advantages over the electrical ones. Data centres based on optical technology present a lower cost of materials, i.e. thousands of electrical links would be required to replace a single high bandwidth fibre cable. In the same way, optical connections bring a lower latency and a higher throughput, which are two essential terms in the current data centre demands.

1.2.4.1. *Wavelength Division Multiplexing (WDM)*

Wavelength Division Multiplexing (WDM) is a technology that puts data from different sources together on an optical fibre. This is possible by multiplexing a number of optical carrier signals onto a single optical fibre by using different wavelengths of laser-light. A WDM system uses a multiplexer at the transmitter to join the several signals together, and a de-multiplexer at the receiver to split them apart.

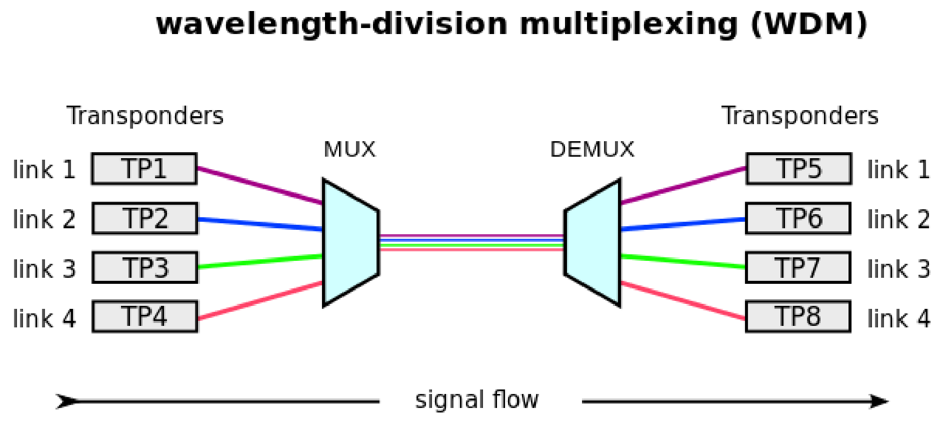


Fig. 8 WDM principle

In general, the transmitters employed in WDM applications require a control mechanism to respect the application's frequency stability requirements. Recommendation ITU-T G.691.1 [7] shows the frequency grid that supports a variety of fixed channel spacing between 12.5 GHz and 100 GHz.

1.2.5. SDN paradigm

Nowadays, Internet has led to the creation of a digital society where everything is connected and is accessible from anywhere. However, traditional IP networks are complex and very hard to manage. In the same way, current networks are also vertically integrated, in other words, control and data plane are bundled together. Today's network switches and routers program their forwarding tables locally, which means that network devices make their own decisions internally about how to forward traffic.

Software Defined Networking (SDN) is an emerging paradigm that changes this way of working, by breaking vertical integration, separating into two layers the control plane and data or forwarding plane (Fig. 9 [10]), giving the possibility to configure a network in the easier possible way, even more if it is compounded by thousands and thousands of servers and other network elements.

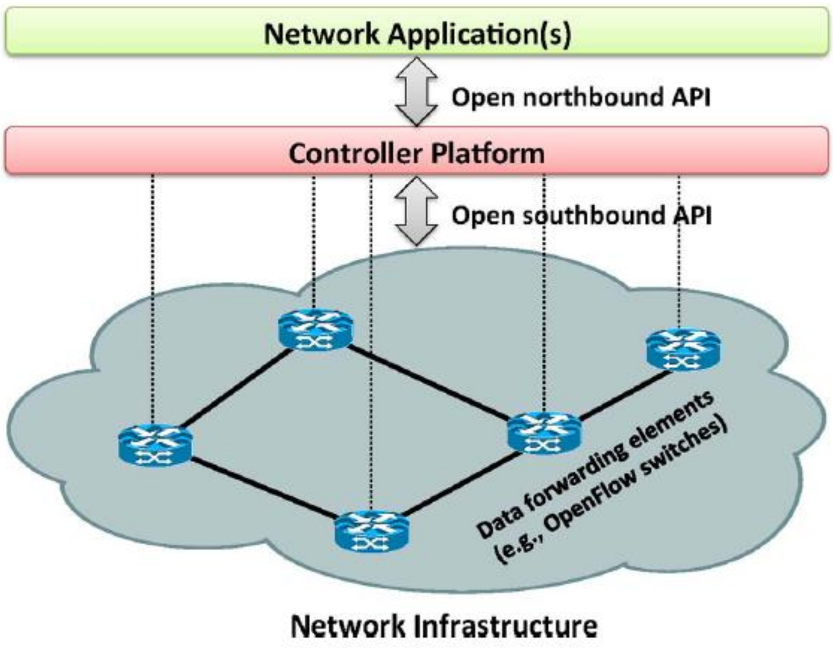


Fig. 9 Simplified view of an SDN architecture.

SDN aims to program the network with software running on a controller. There are two important terms in controller interfaces: northbound and southbound, for this reason, a controller can be thought as a middleware. The first one, the northbound communication, is formed by applications that tell the controller how to program the network. On the other hand, the southbound communication programs the network devices. In conclusion, the controller plays as an arbiter between the physical topologies and the applications that wish to program them.

1.2.5.1. Northbound and Southbound API

The terms northbound and southbound can apply to almost any type of network or computer system. A northbound API is an interface that allows the communication between any components with a higher-level component. Otherwise, southbound interface allows a particular component to communicate with a lower-level component. In spite of, these terms have been used increasingly because of APIs used in SDN networking.

In SDN, the southbound interface needs a protocol in order to configure different elements in the infrastructure layer, such as SNMP [34], OpenFlow [29], etc. In this context is used OpenFlow protocol which is the first standard communications interface defined between the controller and forwarding layers of an SDN architecture. As it is briefly explained previously, the main controller's function is to establish communication between the SDN controller and the network nodes in order to define network flows and implement requests relayed to it via northbound APIs.

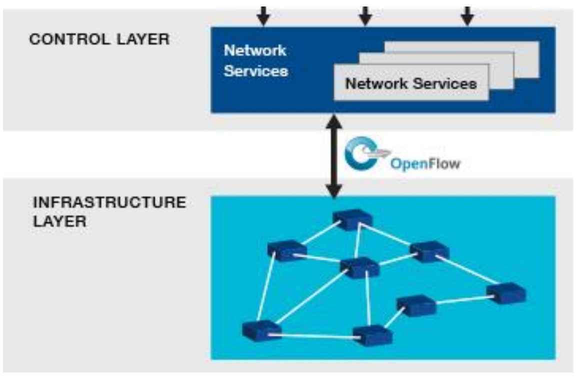


Fig. 10 Scheme of communication via OpenFlow [11]

Northbound in SDN protocol is the interface between software applications and the SDN controller. One of the most common API technology used at the northbound interface, as it will be explained in following sections, is the Representation State Transfer (REST) API. In data centre environment, northbound APIs can include management solutions for automation and orchestration. In the same way, southbound APIs include communication with network virtualization protocols, or the integration of a distributed computing network.

1.2.5.2. Rest API

One possible API technology used by northbound interface, and as it will be explained in following sections is based on a REST API.

Firstly, an API (Application Programming Interface) is an interface presented by a software able to collect information from or exchange to a set of resources. A REST (Representational State Transfer) is an architecture style for designing networked applications. Consequently, a REST API is a set of functions which developers can perform requests and receive responses. REST APIs use the HTTP/HTTPS protocol to execute these operations on resources represented by Uniform Resource Identifier (URI) strings. Among others, the main requests that API costumers are capable of sending are: GET, POST, PUT and DELETE.

- GET: read a specific resource by an identifier.
- POST: create a new resource.
- PUT: update a specific resource by an identifier.
- DELETE: remove a specific resource by an identifier.

1.2.6. Management and control in this scenario

As it has been explained in previous sections, this scenario is formed by data centre architectures based on the paradigm of SDN to configure their network elements and a management plane commanded by the Orchestrator, which allows to have an overview of the entire physical infrastructure in order to use their resources in the most optimal possible way.

In this scenario, the Orchestrator is based on *OpenStack* [12], which is a platform that offers multiple accessible services for managing the entire infrastructure and applications. On the other hand, the SDN controller is based on an open platform called *OpenDayLight* [13, 14] which is a Java open source controller infrastructure hosted by The Linux Foundation whose goal is provide a platform which allows the easily programming and facilitates the work between hardware and southbound protocols.

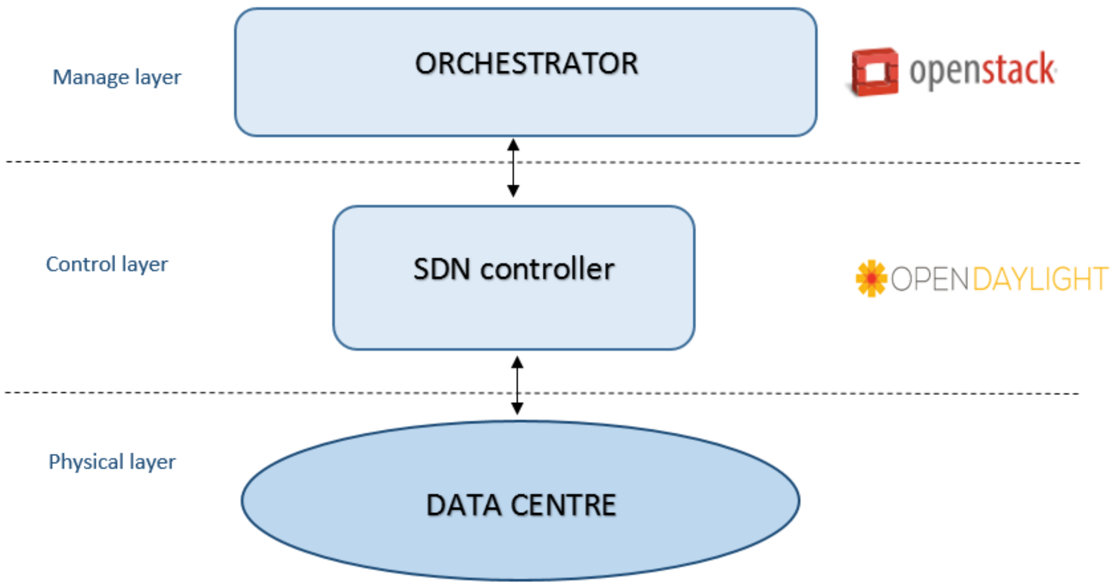


Fig. 11 General layer perspective with their platforms.

1.3. Objectives

In this project is requested the design and implementation of a graphical tool which allows the data centre administrator to observe and interact with the data centre resources. In particular, the provisioning of the VDCs service has been considered in this project. The tool has to communicate with the Orchestrator via REST API interface, with the objective to have a global and updated view of resources of an optical data centre. The tool is also a way to automate the provision of VDCs upon request from the end-users (e.g., service providers).

The main functionalities of the design and development of the tool are the following:

- *Load and draw topology of Data Centre:* load a text file with a matrix which describes the topology of an optical data centre.
- *Load and draw Virtual Data Centre's topology for many different tenants:* load a text file with a matrix which describes the topology of a VDC.
- *Load physical characteristics of topologies' elements:* load different text files in order to assign the resources to each element.
- *Network Physical Parameters visualization:* possibility to show the physical parameters of different network components.
- *Establishment of the communication with the* Orchestrator*:* when a VDC is loaded, the graphical interface will establish a communication with the management plane, responsible of the Virtual Date Centre Embedding (VDCE).
- *VDCE visualization:* show easily how the VDCs are mapped inside the data centre after the connection with the Orchestrator*.*

Other additional functionalities have been incorporated:

- *Visualization of Provisioning Time of each VDC:* fill a table with the time needed for the creation and mapping the VDC.
- *Daily LOG text file:* write in a text file the state of all the connections with the Orchestrator.
- *LOG visualization:* possibility to show the state of all the connections with the Orchestrator in one session.

1.4. Motivation

The development and design of a graphical interface allows to dispose of an auxiliary tool for data centre administrators, making easier the provisioning of services (VDCs) and at the same time it represent a valid tool to check the status of current usage of the data centres resources. Additionally, the development of the software modules to implement the GUI has meant a continuous learning process of both programming language and technologies. These are a very important abilities in the actual technology societ

2. VIRTUAL DATA CENTRES (VDC)

The increase of data centre's size and the high number of elements inside them, have caused a very important increment of power consumption, expense, etc. In order to solve these issues, technology's society has instituted a term of 60's called virtualization on the current data centres. Virtualization of data centres is an emerging concept which expects apply the concept of network virtualising in the data centres. In other words, create virtual instances of data centres' resources, also known as Virtual Data Centres (VDC). A virtualized data centre provides computational and network resources allowing tenants to apply their own policies, define addresses' spaces, manage their VMs independently, etc.

A Virtual Data Centre can be described as an Infrastructure as a Service (IaaS) that extends virtualization concepts to all the data centres resources and services. VDCs allow to data centres operators the possibility to give part of its infrastructure up to multiple tenants logically separated between them. These tenants VDC are a virtual infrastructure formed by computing resources (Virtual Machines, VMs) interconnected with virtual links with a certain capacity.

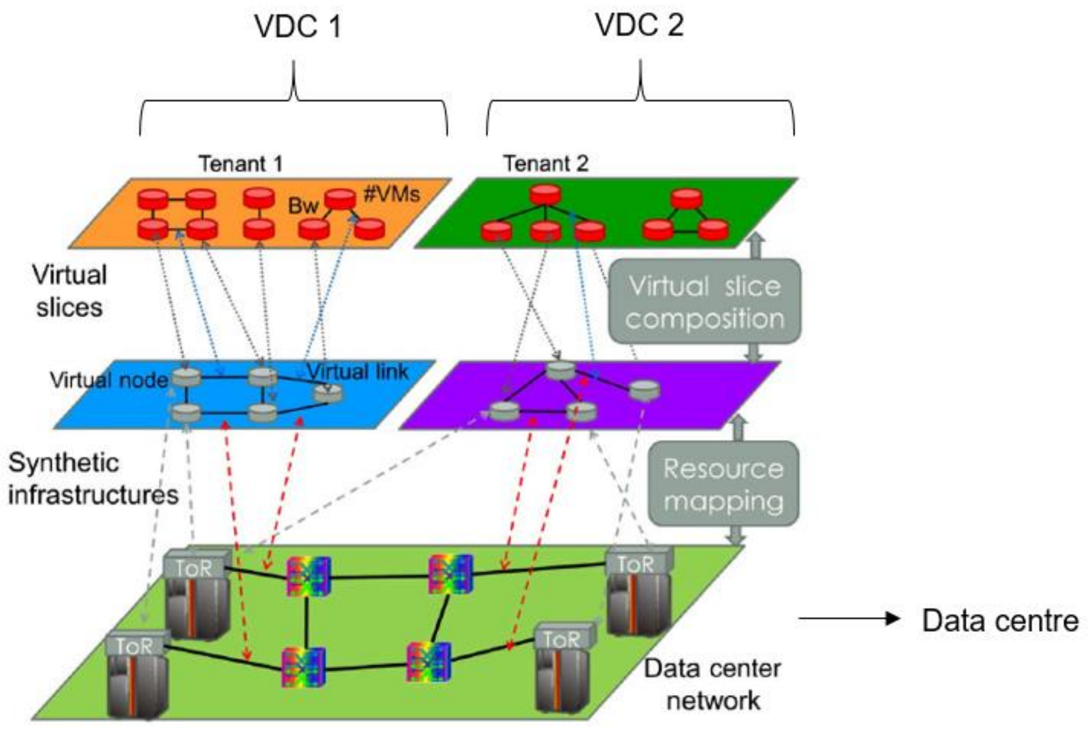


Fig. 12 Multi VDC scenario

By a VDC request, a tenant reserves a portion of a DC (e.g. a petition can require a determined number of virtual machines specifying the bandwidth of the connections between them and their operating system). In Fig. 12 [6] it is shown an example of two tenant VDC build in a data centre. Tenant 1 and tenant 2 can share the same rack, even the same server, without interfering with each other.

However, the implementation of VDC in a data centre brings the challenge of how optimally allocate the resource demands of many virtual data centres on a physical infrastructure, so that the data centre operating costs are reduced, improving its revenue and fulfilling the agreements of quality service. This challenge is known as Virtual Data Centre Embedding (VDCE) problem.

2.1.1. Virtual Data Centre Embedding (VDCE)

The issue of assignment the resources demanded by a VDC into a physical data centre in the most optimal way possible is called VDCE problem. Mainly, VDCE consists in mapping the VMs of a VDC onto the physical resources of a data centre, which are limited, with the objective of minimizing the number of blocked VDCs.

The aims of VDCE are:

- Maximize number of VDCs assigned into a data centre
- Reduce costs of assignment
- Reduce costs in intra data centre communication
- Reduce power consumption in data centre, embedding resources in the way against VDC failures

The responsible to sort it out and map many VDCs into a data centre is the Orchestrator located in the management plane. It is the responsible to process the data centre topology and the request of the VDC. Also that, it contains the algorithms which will decide where instance the different VMs of the VDC inside the data centre. Then, a service of Orchestrator, named Nova, will establish them in the correct server.

From a point of view of resources assignment, the capacity of data centres components are fragmented with the purpose of obtaining the individual instances that will conform the VDCs. In general, the elements that are to take into account are the following:

- Servers: servers are fragmented in instances called virtual machines whose capacities used in the mapping procedure are: Central Processing Unit (CPU), hard disk capacity and Random-Access Memory (RAM).
- Switches and routers: in these cases there are taken into account the number of available virtual ports and buffer capacity of each one. Also that, could be important CPU and RAM.
- Links: for links the most important thing is the bandwidth, but sometimes the delay and the jitter are also used.

2.1.1.1. Algorithm's complexity

As it is said previously, algorithms are in charge of the optimally assignments of VDC's resources into a data centre. Keeping this in mind, there are three approaches to confront this need using three different types of algorithms:

- Exact algorithms: this type of algorithms are implemented by mathematical techniques. They supply the optimum problem solution but can become inefficient depending on the instance's size. Therefore, this kind of algorithms is recommended for little instances of VDCE.
- Heuristic algorithms: finding the most optimal solution of a VDC mapping into a data centre with hundreds of thousands servers can take a lot of time. For this reason, there are needed some algorithms whose solution is based on a balance between the optimal solution and the execution time to assign the resource. That's why heuristic algorithms can obtain the solution in a shortest execution time but not the most optimal.
- Meta-heuristic algorithms: they are iterative algorithms that are improving their solution in each iteration. It is chosen the best solution between the proposed when some parameter is reached.

To conclude, thanks to the virtualization in data centre so many VDCs can be provisioned in just one physical topology. This supposes a significant saving costs and an easier way to new applications deploy, because of the non-necessity to have a data centre for each company that wants to offer a service, but just knowing the amount of required sources to offer it.

3. IMPLEMENTATION OF THE INTERFACE

In order to make this project, it was needed a language to create a Graphic Interface, which allowed us to implement some of the required functions. It was decided that Java would be the best option because of its simplicity; moreover, using Java allows us to create a software from scratch and no an extension of another platform of manage plane, in this case *OpenStack*.

3.1. Java

Java is a programming language introduced by Sun Microsystems in 1995, designed to use in the distributed environment of the Internet. Some of the major Web browsers include a Java virtual machine. It was designed to have the "look and feel" of the C++ and enforces an object-oriented programming model. There are some characteristics that make Java a very potential manner of programming:

- Java's applications are compiled into what Java calls bytecode, which can be run on a server or client so these can be used on a single computer or in different clients in a network.

- Programs written in Java cannot contain references to data external to themselves of other known objects, for this reason the Java's code is robust because an instruction addressed to data storage of another application or operating system would cause the program or the operating system "crash". Instead of this, Java Platform is a set of dynamically loadable libraries that can call at run time.

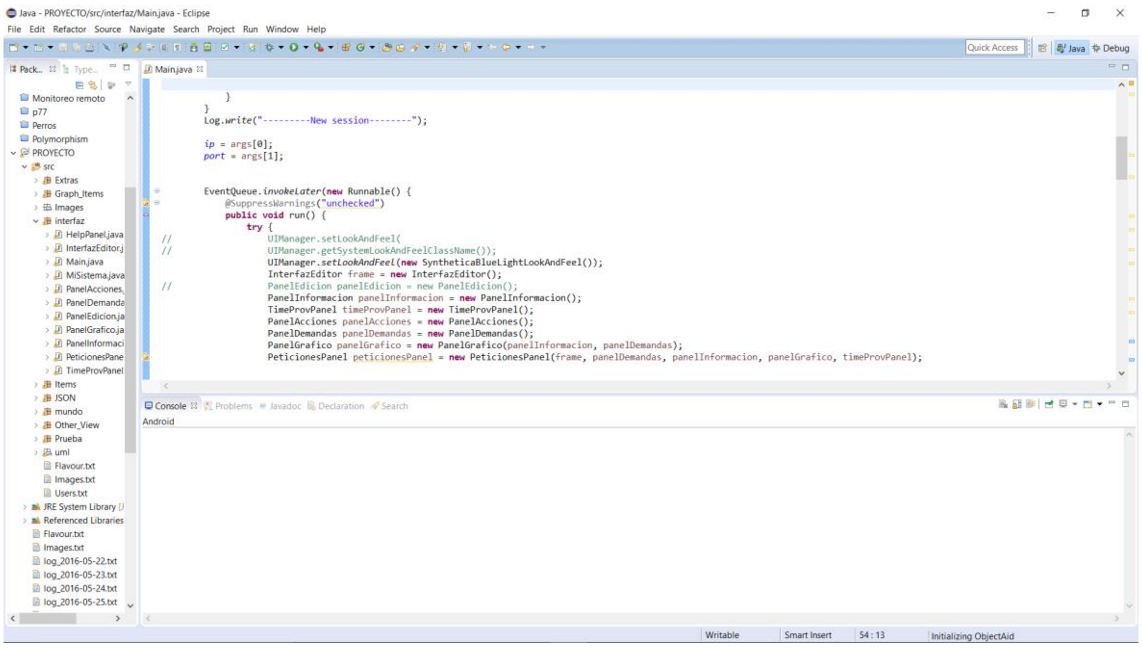
- Java is object-oriented, which means that it is based on the concept of "objects" which are some data structure in a location in memory, which have a "state", "method" and "identity".

- Simplicity to learn it.

Having decided a language, it is necessary a tool to work with Java easily. These tools are called Java IDEs. An IDE (Integrated Development Environment) is a software application which enables users to write and debug Java programs. There are a lot of Java IDEs but in this project is used Eclipse, version: Mars.1 Release (4.5.1).

3.2. Eclipse

Eclipse is a Free and Open Source IDE platform based on Java which contains a base workspace and an extensible plug-in system for customizing the environment. This tool started in 2001 when IBM donated three million lines of code from its Java tools. Originally, the goal of Eclipse was to complement the community that surrounds Apache.



```
            }
        }
        Log.write("---------New session--------");

        ip = args[0];
        port = args[1];


        EventQueue.invokeLater(new Runnable() {
            @SuppressWarnings("unchecked")
            public void run() {
                try {
//                  UIManager.setLookAndFeel(
//                  UIManager.getSystemLookAndFeelClassName());
                    UIManager.setLookAndFeel(new SyntheticaBlueLightLookAndFeel());
                    InterfazEditor frame = new InterfazEditor();
//                  PanelEdicion panelEdicion = new PanelEdicion();
                    PanelInformacion panelInformacion = new PanelInformacion();
                    TimeProvPanel timeProvPanel = new TimeProvPanel();
                    PanelAcciones panelAcciones = new PanelAcciones();
                    PanelDemandas panelDemandas = new PanelDemandas();
                    PanelGrafico panelGrafico = new PanelGrafico(panelInformacion, panelDemandas);
                    PeticionesPanel peticionesPanel = new PeticionesPanel(frame, panelDemandas, panelInformacion, panelGrafico, timeProvPanel);
```

Fig. 13 Screenshot of Java Eclipse

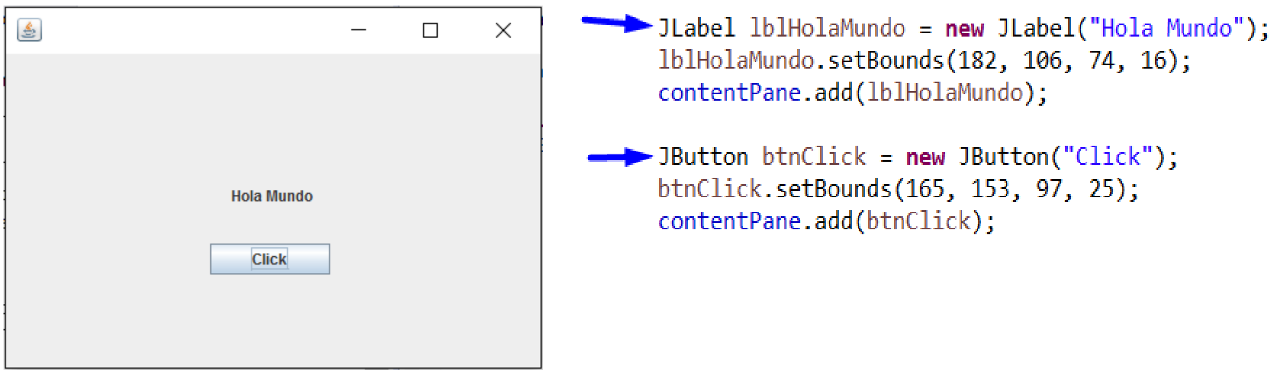
As it is explained previously, Java uses libraries which contain the functions that allow the programmer a well-known set of useful facilities, such as container classes and regular expression processing. Also that, the library provides an abstract interface to tasks that normally depend on the hardware and operating system.

In this project, apart from the standard Java libraries, there have been required other libraries and package to develop every required functionality, such as: Javax.Swing, GSON and Java.net.

3.3. Libraries and package

3.3.1. Javax.Swing package

Java provides a lot of options to create graphical interfaces. In this project, is used the javax.Swing package which contains the most important classes and interfaces of Java Swing (JSwing). JSwing API is based on Model-View-Controller and is set of extensible GUI Components build upon top of AWT API and acts as replacement of it as it has almost every control corresponding to AWT controls. AWT (Abstract Window Toolkit) is the original user-interface widget toolkit preceding Swing.



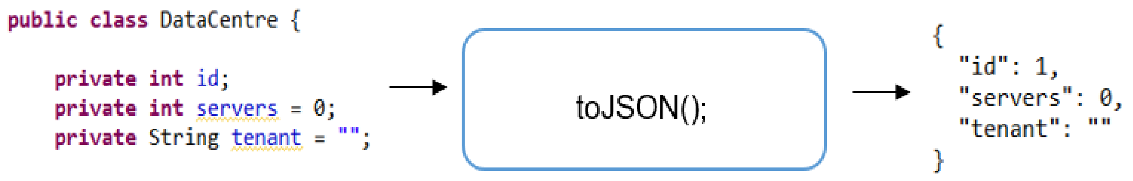
```
JLabel lblHolaMundo = new JLabel("Hola Mundo");
lblHolaMundo.setBounds(182, 106, 74, 16);
contentPane.add(lblHolaMundo);

JButton btnClick = new JButton("Click");
btnClick.setBounds(165, 153, 97, 25);
contentPane.add(btnClick);
```

Fig. 14 Example of a simple GUI using JSwing

3.3.2. GSON

As it is explained in previous sections, the GUI communicates with the Orchestrator by a REST API interface. Moreover, in order to create or take some resource to/from the Orchestrator, it is needed to send different files in JSON format. JSON (JavaScript Object Notation) is a text format based on subset of the JavaScript Programming Language. It is familiar with C-programmers but completely language independent, for this reason JSON is an ideal data-interchange language.



```
public class DataCentre {

    private int id;
    private int servers = 0;
    private String tenant = "";
```

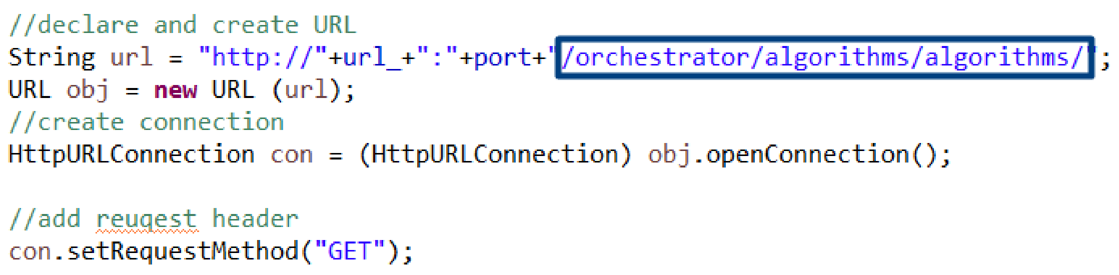
```
{
  "id": 1,
  "servers": 0,
  "tenant": ""
}
```

Fig. 15 Simple example of toJSON( ) method of GSON library

In order to make the conversion from Java class to JSON or in the other way round (see Fig. 15), amongst other functionalities, Google offers a library called GSON which is an open source Java library and contains so many different methods that makes easier the possibility to work with JSON files.

3.3.3. Java.net Package

Java.net package provides the classes for implementing the network applications required to establish the communication between the GUI and the Orchestrator. This package can be divided into two sections, *Low Level API* and *High Level API.* The first one deals with “Addresses”, “Sockets” and “Interfaces” abstractions, and HL API is responsible of “URIs”, “URLs” and “Connections”.



```
//declare and create URL
String url = "http://"+url_+":"+port+"/orchestrator/algorithms/algorithms/";
URL obj = new URL (url);
//create connection
HttpURLConnection con = (HttpURLConnection) obj.openConnection();

//add reuqest header
con.setRequestMethod("GET");
```

Fig. 16 Code of a GET request made with Java.net functionalities

Fig. 16 shows the implementation of a GET request using functionalities of Java.net package. First of all, the URL with URI is declared and created. The URI (blue square) indicates to which module of the Orchestrator the request is sent. In this case, the GET request is sent to algorithms module in order to take the list of available algorithms. Then, the connection is instanced using the previous URL declared and finally, the request “GET” header is added.

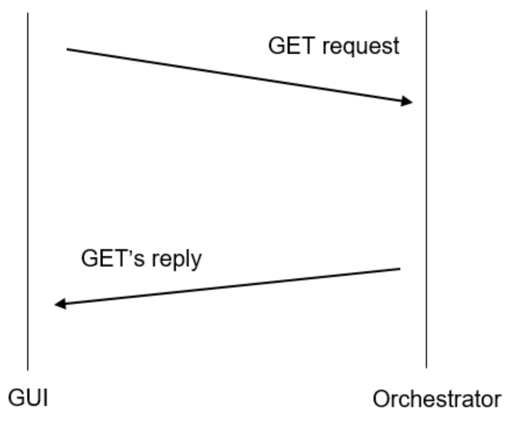


Fig. 17 Scheme of messages in a GET petition

In Fig. 17 it can see the participating messages in a GET request. As it is explained in GSON library section, the data of requests and replies is in JSON format. In this scenario, the GET reply would be a JSON text with all the available algorithms and their parameters, then, GSON methods will convers this JSON text into the different Java class required.

4. VALIDATION OF THE GUI

In this section, the implemented functionalities of the graphical interface are tested in two parts. The first one will show the well-work of interface’s functions. For example, a data centre topology will be loaded with its elements and the physical information of each one. Also that, buttons of load VDC’s topology will be tested and the panel information which will represent the physical information of each topologies’ element. Then in the second part, some VDCs tenant petition will be loaded and via a REST API, they will be sent to the orchestration server. It will process and instance the VDC topology inside the data centre.

In this context is considered that all racks host all types of computing resources while a single VM is always mapped onto a single rack. However, VMs belonging to the same VDC are mapped in different racks in order to provide some degree of protection against rack failures. Even so, VMs of different VDCs can be mapped in the same rack, even sharing a server. In the same way, virtual links have to be assigned in many physical optical links respecting the wavelength continuity constraint, so different VDCs must not share optical resources between them for isolated purposes.

Also that, the SDN controller will be the responsible to configure the network elements of the DC topology in order to establish the communication between the VMs of the VDC. After that, it will able to see graphically where have been mapped every VM of each VDC and how the data centre’s servers resources and links’ wavelength have been updated.

The main functionalities that will be tested are:

- Load topologies of DC and VDCs
- Visualize network elements information
- Communication with the Orchestrator
- Show mapping of VDCs
- Update busy or free DC’s resources

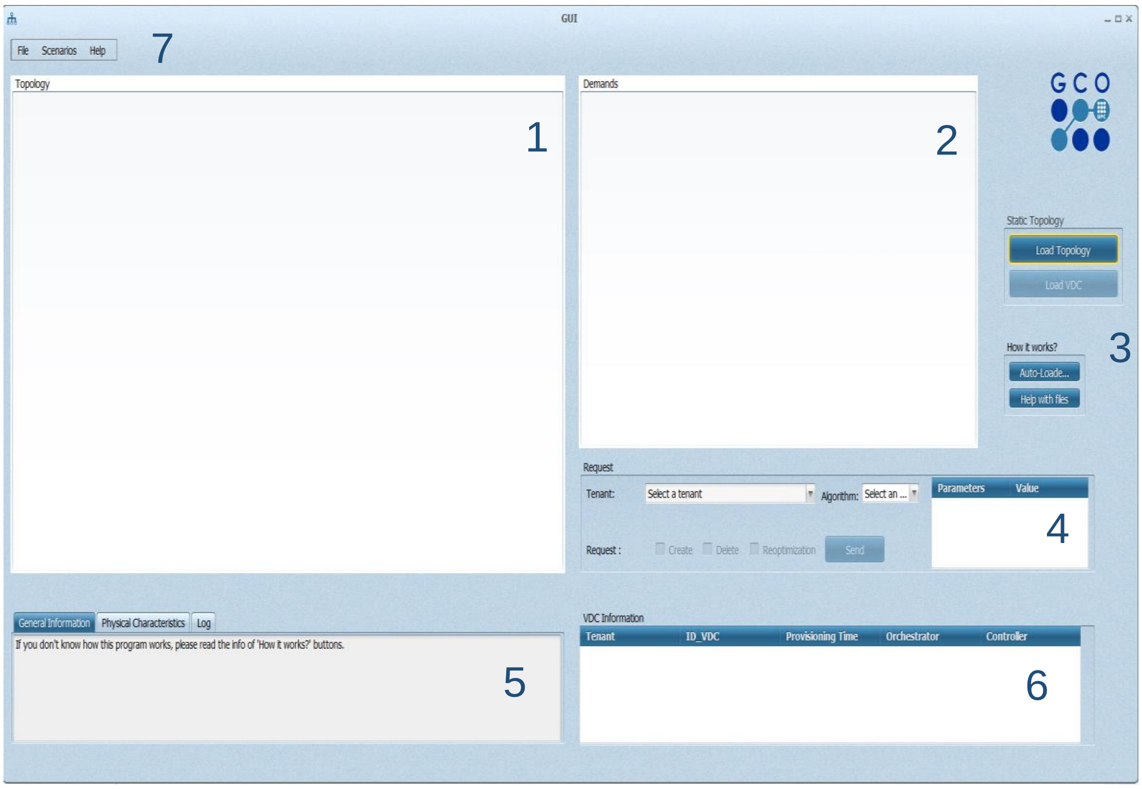
4.1. Test Network Characteristics

As has been discussing during the document, this interface is focused on a data centre network. In the following tests, the DC network in which the different VDC will be mapped, has the characteristics that are shown in the following table:

Parameters	Value
Number of ToRs:	8
Number of Optical Switch:	1
Number of Links:	8
Link's Channel Spacing:	100 GHz
Number of Racks:	8
Number rack's servers:	32

4.2. A global description of the interface

Before starting with the simulation of the project, in Fig. 18 it is shown a general perspective of the graphical interface:



If you don't know how this program works, please read the info of 'How it works?' buttons.

Fig. 18 Global perspective of the graphical interface

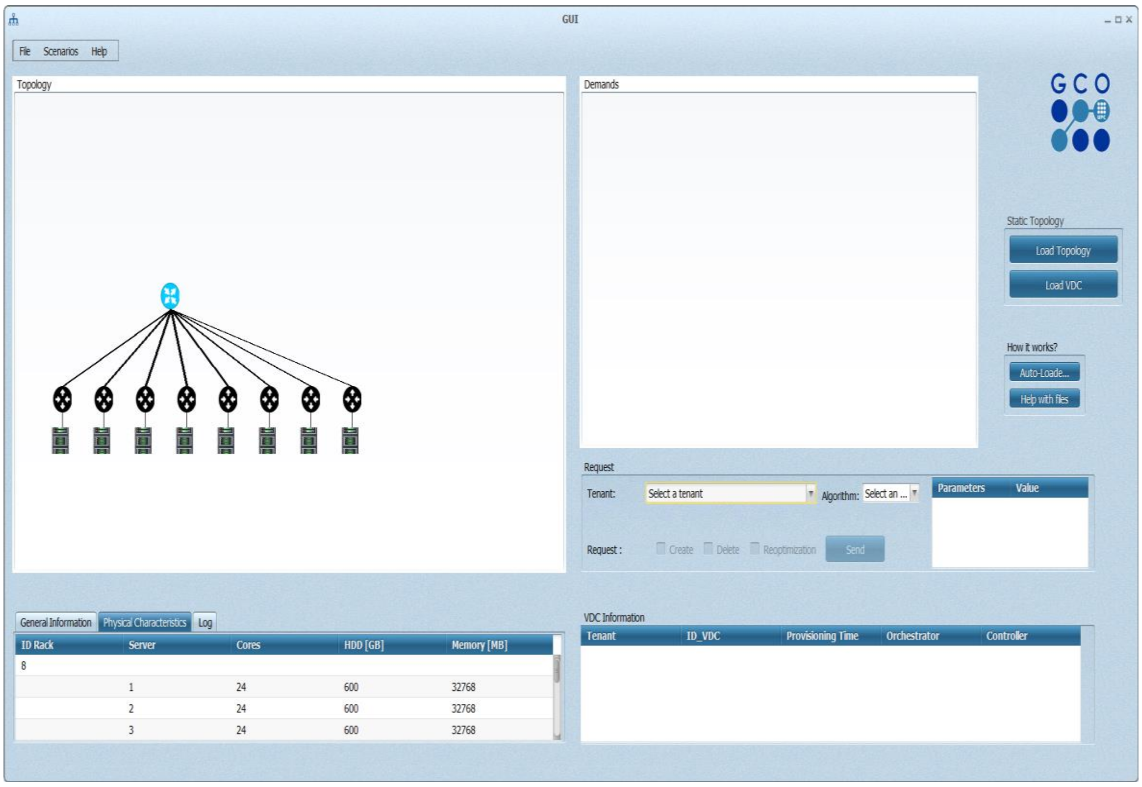
As it can see in the image, the interface can be divided in 7 interest spaces. 1 and 2 are the main panels where the DC (1) and VDCs topologies (2) are displayed. Spaces 3 and 7 contain the buttons and the toolbar in charge of load the files of topologies. For this reason, they also include some help buttons which explain how to make them and the structure of other files needed for the well-working of the application. Also that, (7) gives the possibility of using the interface for other scenarios than intra-data centres, as it will be explained in section 4.

In the middle of the screen under the VDC panel, there is the request panel (4). This panel allows the user to select the tenant so that after creating its VDC, select the algorithm which with the mapping will be established and shows a table which will be filled with the algorithm's parameters and its values. Finally, in the lower space of the interface, there are the panels 5 and 6. The latter, (6), contains a table where will be represented the values of *Provisioning Time* (see section 1.2.) of each VDC and panel 5 is an informative panel which will show the physical information of topologies' elements, the state of the communications with the server, among others informative messages.

4.3. Testing the graphical interface

Firstly, when the application is executed three text files obtained by the Orchestrator and located in the program source path are auto-loaded. The first file, load the list of tenants which have made a request in order to instance their VDC in our data centre. The second one is the flavour file. This file contains the different models of VM with its physical characteristics (Core, HDD and RAM) and identified by an ID. Tenant will be able to choose which VMs use respecting the requirements needed to offer its service. Then, the third auto-loaded file shows the different possible operating system of the VMs identified by an ID. With these files loaded, the application gets from the Orchestrator the list of the available algorithms with their parameters with the purpose of the data centre administrator will be able to select one of them in the VDCE.

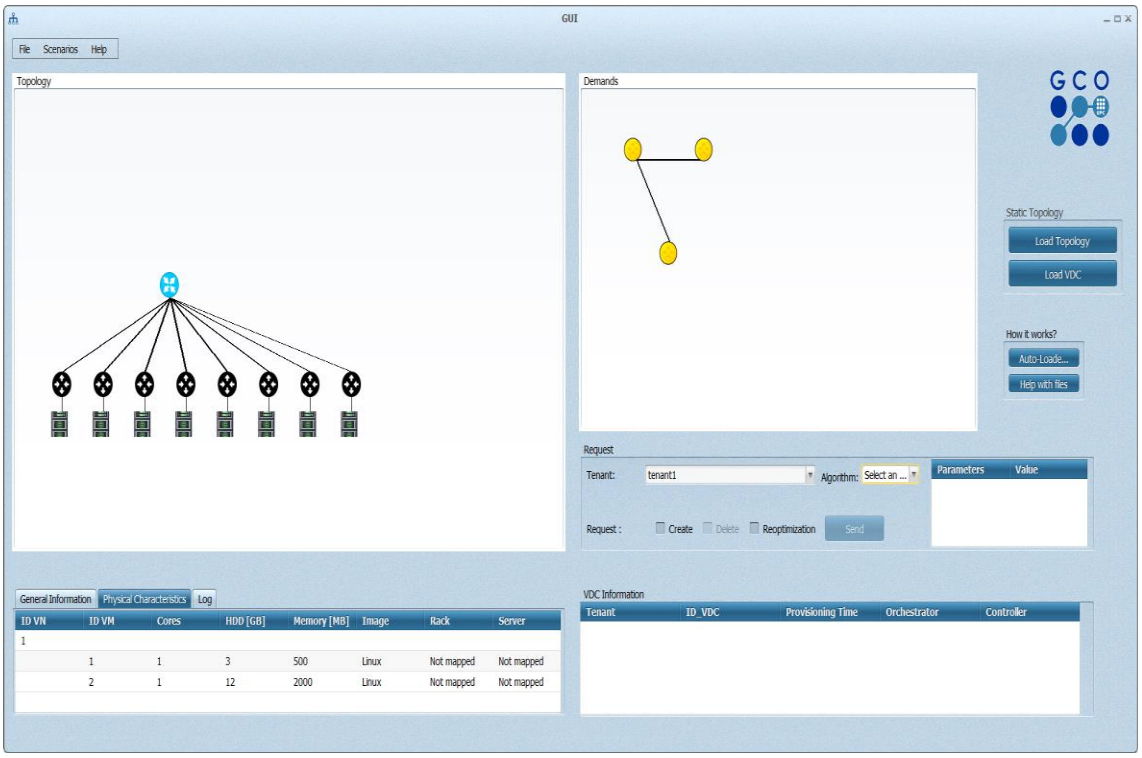
Subsequently, the user can load the data centre topology in which the Orchestrator will route the different elements of the VDC. This topology is represented by a matrix in a text file and contains its number of racks, ToRs, optical switches, links and how they are inter-connected. Also that, this file contains the physical information of all the data centre's servers' resources; number of cores, memory in disk and RAM as well as the channel spacing of the links. Indeed, all the information of each network element is available on the information tags, such as state of servers' resources, state of links' wavelength, number of ToR's ports, among others. Fig. 19 shows the data centre topology loaded and the information of a rack's servers.



ID Rack	Server	Cores	HDD [GB]	Memory [MB]
8				
	1	24	600	32768
	2	24	600	32768
	3	24	600	32768

Fig. 19 Data centre topology loaded and rack information shown.

Once the DC topology has been loaded, the user can select a tenant and load the file of its VDC topology. This file is formed with a topology matrix, consequently, with the number of Virtual Nodes (VN) and each one with its number of VMs. In the same way, it contains the ID of flavour and image of each VM which the tenant has chosen previously in our VMs catalogue. Also with DC elements, the information of each VN can be shown on the panel information but in this case apart from physical characteristics there is represented if the VMs' VN are mapped and in case of be, in which rack and server. Fig. 20 shows the DC and VDC (id: 1) of tenant 1 loaded. Also that, on the information tags it is shown the physical and mapping characteristics of VMs of VN: 1.

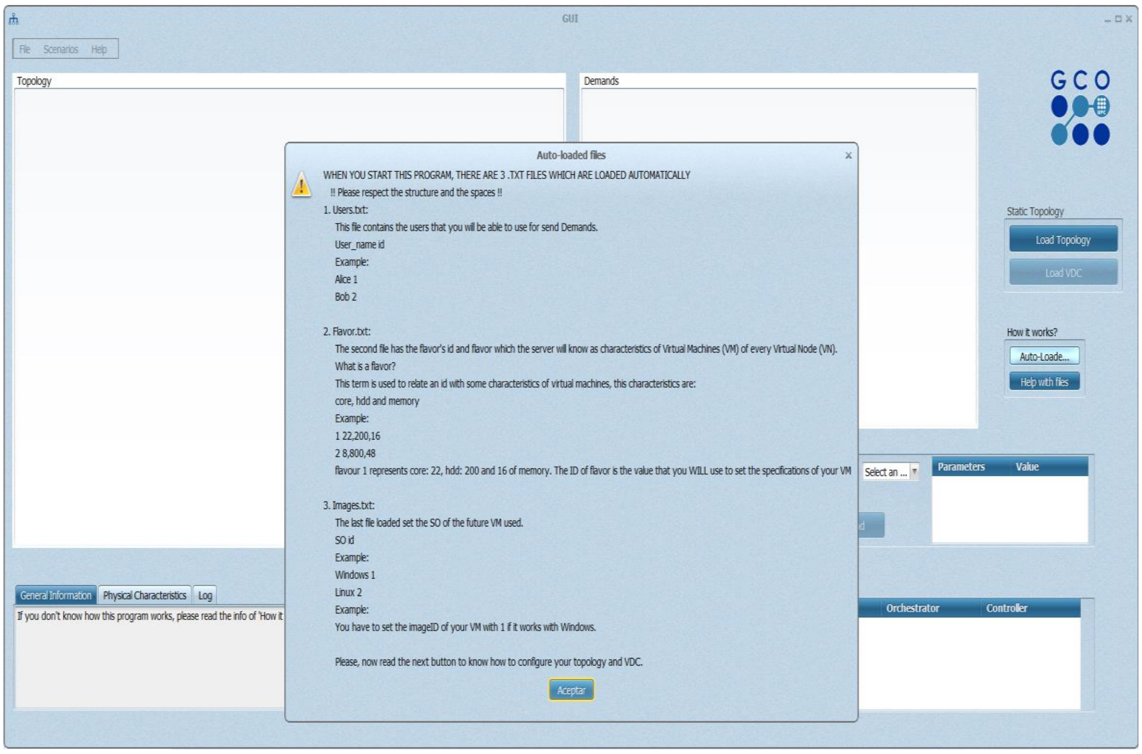


ID VN	ID VM	Cores	HDD [GB]	Memory [MB]	Image	Rack	Server
1							
	1	1	3	500	Linux	Not mapped	Not mapped
	2	1	12	2000	Linux	Not mapped	Not mapped

Tenant	ID_VDC	Provisioning Time	Orchestrator	Controller

Fig. 20 DC and VDC loaded. On information tags, physical characteristics and mapping information.

In order to the user can know the structure topology and the physical characteristics files, some HELP buttons are implemented. An example of HELP button of the auto-loaded files is shown in Fig. 21.



If you don't know how this program works, please read the info of 'How it

WHEN YOU START THIS PROGRAM, THERE ARE 3 .TXT FILES WHICH ARE LOADED AUTOMATICALLY
!! Please respect the structure and the spaces !!
1. Users.txt:
This file contains the users that you will be able to use for send Demands.
User_name id
Example:
Alice 1
Bob 2

2. Flavor.txt:
The second file has the flavor's id and flavor which the server will know as characteristics of Virtual Machines (VM) of every Virtual Node (VN).
What is a flavor?
This term is used to relate an id with some characteristics of virtual machines, this characteristics are:
core, hdd and memory
Example:
1 22,200,16
2 8,800,48
flavour 1 represents core: 22, hdd: 200 and 16 of memory. The ID of flavor is the value that you WILL use to set the specifications of your VM

3. Images.txt:
The last file loaded set the SO of the future VM used.
SO id
Example:
Windows 1
Linux 2
Example:
You have to set the imageID of your VM with 1 if it works with Windows.

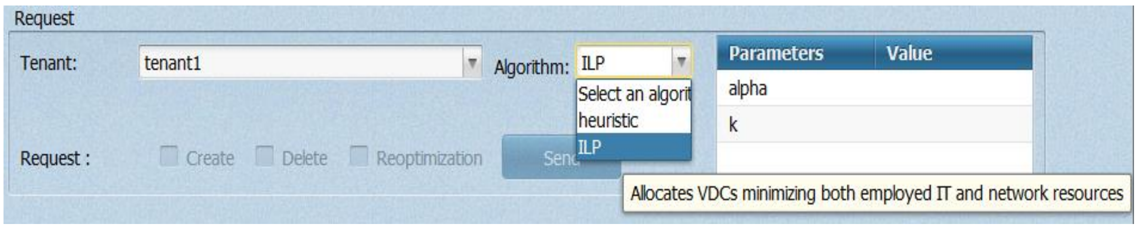
Please, now read the next button to know how to configure your topology and VDC.

Fig. 21 Auto-loaded files button HELP.

At this point, the main functionalities of the client interface are tested and the user could create the request of the VDC to the Orchestrator in order to proceed with the VDC provisioning.

4.4. VDC provisioning: Experimental validation

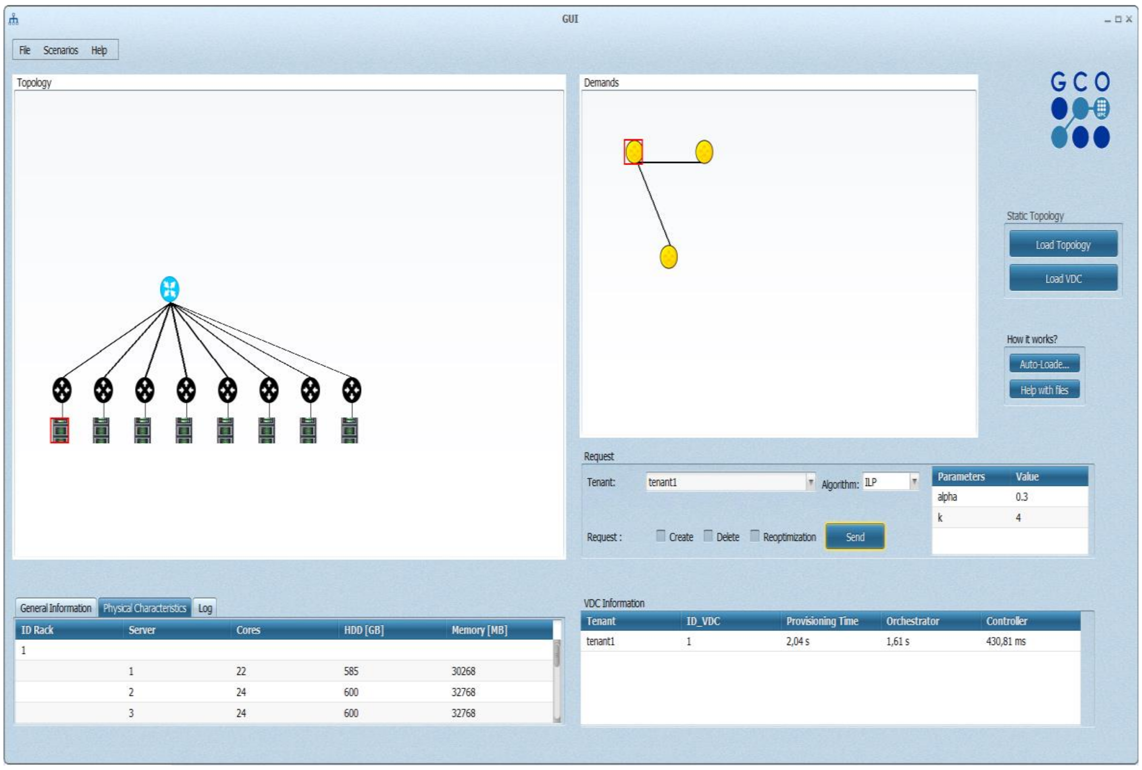
The user can proceed with the VDCE selecting the “Create” box and clicking in the button “Send”. Before that, it is necessary to select the algorithm depending on the data centre needs. As it can see in Fig. 22, the user can select one of the available algorithms and have to complete its parameters. Additionally, algorithms and parameters, have a brief description in order to know the functionality of each one. In this case it is selected ILP (Integer Linear Programming) algorithm which in short, hopes to find the allocation of the VMs in the lower number of servers and using the low number of network elements possible.



Allocates VDCs minimizing both employed IT and network resources

Fig. 22 User selecting ILP algorithm

With the VDC request sent, the Orchestrator processes the DC and VDC topology and applies the ILP algorithm in order to route the different VMs of the VDC on the DC. Immediately, its service Nova instance the VMs on the correct data centre’s servers and the SDN controller configures the network elements in order to establish the connections between them. At this point, the VDC is mapped in the DC. Now, the information of VMs shows in which rack and server are each VM instanced. Consequently, the resources information of each rack’s servers used are updated. As it can see in Fig. 23, rack: 1 has updated its resources of server: 1 because some VMs of the VN: 1 are mapped in this server. In the same way, the virtual link between two virtual nodes is mapped as two links in the data centre. For this reason, the wavelengths used in the links are updated from “free” to “busy”. An example of this situation is represented in Fig. 24.



Request

Tenant: tenant1 Algorithm: ILP

Request : ☐ Create ☐ Delete ☐ Reoptimization

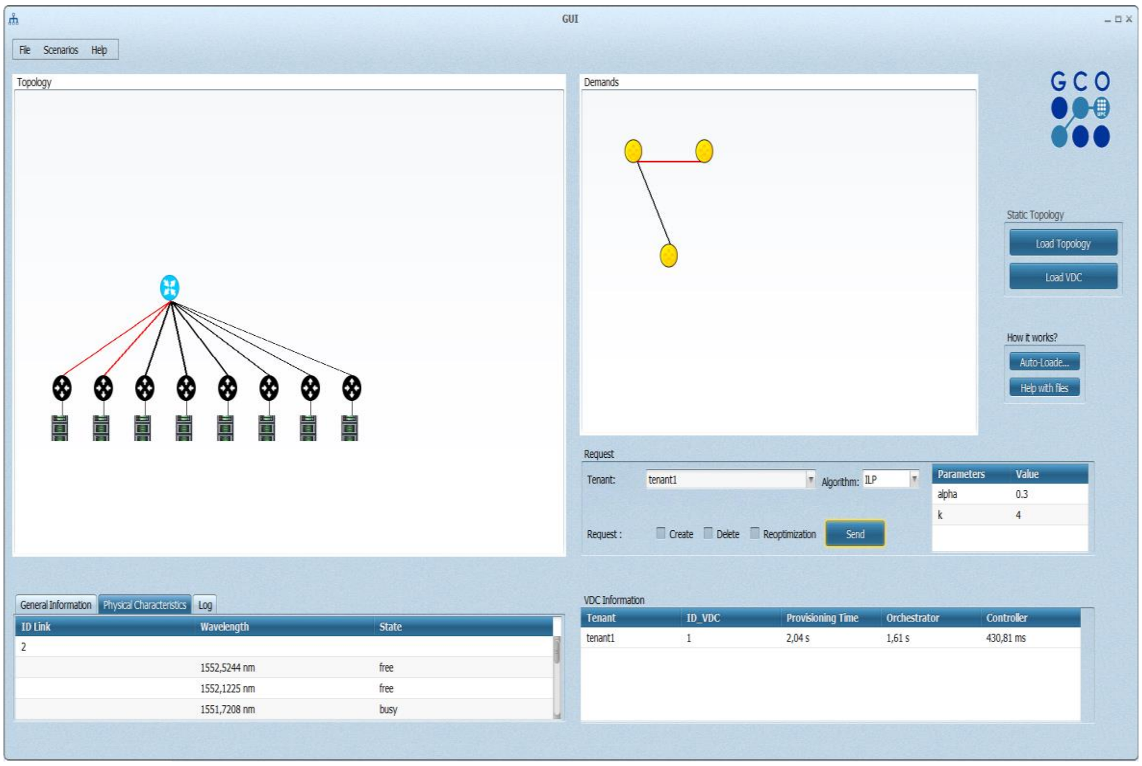
Parameters	Value
alpha	0.3
k	4

ID Rack	Server	Cores	HDD [GB]	Memory [MB]
1				
	1	22	585	30268
	2	24	600	32768
	3	24	600	32768

VDC Information

Tenant	ID_VDC	Provisioning Time	Orchestrator	Controller
tenant1	1	2,04 s	1,61 s	430,81 ms

Fig. 23 VDCE visualization. Servers' resources updated.



Request

Tenant: tenant1 Algorithm: ILP

Request : ☐ Create ☐ Delete ☐ Reoptimization

Parameters	Value
alpha	0.3
k	4

ID Link	Wavelength	State
2		
	1552,5244 nm	free
	1552,1225 nm	free
	1551,7208 nm	busy

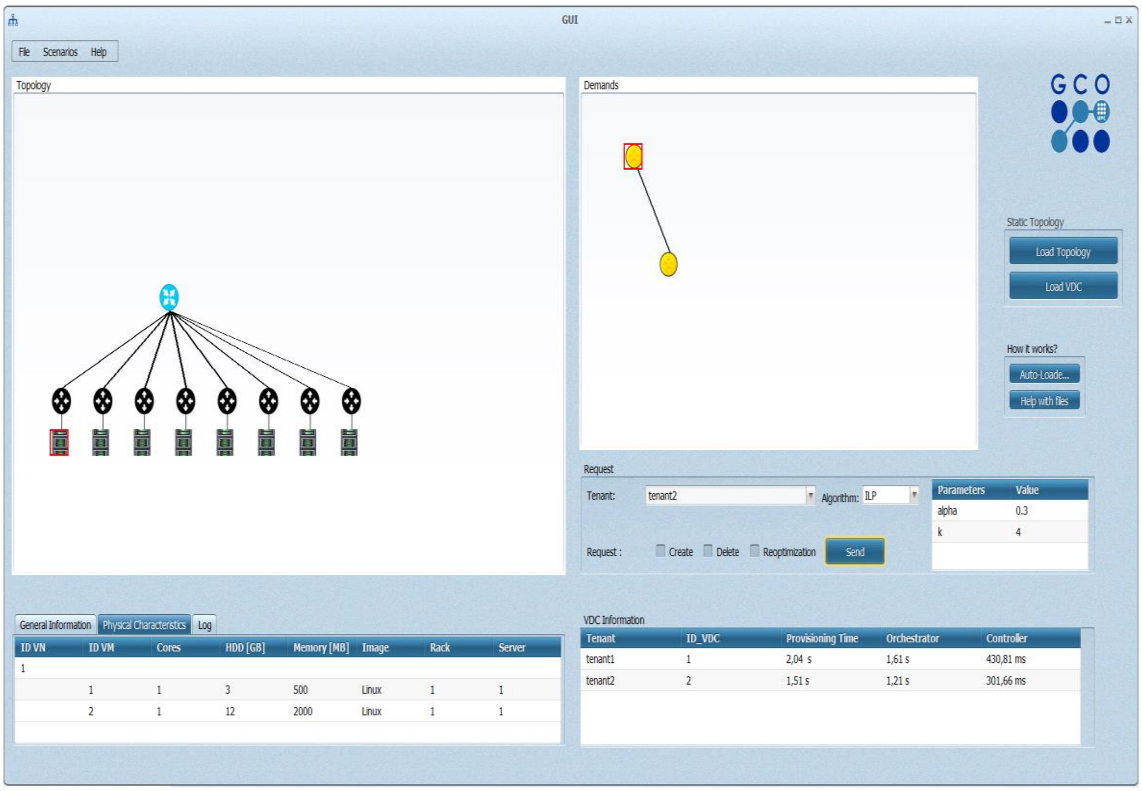
VDC Information

Tenant	ID_VDC	Provisioning Time	Orchestrator	Controller
tenant1	1	2,04 s	1,61 s	430,81 ms

Fig. 24 VDCE visualization. State of links' wavelengths updated.

As it can see in Fig. 23 and 24, it has been created a new row on the table in the VDC Information panel. This row contains the values of Provisioning Time which is the time that has been required to instance the VDC on the DC. In this case, the total delay of the operation has been of 2.04 s. The Orchestrator has spent 1.61 s in the process of the VDC topology establishment and reply a confirmation. On the other hand, the SDN controller has used 430.81 ms for configure the different network elements of the DC based on the route received from the Orchestrator. The representation of these values will allow to the user to create statistics of the performance of the different layers when a VDC request is sent.

In the following image (Fig. 25) it has been instanced a new VDC (id: 2) of a new tenant in order to verify the well-working in multi-tenant situations. In this case, VDC is formed by 2 VNs each one with their VMs which can be mapped in the same rack even in the same server than VMs of the first tenant. On the other hand, the virtual link of VDC: 2 could be instanced in the same link of VDC: 1 but never using the same wavelength. Besides, in the same way as the previous case, a new row has been created with the Provisioning Time information of tenant 2 VDC.



Request

Tenant: tenant2 Algorithm: ILP

Request : Create Delete Reoptimization

Parameters	Value
alpha	0.3
k	4

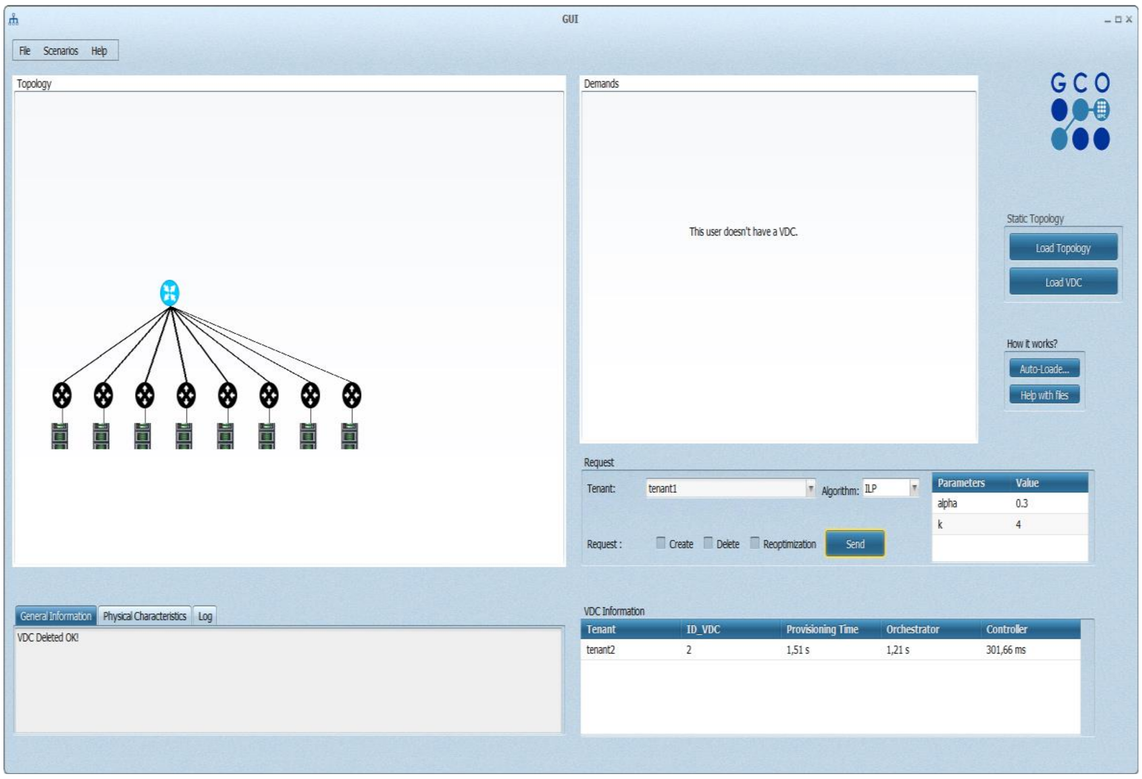
ID VN	ID VM	Cores	HDD [GB]	Memory [MB]	Image	Rack	Server
1							
	1	1	3	500	Linux	1	1
	2	1	12	2000	Linux	1	1

VDC Information

Tenant	ID_VDC	Provisioning Time	Orchestrator	Controller
tenant1	1	2,04 s	1,61 s	430,81 ms
tenant2	2	1,51 s	1,21 s	301,66 ms

Fig. 25 Multi-tenant VDC example.

By now, there are 2 VDCs optimally instanced on the DC but maybe they could be more optimal than when they were installed. Consequently, the user can send a re-optimization request to the Orchestrator to obtain a new relocation of all the VDCs previously instanced. This is possible, selecting de “reoptimize” box and clicking in the button “Send”. In case of re-optimize a data centre’s resources, in other words, reallocate the VMs and the virtual links previously instanced, both servers and links have been updated, leaving free the resources that were using and vice versa. Also that, it is possible to remove a VDC of the DC. Fig. 26 represents this situation, VDC of tenant: 1 has been deleted by sending a delete request to the server. If the request is processed correctly a confirmation message is shown on information panel and the VDC is automatically deleted, in consequence, the resources used by the VDC deleted have been freed.



This user doesn't have a VDC.

Tenant: tenant1 Algorithm: ILP

Request : Create Delete Reoptimization Send

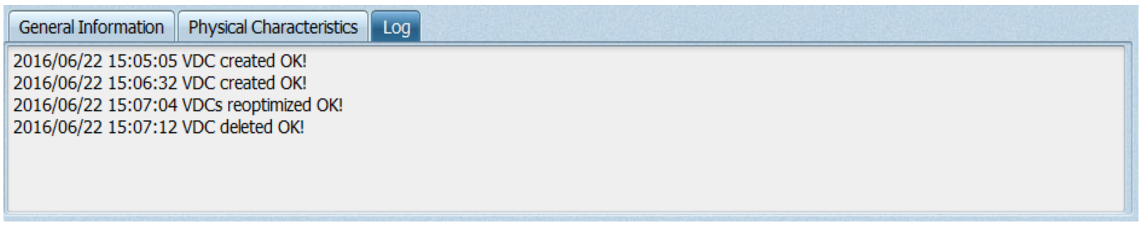
Parameters	Value
alpha	0.3
k	4

VDC Deleted OK!

Tenant	ID_VDC	Provisioning Time	Orchestrator	Controller
tenant2	2	1,51 s	1,21 s	301,66 ms

Fig. 26 VDC of tenant: 1 has been deleted.

Additionally, in the information panel there is a Log tag which contains the information of states of the different communications established with the orchestration server during the session, in order to have a control of every request sent. This can be a good support to known the state of correct communications or to know the errors which have been experimented. Also that, a Log text file is automatically created every day on the source path with this information.



```
2016/06/22 15:05:05 VDC created OK!
2016/06/22 15:06:32 VDC created OK!
2016/06/22 15:07:04 VDCs reoptimized OK!
2016/06/22 15:07:12 VDC deleted OK!
```

Fig. 27 Log tag with the session information.

Finally, as it will be explained in the following section, the toolbar in the upper zone, apart from loading the topology files or show information about them, allows the user to use the interface for other future scenarios. In this project, it is talked about connection intra-DC, in other words, connection between servers inside a DC, but with this option the user will be able to use the same application for future scenarios such as connections between DCs. It is important to know that all the information loaded while the use of “Intra-data centre” scenario will remain if the user decides to change the interface for another scenario.

5. CONCLUSIONS AND FUTURE WORKS

Since the beginning of this project, the main idea was the fulfilment of a graphical tool to facilitate the task of management and control of data centres. To provide this tool, there were proposed some objectives and requirements that they had to be achieved. Having reached the end of the project, all the features have been made successfully and even there have been implemented some extra functionalities that have appeared during the development.

This interface can be used primarily by data centres administrators; by the way it could also be used by enterprises, groups, or any entity that research or work with data centres, because it allows to perform visually tasks that were developed by text in a more complex way until now. As in the test section is discussed, this project does not only want to stay in the use for connections within a DC, but it is also intended that in the near future any other person can expand it for other scenarios such as inter-DC connections. This would allow to work in many different environments using just one application.

Personally, carrying this project out has meant a great challenge to me because during the course of its realization, there have been lots of difficulties and obstacles in the development and design as well as writing the documentation. Therefore, to overcome them I had to apply concepts learned in the degree and I have learned many new others. Overall, the experience of being able to realize this project has been remarkably good not only for the concepts learned, but because it has allowed me to improve in terms of organization and self-learning.

6. ABBREVIATIONS

API – Application Programming Interface

AWT – Abstract Window Toolkit

CPU – Central Processing Unit

DC – Data Centre

DCN – Data Centre Network

GUI - Graphical User Interface

HTTP – Hypertext Transfer Protocol

JSON – JavaScript Object Notation

MMF – Multi Mode Fibre

ODC – Optical Data Centre

RAM – Random-Access Memory

REST – Representational State Transfer

SDN – Software Defined Network

TOR – Top of the Rack

VDC – Virtual Data Centre

VDCE – Virtual Data Centre Embedding

VM – Virtual Machine

VN – Virtual Node

WDM – Wavelength Division Multiplexing

7. REFERENCES AND BIBLIOGRAPHY

[1] Faizul Bari, Md.; Boutaba, Raouf; Esteves, Rafael; Zambenedetti Granvillem Lisandro; Podlesny, Maxim; Golam Rabbani, Md.; Zhang, Qi; Faten Zhani, Mohamed, "Data Center Network Virtualization: A Survey", *IEEE Communicaions Surveys & Tutorials*, vol. 15, num. 2, pp. 909-928, September 2012.

[2] Correa, E. S.; Fletscher, L.A.; Botero, J. F., "Virtual Data Centre Embedding: A Survey", *IEEE Latin America Transactions*, vol.13, num.5, pp.1661-1670, May 2015.

[3] Kachris, Christoforos; Tomkos, Ioannis, "A Survey on Optical Interconnects for Data Centers", *IEEE Communicaions Surveys & Tutorials*, vol.14, num.4, pp.1021-1036, January 2012.

[4] Vicente Rosa, Raphael; Esteve Rothenberg, Christian; Madeira, Edmundo, "Virtual Data Center Networks Embedding Through Software Defined Networking", *IEEE Network Operations and Management Symposium (NOMS)*, Krakow, pp.1-5, May 2014.

[5] Pagès, Albert; Perelló Jordi; Agraz, Fernando; Spadaro, Salvatore, "Optimal VDC Service Provisioning in Optically Interconnected Disaggregated Data Centers", *IEEE Communications Letters*, April 2016.

[6] Pagès, Albert; Pérez Sanchís, Miguel; Peng, Shuping; Perelló, Jordi; Simeonidou, Dimitra; Spadaro, Salvatore, "Optimal Virtual Slice Composition Toward Multi-Tenancy Over Hybrid OCS/OPS Data Center Networks", *IEEE/OSA Journal of Optical Communications and Networking*, vol.7, num.10, pp.974-986, October 2015.

[7] ITU-T Recommendation G.694.1, "Spectral grids for WDM applications: DWDM frequency grid", February 2012.

[8] Stryer, Paul, "Understanding Data Centers and Cloud Computing", *Global Knowledge*.

[9] Joyanes Aguilar, Luis, "Cloud computing and data center: The new industrial revolution. How will it change the work in business and organizations?", *Journal Sociedad y Utopia*, nº.36, pp.111-128, November 2010.

[10] Kreutz, Diego; M. V. Ramos, Fernando; Esteves Veríssimo, Paulo; Esteve Rothenberg; Azodolmolky, Siamak; Uhlig, Steve, "Software-Defined Networking: A Comprehensive Survey", *Proceedings of the IEEE*, vol.103, num.1, pp.14-76, January 2015.

[11] Miller, Rich, “As Cloud Wars Intensify, Google Adds More Data Centers”, DataCenter Frontier, http://datacenterfrontier.com/cloud-wars-intensify-google-adds-more-data-centers/, March 2016.

[12] “What is OpenStack?”, OpenStack., http://www.openstack.org/software/.

[13] “OpenDaylight: Open Source SDN Platform”, OPENDAYLIGHT, https://www.opendaylight.org/.

[14] “OpenDayligh Project”, Wikipedia, https://en.wikipedia.org/wiki/OpenDaylight_Project.

[15] “Cisco Global Cloud Index: Forecast and Methodology, 2014-2019 White Paper”, Cisco, http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.html, April 2016.

[16] “Package javax.swing”, Oracle, https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html.

[17] “Package java.net”, Oracle, https://docs.oracle.com/javase/7/docs/api/java/net/package-summary.html.

[18] “GSON user guide”, Google, https://sites.google.com/site/gson/gson-user-guide.

[19] “GSON”, Wikipedia, https://en.wikipedia.org/wiki/Gson.

[20] “Eclipse Is…”, Eclipse, https://www.eclipse.org/.

[21] Baca, Steve, “Cloud Computing: What It Is and What It Can Do for You”, Global Knowledge, March 2010.

[22] Hales, John, “SND and Cloud Computing”, Global Knowledge, June 2014.

[23] “Data Center Architecture Overview”, Cisco, http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/DC_Infra2_5/DCInfra_1.html, May 2008.

[24] “¿Qué es un Data Center?”, acensblog, http://www.acens.com/blog/que-es-un-data-center.html, April 2008.

[25] “Ventajas e inconvenientes de la Virtualización del Centro de Datos”, ITCIO.es, http://www.itcio.es/virtualizacion-centro-datos/informes/1005208011402/ventajas-inconvenientes-virtualizacion.1.html, March 2009.

[26] Jiménez, Almudena, “¿Qué es la virtualización del Centro de Datos?”, ITCIO.es, http://www.itcio.es/virtualizacion-centro-datos/informes/1005213011402/virtualizacion-centro-datos.1.html, March 2009.

[27] “Virtual Data Center”, TELEFONICA, http://www.movistar.es/grandes-empresas/soluciones/fichas/virtual-data-center/.

[28] Muñoz, Santiago, “Los Retos de la Fibra Óptica en el Centro de Datos”, Datacenter Dynamics, http://www.datacenterdynamics.es/focus/archive/2014/11/los-retos-de-la-fibra-%C3%B3ptica-en-el-centro-de-datos, November 2014.

[29] “OpenFlow protocol guide: SDN controllers and apps”, SearchSDN, http://searchsdn.techtarget.com/guides/OpenFlow-protocol-tutorial-SDN-controllers-and-applications-emerge.

[30] Banks, Ethan, “SDN basics: Understanding centralized control and programmability”, SearchSDN, http://searchsdn.techtarget.com/tip/SDN-basics-Understanding-centralized-control-and-programmability.

[31] “How Do SND Southbound APIs Work?”, sdx central, https://www.sdxcentral.com/sdn/definitions/southbound-interface-api/.

[32] “Software-defined data center”, Wikipedia, https://en.wikipedia.org/wiki/Software-defined_data_center.

[33] “Software-Defined Networking (SDN) Definition”, Open Networking Foundation, https://www.opennetworking.org/sdn-resources/sdn-definition.

[34] “SNMP tutorial”, ManageEngine, https://www.manageengine.com/network-monitoring/what-is-snmp.html.