UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Analysis and Simulation of Optical Switches for Intra-Datacenter Networks

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Odiri Adetupe Osemwengie

In partial fulfilment

of the requirements for the degree in

TELECOMUNICATION SYSTEM ENGINEERING

Advisor: Jaume Comellas Colome

Barcelona, June 2018

Abstract

This project consists of doing a feasibility study for the interconnections within a data center. There are so many architecture designs of a data center network (DCN), which is one of the most basic and fundamental in a data center because its directly reflects on its scalability, cost, fault-tolerance, agility and power consumption.

In this project, we are going to focus on a three-tier topology named “Data Center Fabric”. “Data Center Fabric” is an architecture deployed by Facebook, and we are going to design and simulate the traffic performance of two cases. These two cases are referent to “Data Center Fabric”. The first case, which is case A, is the topology itself while the second case, which is case B, is a modified topology of “Data Center Fabric”.

Resum

Aquest projecte consisteix en realitzar un estudi de factibilitat per a les interconnexions dins d'un centre de dades. Hi ha tants dissenys d'arquitectura d'una xarxa de centre de dades (DCN), que és un dels més bàsics i fonamentals en un centre de dades perquè es reflecteix directament en la seva escalabilitat, cost, tolerància a errors, agilitat i consum d'energia.

En aquest projecte, ens enfocarem en una topologia de tres nivells anomenada "Data Center Fabric". "Data Center Fabric" és una arquitectura implementada per Facebook, i anem a dissenyar i simular el rendiment del trànsit de dos casos. Aquests dos casos es refereixen a "Data Center Fabric". El primer cas, que és el cas A, és la topologia en si, mentre que el segon cas, que és el cas B, és una topologia modificada de "Data Center Fabric".

Resumen

Este proyecto consiste en realizar un estudio de factibilidad para las interconexiones dentro de un centro de datos. Hay tantos diseños de arquitectura de una red de centro de datos (DCN), que es uno de los más básicos y fundamentales en un centro de datos porque se refleja directamente en su escalabilidad, costo, tolerancia a errores, agilidad y consumo de energía.

En este proyecto, nos enfocaremos en una topología de tres niveles llamada "Data Center Fabric". "Data Center Fabric" es una arquitectura implementada por Facebook, y vamos a diseñar y simular el rendimiento del tráfico de dos casos. Estos dos casos se refieren a "Data Center Fabric". El primer caso, que es el caso A, es la topología en sí, mientras que el segundo caso, que es el caso B, es una topología modificada de "Data Center Fabric".

Acknowledgements

A special thanks to my tutor and supervisor Jaume Comellas Coleme that have help me throughout the all project. And, a special thanks to Behnam Shariati, for assisting me whenever I needed it. Without them this project would not have been possible.

Revision history and approval record

Revision	Date	Purpose
0	01/06/2018	Document creation
1	23/06/2018	Document revision
2	30/06/2018	Finalization of Document

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Odiri Adetupe - Student	odiripatience@hotmail.com
Jaume comellas - Project Director	comellas@tsc.upc.edu
Behnam Shariati - coTutor	behnam.sh@ieee.org

Written by:		Reviewed and approved by:	
Date	30/06/2018	Date	02/07/2018
Name	Odiri Adetupe	Name	Jaume Comellas
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables:

1. Introduction

A data center (or datacenter) is a facility composed of networked computers and storage that businesses or other organizations use to organize, process, store and disseminate large amounts of data. A business typically relies heavily upon the applications, services and data contained within a data center, making it a focal point and critical asset for everyday operations.

Data centers have their roots in the huge computer rooms of the 1940s, typified by ENIAC, one of the earliest examples of a data center. Early computer systems, complex to operate and maintain, required a special environment in which to operate. Many cables were necessary to connect all the components, and methods to accommodate and organize these were devised such as standard racks to mount equipment, raised floors, and cable trays (installed overhead or under the elevated floor). A single mainframe required a great deal of power and had to be cooled to avoid overheating. Security became important – computers were expensive and were often used for military purposes.

The boom of data centers came during 1997–2000. Companies needed fast Internet connectivity and non-stop operation to deploy systems and to establish a presence on the Internet. Installing such equipment was not viable for many smaller companies. Many companies started building very large facilities, called **Internet data centers** (IDCs), which provide commercial clients with a range of solutions for systems deployment and operation. New technologies and practices were designed to handle the scale and the operational requirements of such large-scale operations. These practices eventually migrated toward the private data centers and were adopted largely because of their practical results. Data centers for cloud computing are called **cloud data centers** (CDCs). But nowadays, the division of these terms has almost disappeared, and they are being integrated into a term "data center".

Communications in data centers today are most often based on networks running the IP protocol suite. Data centers contain a set of routers and switches that transport traffic between the servers and to the outside world which are connected according to the data center network architecture.

1.1. Objectives

The principal objective of this project is to do a feasibility study on the interconnections of a data center, so a three-tier topology named “Data Center Fabric” was chosen as the center of our study. The “Data Center Fabric” was chosen because it is a recent architecture design which was deployed by Facebook and is a good example how a data center interconnections works.

The “Data Center Fabric” follows a multi-rooted tree base network topology composed of three layers of network switches, namely layer 1, layer 2, and layer 3. The layer 2 switches interconnect multiple layer 1 switches together. All the layer 2 switches are connected to each other by layer 3 switches. layer 3 switches are also responsible for connecting the data center to the Internet. In this project we will be focusing on layer 2 and 3.

In this way, the objectives of this project can be summarized as follows:

1. Analysis of the network interconnection between layer 2 and layer 3 in a “Data Center Fabric”.

2. Analysis of the network interconnection between layer 2 and layer 3 in a modified “Data Center Fabric”.

3. Conclusion of both the analysis of the network interconnection of a “Data Center Fabric” and a modified “Data Center Fabric”.

It is important to note that the documentation of this project will not enter the development and Implementation phase due to cost, a development and implementation will be very expensive.

Therefore, from here it is understood that **this document will not explicitly include hardware implementation concepts**.

1.2. Requirements and Specifications

1.2.1. Project requirements

- MATLAB: matlab is a software application that will be needed to design the network topology of the data center since the project is a design project. The matlab version is release “2018a”, an earlier version to release ”2018a” can’t be used because some functions in the project codes may not functioned well provoking instabilities in the analysis and simulation in the “matlab script”

1.2.2. Project specifications

- The number of clusters/pods must be 8.

- The number of fabric switches must be 4.

- The number of the spine planes must be the same as the number of fabric switches, which in this case is 4.

- The number of spine switches in a spine plane is 3.

- The bandwidth of a link in the network is 50 Gbps

1.3. Project plans

This section presents the structure of the project, tasks to be overcome, a Gantt diagram with the temporary planning and the incidents that arose during its execution.

1.3.1. Project Structure

The project is divided into two sections: Analysis and design of the network interconnection of layer 2 and layer 3 (case A), Analysis and design of the network interconnection of layer 2 and layer 3 (case B). Both sections are not independent from each other, we must go from one section to another (back and forward) until the aims of this project is obtained.

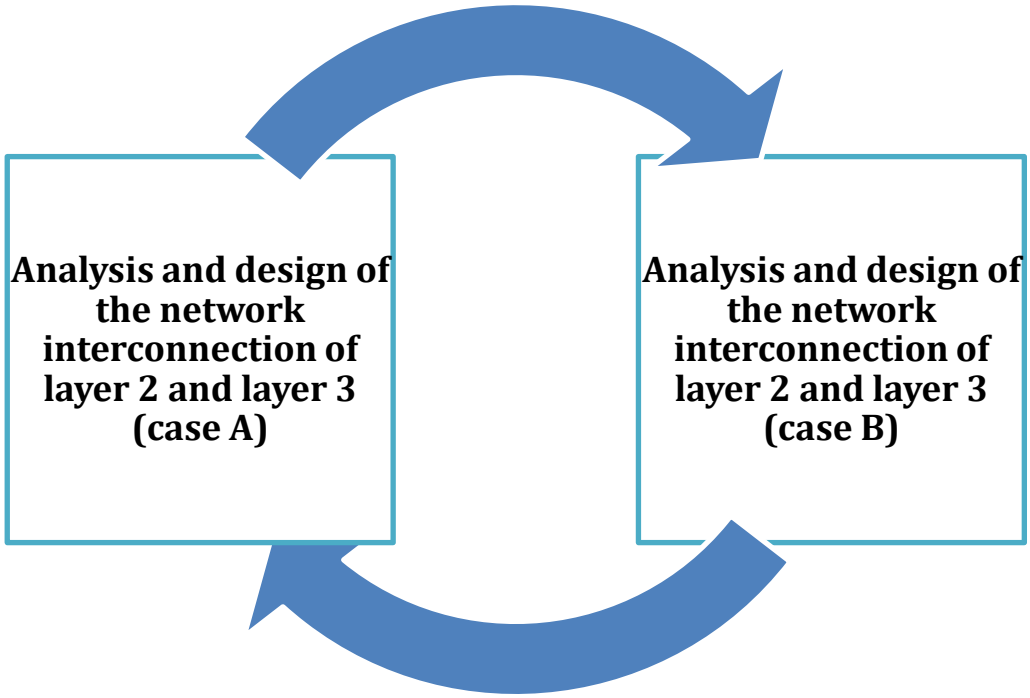

Figure 1: Project Structure

1.3.2. *Work Packages, Tasks and Milestones*

<table>
<tr><td colspan="2">Project: Analysis and design of the network interconnection of layer 2 and layer 3 (case A).</td><td>WP ref: WP 1</td></tr>
<tr><td colspan="2">Major constituent: MATLAB</td><td></td></tr>
<tr><td colspan="2" rowspan="2">Brief description:
Choosing an architecture structure for the data center network, variables, and factors to consider in this structure.</td><td>Planned start date: 05/03/2018
Planned end date: 07/05/2018</td></tr>
<tr><td>Start event: First meeting with the project director and supervisor
End event: Final report delivery.</td></tr>
<tr><td colspan="2">Internal task T1: Define the physical links between switches of layer 1 and 2.
Internal task T2: Define the physical links between switches of layer 2 and 3.
Internal task T3: Implement the routing and wavelength assignment in the interconnection between layer 2 and 3 for case A.
Internal task T4: Analysis the traffic performance in the interconnection between layer 2 and 3 for case A.</td><td>Deliverables:
Project Critical Review</td><td>Dates:
07/05/2018</td></tr>
</table>

<table>
<tr><td colspan="2">Project: Analysis and design of the network interconnection of layer 2 and layer 3 (case B)</td><td>WP ref: WP 2</td></tr>
<tr><td colspan="2">Major constituent: MATLAB</td><td></td></tr>
<tr><td colspan="2" rowspan="2">Short description:
Changing the topology of the architecture structure by implementing a ring topology in layer 3 for the data center network.</td><td>Planned start date: 08/05/2018
Planned end date: 02/07/2018</td></tr>
<tr><td>Start event: First meeting with the project director and supervisor
End event: Final report delivery</td></tr>
<tr><td colspan="2">Internal task T1: Define the physical links in the ring topology between switches of layer 3.
Internal task T2: Implement the Routing and wavelength assignment in the interconnection between layer 2 and 3 for case A.
Internal task T3: Analysis the traffic in the interconnection between layer 2 and 3 for case B.
Internal task T4: Comparation between case A and B.</td><td>Deliverables:
Final report</td><td>Dates:
02/07/2018</td></tr>
</table>

Milestones

WP#	Task#	Short title	Milestone / deliverable	Date (week)
1	4	PCR	Project Critical Review	07/05/2018
2	4	FR	Final Report	02/07/2018

1.3.3. *Gantt diagram*

Weeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Tasks																	
WP1-T1																	
WP1-T2																	
WP1-T3																	
WP1-T4																	
WP2-T1																	
WP2-T2																	
WP2-T3																	
WP2-T4																	

Table 1: Gantt Diagram

1.3.4. Incidences and modifications of work plan

The initial plan was to do a feasibility study on using space division multiplexing (SDM) techniques for the interconnections of data centre but since we were not going to do a hardware implementation and only do a software implementation, so it was not necessary. In software, the SDM can be visualized as a normal physical link.

Another incident is that during the wavelength assignment of an optical channel, it is very difficult to ensure that the wavelength assignment is done correctly even after revising it several times and redoing the MATLAB codes due to the large number of optical paths found.

2. Case Study of the "Data Center Fabric" Topology:

DC fabric was chosen because it is topology deployed by Facebook and the simplicity of its infrastructure design. Facebook's network infrastructure needs to constantly scale and evolve, rapidly adapting to our application needs. The amount of traffic from Facebook to Internet – we call it "machine to user" traffic – is large and ever increasing, as more people connect and as we create new products and services. However, this type of traffic is only the tip of the iceberg. What happens inside the Facebook data centers – "machine to machine" traffic – is several orders of magnitude larger than what goes out to the Internet.

Their back-end service tiers and applications are distributed and logically interconnected. They rely on extensive real-time "cooperation" with each other to deliver a fast and seamless experience on the front end, customized for each person using their apps and their site. They are constantly optimizing internal application efficiency, but nonetheless the rate of their machine-to-machine traffic growth remains exponential, and the volume has been doubling at an interval of less than a year.

To satisfy the traffic exponential growth and keep the networking infrastructure simple, Facebook deployed the data center fabric.

2.1. About "Data Center Fabric" Topology

A data center fabric is a system of switches and servers and the interconnections between them that can be represented as a fabric. Because of the tightly woven connections between nodes (all devices in a fabric are referred to as nodes), data center fabrics are often perceived as complex, but it is the very tightness of the weave that makes the technology inherently elegant. A data center fabric allows for a flattened architecture in which any server node can connect to any other server node, and any switch node can connect to any server node (server refers to both compute and storage). The term "fabric" comes from the criss-cross nature of a fabric switch, which is often visualized as a checkerboard of connections, like figure 2.

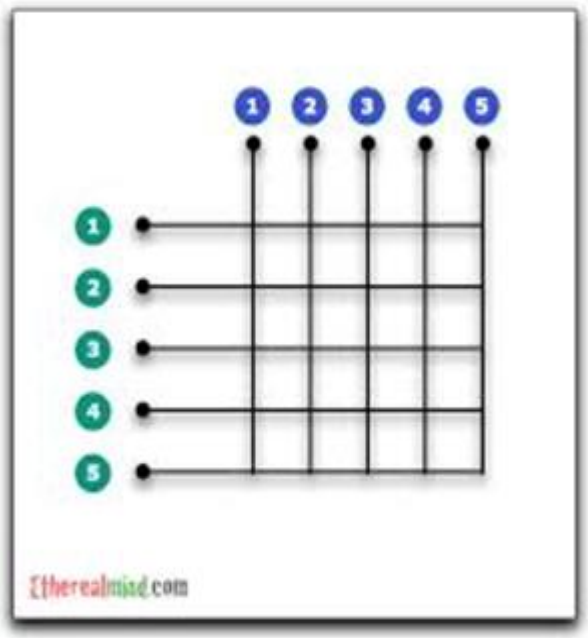

Figure 2: The diagram for a fabric switch

2.2. Anatomy of Data Center Fabric

DC fabric consists of cluster/pod, each cluster/pod aggregates a bunches of Tor switches which itself are connected to the 4 fabric switches. The pod is not defined by any hard-physical properties; it is simply a standard “unit of network”. Pods are interconnected by four spine planes, which can scale up to 48 spine switches. Each Tor switches are connected to all 4 fabric switches in a pod while each fabric switches are connected to a spine plane; the ToR switch is a switch which a manages a group of servers placed in a rack. The number of inputs ports in fabric switches can be equal to the number of output ports which can give us up to statically non-blocking performance both sides. The bandwidth of a link in a port scale in Gigabits but it can also scale to higher ports speeds as well. DC fabric offers many parallel paths between servers. Although figure 3 looks simply but it is impossible to picture it to an actual scale that becomes a just like figure 3.

The DC fabric follows a multi-rooted tree base network topology composed of three layers of network switches, namely layer 1, layer 2, and layer 3. The Tors switches are layer 1, the fabric switches are layer 2, and the spine planes are layer 3.

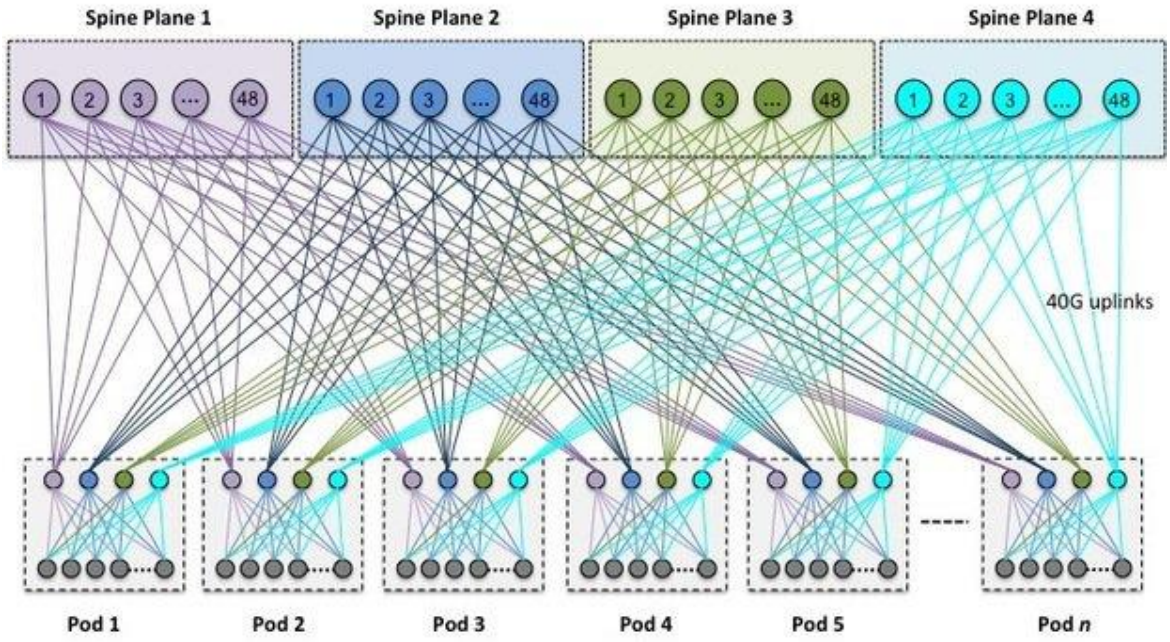

Figure 3: Data Center Fabric

2.3. Characteristics of Data Center Fabric

- Each path has equal performance
- The load on the DC fabric is distributed, different flows are taken different paths coinciding on this large distributed infrastructure.
- DC fabric can survive multiple device failures, components failures and links failures without any production impact.
- It offers us a very easy way to scale, when we need more compute capacity we add more pods, which are easy units of growth, and when we need more performance we add more spine switches on the spine planes.
- It automates everything related to itself, making it easy to deal with and it is faster to deploy.

3. Methodology / project development:

From this point on, we will enter directly into the detailed explanation of how we have designed our topology using MATLAB. The case study of DC fabric is divided into two case which are case A and case B, the project specifications are applied in both cases which is:

- The number of clusters/pods is 8.
- The number of fabric switches by cluster is 4.
- The number of the spine planes is 4.
- The number of spine switches in a spine plane is 3.
- The bandwidth of a link is 50 Gbps
- The number of ToR switches is 2 (optional).

We will be centring our study in the interconnections between layer 2 and layer 3 because it is the core part of this network.

3.1. Topology Design.

3.1.1. Case A

The first step was to design the clusters, the design created from matlab is figure 4. We decided to put only two ToRs instead of 48ToRs to visualize better the figure, there are two ToRs by cluster and each ToRs are connected to the 4 fabric switches in the cluster. There are 8 clusters and each link are bidirectional.

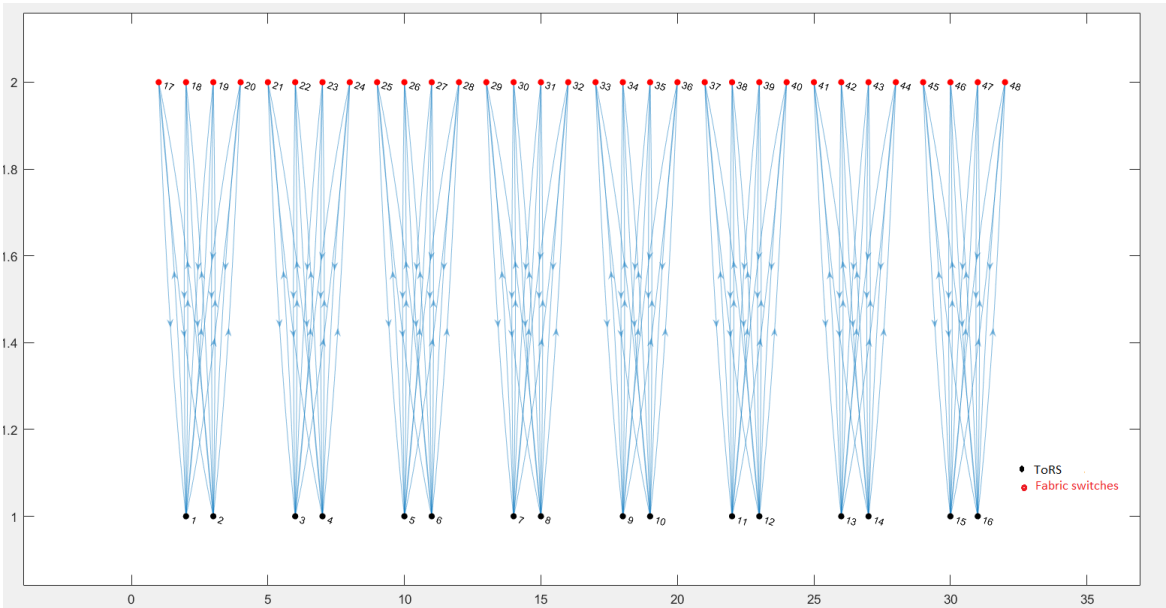

Figure 4: Clusters for Case A

The next step was to connect the clusters to spine planes, the design created from matlab is figure 5. Each fabric switches are connected to a spine plane, it means that the fabric switch is connected to the spines switches within a local spine plane.

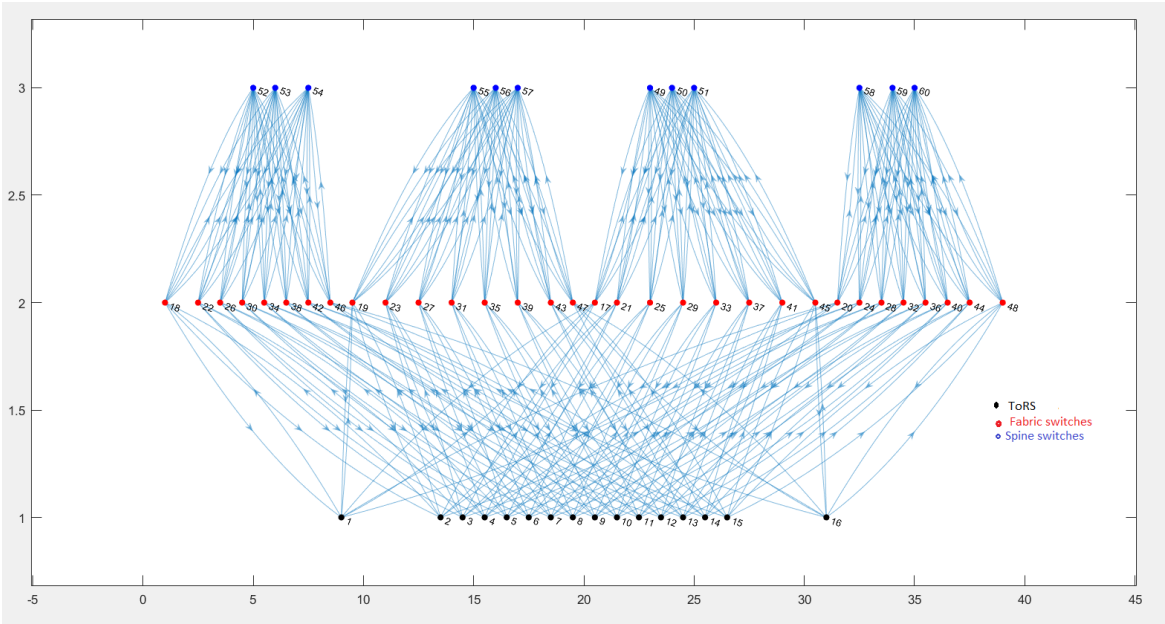

Figure 5: Clusters Interconnecting with Spine Planes for Case A

Since we are only interested in the interconnection between layer 2 and layer 3 so we subtract layer 2 and layer 2 and reset the id of the switches. The result of this design is figure 6. Each of the fabric switches connected to the same spine plane are from different cluster.

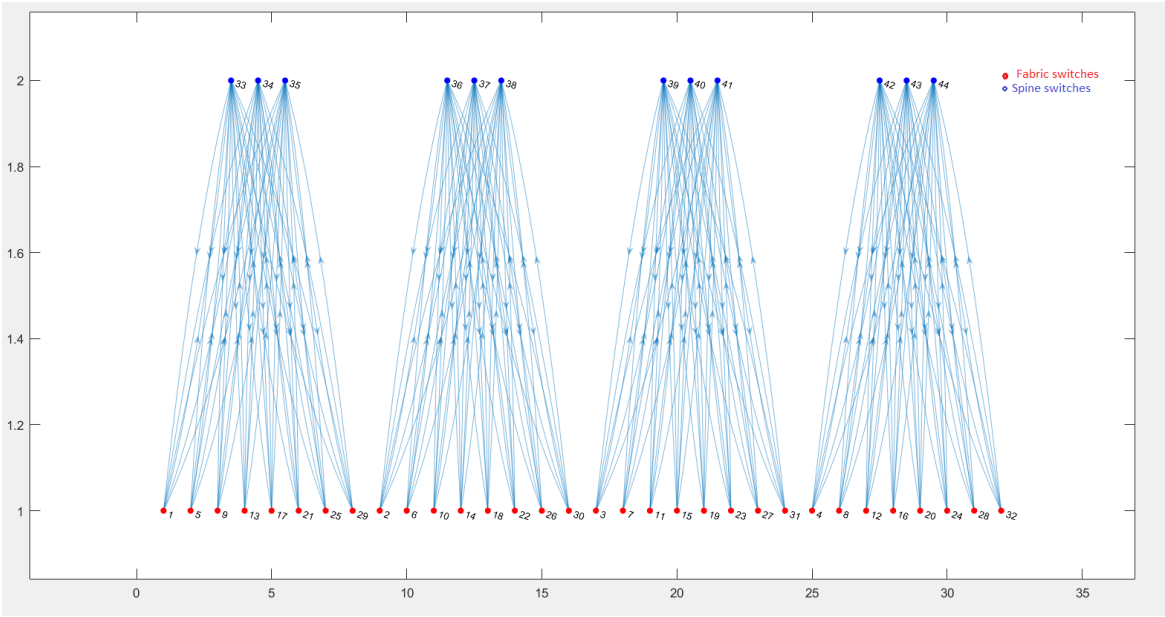

Figure 6:Fabric Switches Interconnecting with Spine Planes for Case A

The structural shape of figure 6 so that it can be visualize better is:

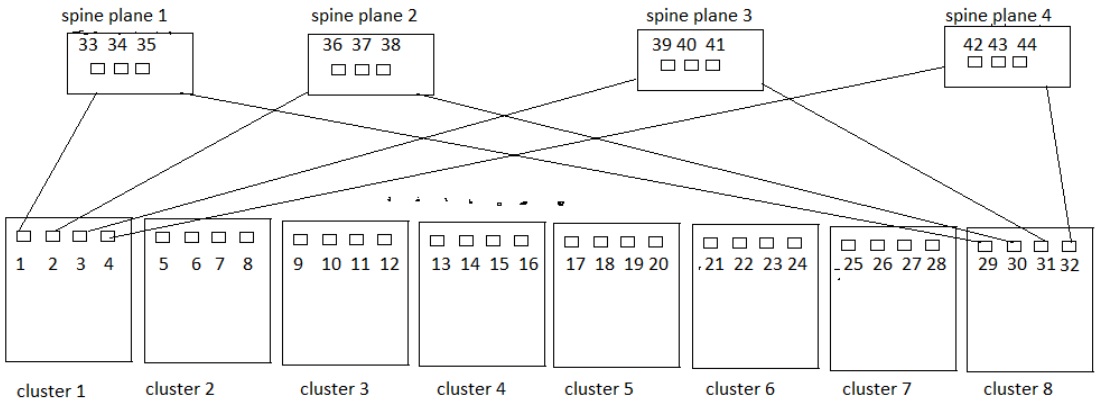

Figure 7:Cluster Interconnecting with Spine Planes for Case A

3.1.2. Case B

The topology used in this case is a little different from case A. In this case, the spine planes are connected by a ring topology. The design steps are the same as in case A, just added more links to the spine planes, the design created from matlab is figure 8:

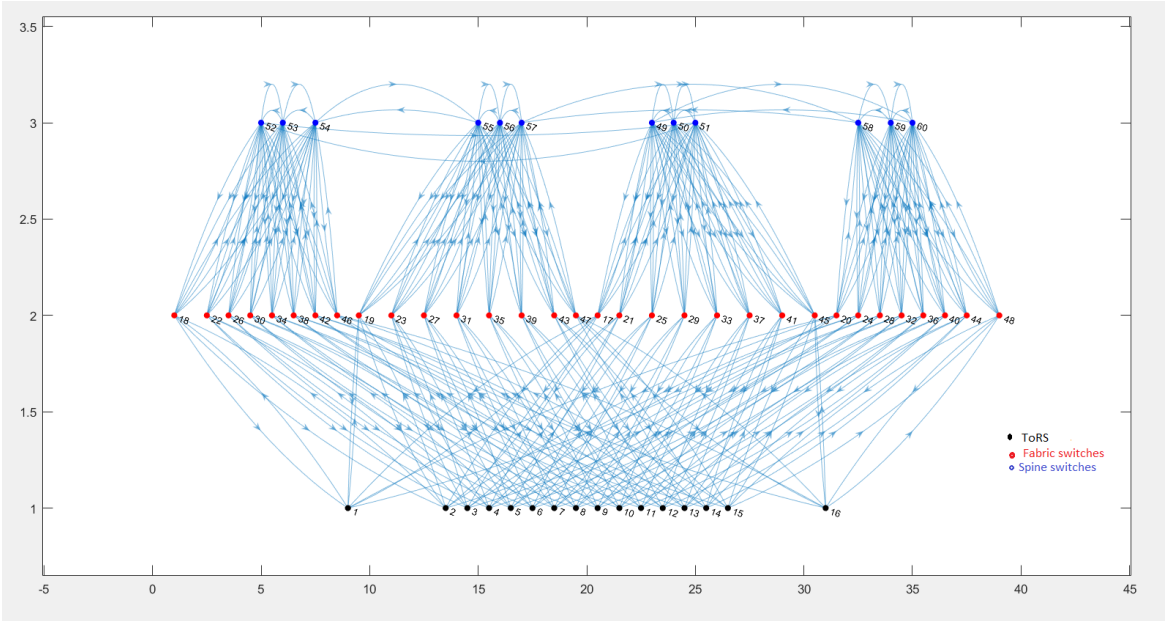

Figure 8:Clusters Interconnecting with Spine Planes for Case B

Just like case A since we are only interested in the interconnection between layer 2 and layer 3 so we subtract layer 2 and layer 2 and reset the id of the switches. The result of this design is figure 9:

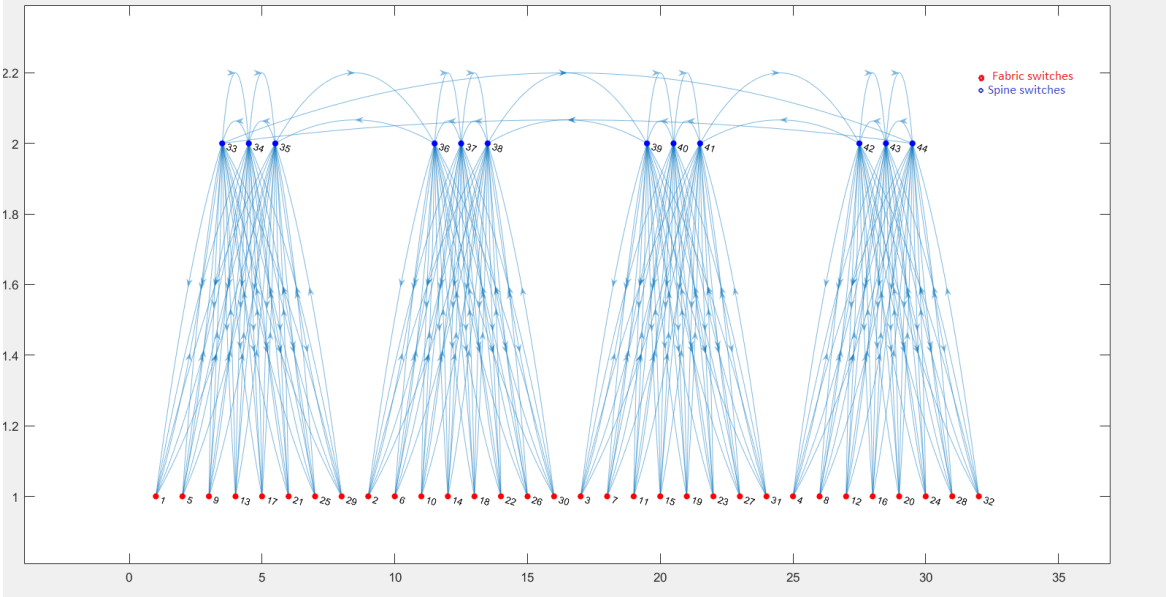

Figure 9: Fabric Switches Interconnecting with Spine Planes for Case B

The structural shape of figure 9 so that it can be visualize better is:

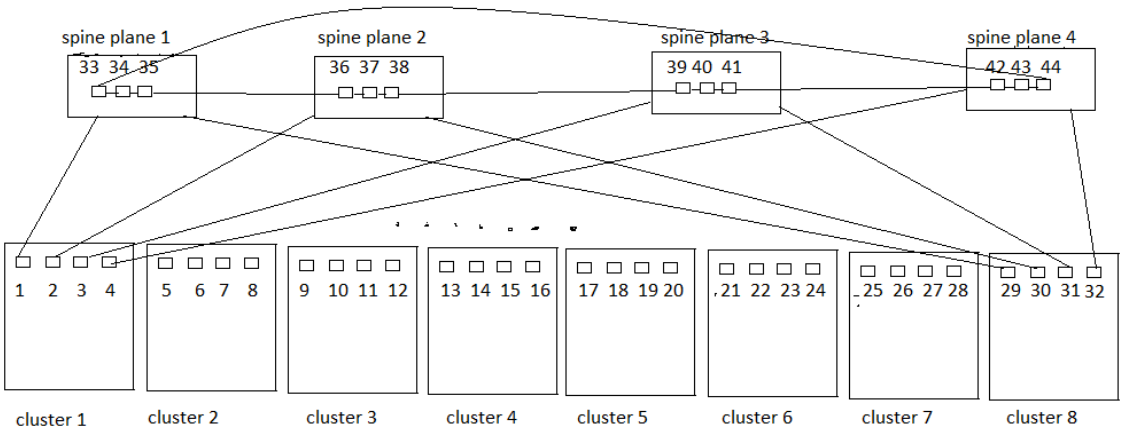

Figure 10: Cluster Interconnecting with Spine Planes for Case B

3.2. Traffic Matrix Generated for Case A and Case B

Since we are interested in the communication between clusters, we are going to generate an inter-cluster traffic matrix (32x32) with different distributions for Case A and Case B, the dimension of the matrix is 32x32 because the number of fabric switches are 32. The matrix uses a “normal” probability distribution with different means and standard deviation.

Having 32 rows of the traffic matrix, “normal” distributions are distributed as follows:

Rows id	Mean (μ)	Standard deviation (σ)
row 1 to row 8	100	5
row 9 to row 16	80	5
row 17 to row 24	60	5
row 25 to row 32	45	5

Table 2: Normal distribution in traffic matrix

An example of the traffic matrix is seen in figure 8:

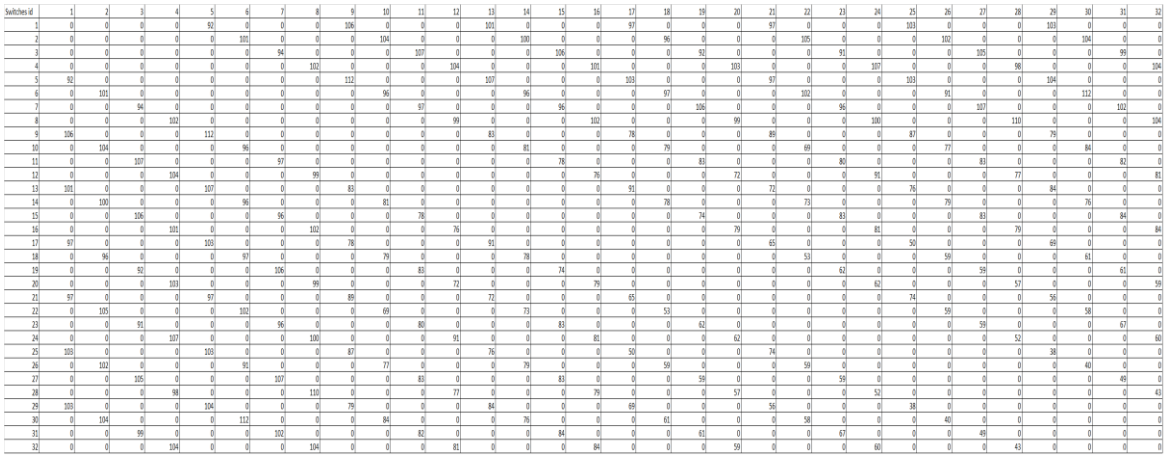

Figure 11:Inter-cluster Traffic Matrix within local spine plane

In figure 8, the matrix cells which have the value of zero is because there is no link between a source id and its destination id. Since the links are bidirectional, the upper triangular matrix must be equal to the lower triangular matrix meaning the traffic matrix is a symmetric matrix. This inter-cluster traffic matrix only generates traffic to fabric switches which are connected to the same local spine plane.

We also generated another inter-cluster traffic matrix (case B) which can only by applied in Case B because in this case the matrix not only generates traffic to fabric switches which are connected to the same local spine plane but also to other fabric switches in other clusters.

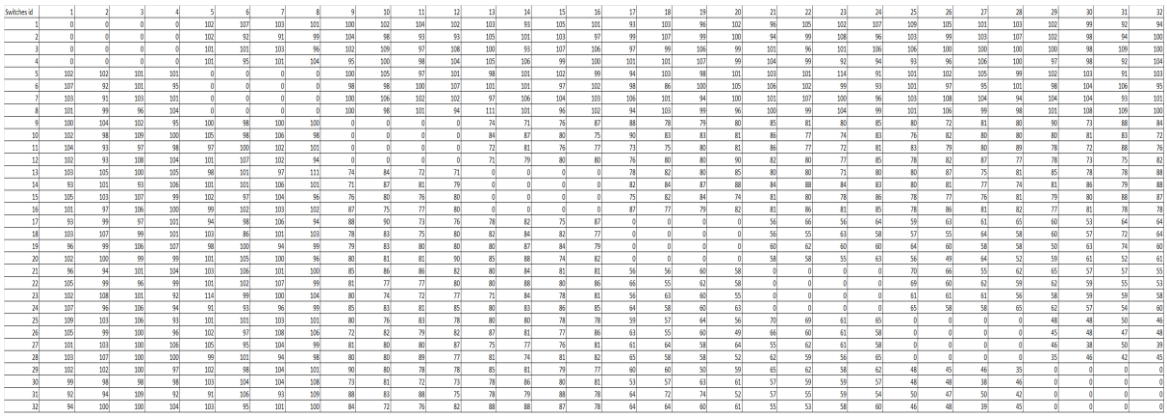

Figure 12:Inter-cluster Traffic Matrix between fabric switches

3.3. Routing and Wavelength Assignment (RWA)

There are different types of RWA problems and so also there are different solutions to them. To solve the routing problem in the topology, we used two methods to see its behaviour: Fixed path routing and Fixed Alternate routing. These two methods can only be use if the demand is fixed which means all the paths that are to be set up in the network are known beforehand. To solve the wavelength assignments, we used the First fit algorithm. We use fixed path routing or fixed alternate routing to find a path in the network, after founding a path, we use the first fit algorithm to assign a wavelength to this path. When a wavelength is assign to a path, the path becomes a “light path” or “optical channels”. A Light path is a path from the source to the destination in optical domain, meaning data remains in optical domain without any optical-electronic- optical conversion in between the source and the destination. These “light paths” must be chosen without violating any of the following two constraints:

- Wavelength Capacity constraint: states that a wavelength may be used only once per link at any given point in time.
- Wavelength Continuity constraint: states that the “light path” uses the same wavelength on all links it traverses from source to destination.

3.3.1. Fixed path routing and first fit algorithm

A fixed route is selected for each source-destination pair of nodes in the network. The routes are fixed; they may only change if there is a change in the topology of the network. We calculate the shortest path off-line using the Dijkstra’s algorithm. In figure 13, an example of a fixed routing is illustrated, the coloured links are the fixed route. Paths that share links in common can’t be seen in the figure.

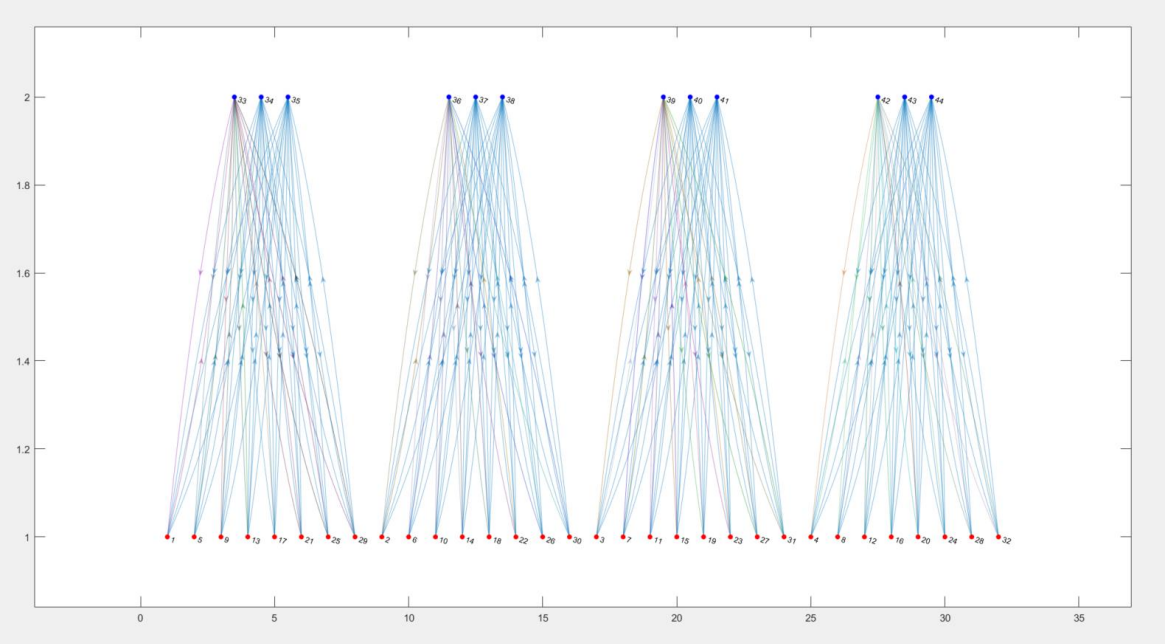

Figure 13: Fixed path routing for Case A

After finding the paths, the next step was to use the first fit algorithm to assign the wavelengths to the paths founded. In this scheme, all wavelengths are numbered. When searching for available wavelengths, a lower numbered wavelength is considered before a higher-numbered wavelength. The first available wavelength is then selected. This scheme requires no global information. Knowing that the bandwidth of a link is 50 Gbps

s, we divide the traffic matrix by the bandwidth giving us the number of channels to assign to a source-destination node pair and then we round the number to an integer:

$$channel\ matrix = round(\frac{traffic\ matrix}{50})$$

Using the channel matrix, we determined the number of wavelength to assign to a path. When a wavelength is assigned to path, the path become automatically “light path”. If a wavelength can’t be assigned to a path due to the limitation in resource of the network, the path is descanted. In figure 14, an example of a “light path” is illustrated, the maximum number of wavelengths is 8. Many of the paths that share links in common that couldn’t be seen in figure 13 has been descanted that why figure 13 still looks like figure 14. The difference will be seen in the data analysis and simulation.

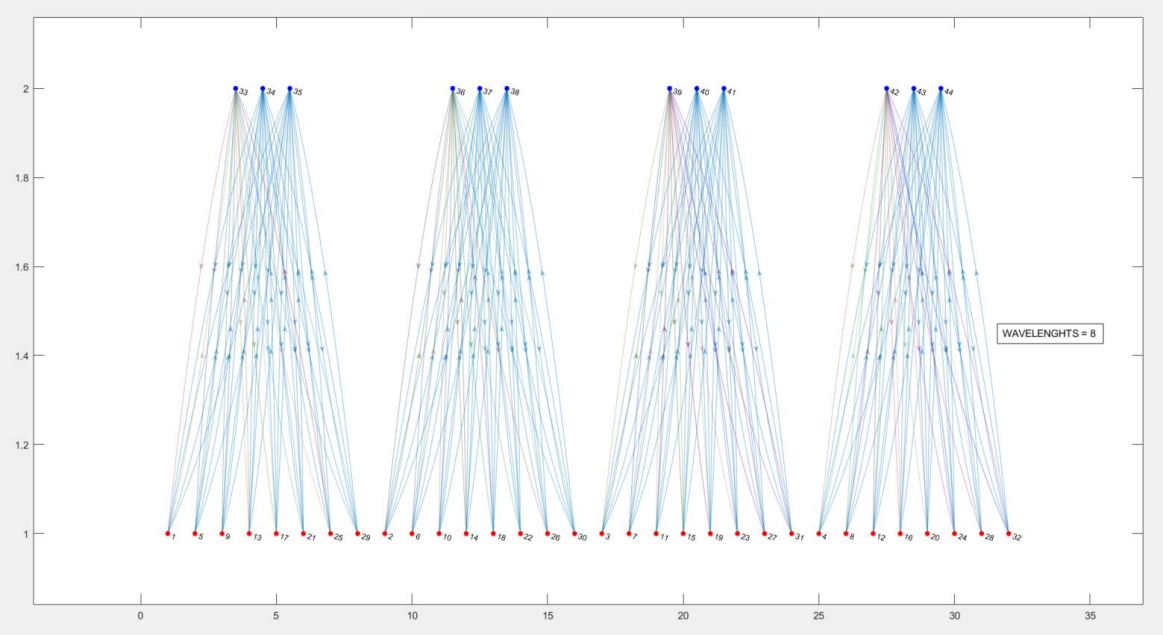

Figure 14: "Light paths" for case A

3.3.2. Fixed alternate routing and first fit algorithm

In fixed-alternate routing, each node in the network must maintain a routing table that contains an ordered list of a number of fixed routes to each destination node. For example, these routes may include the shortest-path route, the second-shortest-path route, the third-shortest-path route, etc. A primary route between a source node and a destination node is defined as the first route in the list of routes to node d in the routing table at node s. An alternate route between source and destination is any route that does not share any links (is link-disjoint) with the first route in the routing table at source node.

To find the paths knowing that the bandwidth of a link is 50 Gbps, we use the channel matrix:

$$channel\ matrix = round(\frac{traffic\ matrix}{50})$$

The channel matrix determines the number of alternative paths that can be assign to a source-destination node pair, for example if the number of channels is 3, the alternative paths for the source-destination node pair will be 2. In figure 15, an example of a fixed alternate routing is illustrated: there are few paths that share links in common.

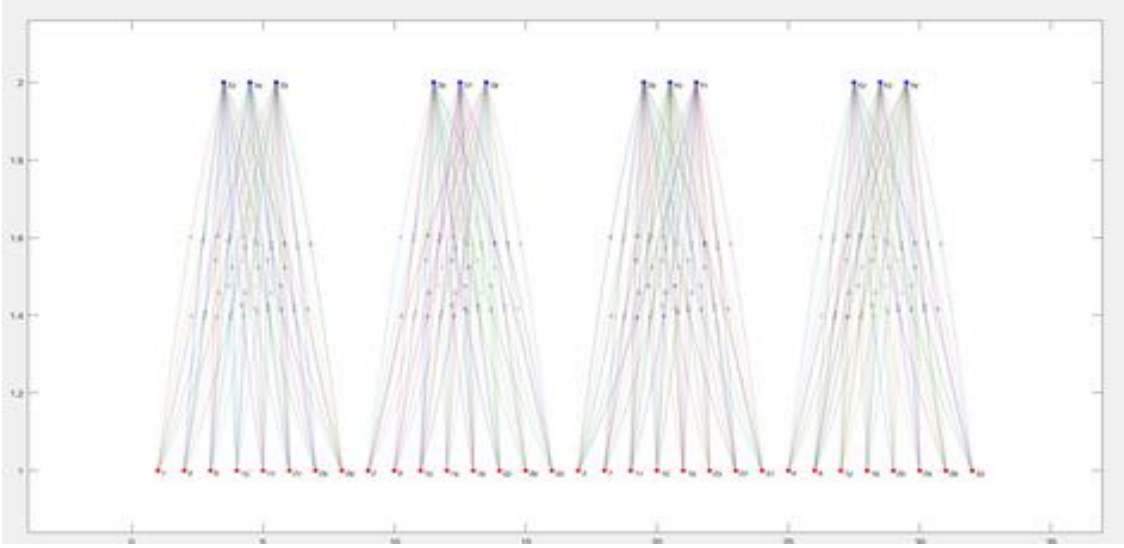

Figure 15 Fixed alternate routing for case A

After finding the paths, the next step was to use the first fit algorithm to assign the wavelengths to the paths founded. Just like in the case of fixed routing, the wavelength with the lowest number is selected from the available wavelengths except that we are not using the channel matrix since it has already been used to calculate the paths, so we are just going to assign only one wavelengths to a path. In figure 16, an example of a “light path” is illustrated, the maximum number of wavelengths is 8. Few of the paths that share links in common that couldn’t be seen in figure 15 has been descanted that why figure 15 still looks like figure 16. The difference will be seen in the data analysis and simulation.

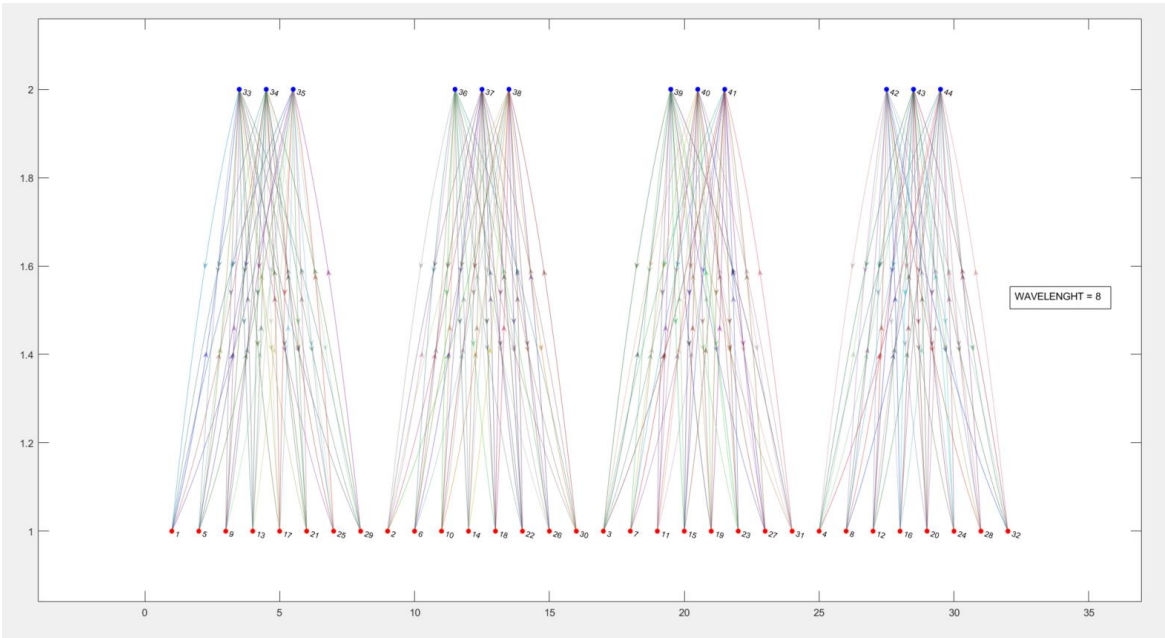

Figure 16: "Light paths" for case A

4. Simulation Results

There are four matlab scripts with different data analysis and simulations:

- "Case_A_Fixed.m": it belongs to case A and uses the fixed path routing. The matlab codes can be found in **Annex A.**
- "Case_A_Alternate.m": it belongs to case A and uses the fixed alternate routing. The matlab codes can be found in **Annex B.**
- "Case_B_Fixed.m": it belongs to case B and uses the fixed path routing. The matlab codes can be found in **Annex C.**
- "Case_B_Alternate.m": it belongs to case A and uses the fixed alternate routing. The matlab codes can be found in **Annex D.**

Note that the topology the DC fabric of case A is different from case B. Spine planes in the topology of case B are connected by a ring network.

4.1. "Case_A_Fixed.m".

A sample of how the wavelengths are assigned to the path using the fixed path routing and first fit technique is shown in figure 17.

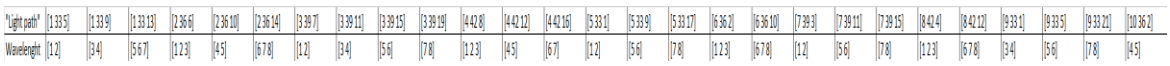

Figure 17: "Light path" with wavelength = 8

As explained before, the channel matrix determines the number of wavelengths assigned to a path. when a connection is about to be set up, the network looks for the wavelength available with the lowest index to assign to it. If there is no available wavelength, the connection is blocked. If the number of wavelength assign to a connection is 2 and the network can only allocate one wavelength, the path/connection will be descanted. If the maximum number of wavelengths is 8, it means the link of the network can accommodates 8 wavelengths.

To be able to understand the network behaviour we calculate and simulate the traffic performance, it is shown in figure 18. The traffic acceptance is calculated the following way:

$$traffic\ acceptance = \frac{number\ of\ traffic\ accepted}{total\ number\ of\ traffic}$$

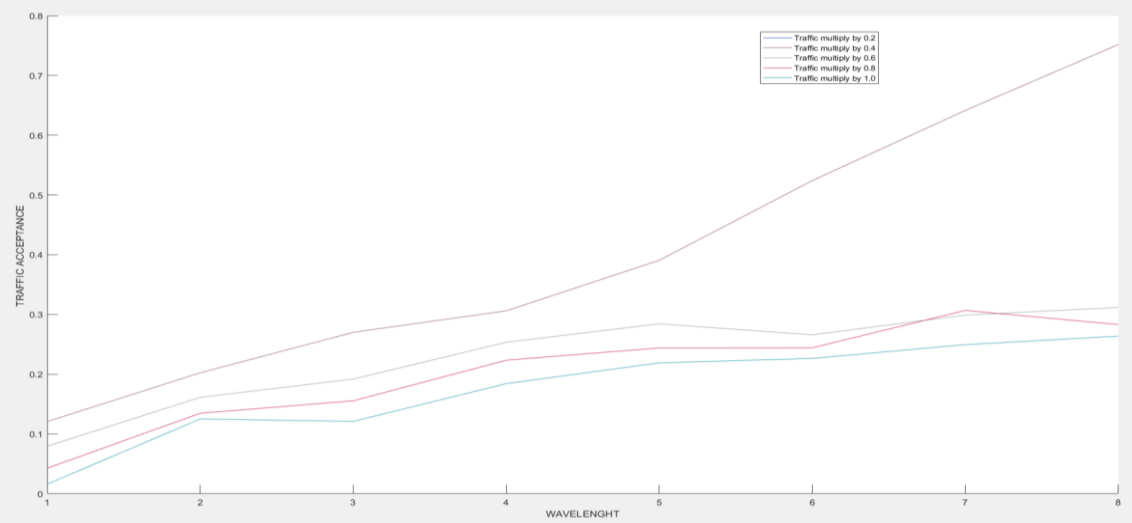

Figure 18: Traffic performance for Case A using fixed routing

The total number of traffic is the sum of all the values in the inter-cluster traffic matrix why number of traffic accepted is the sum of all the values of the traffic of the “Light paths”.

The traffic acceptance is calculated for when the maximum number of wavelengths is 1,2, 3...8. When the inter-cluster traffic matrix is reduced, the traffic acceptance increased.

The traffic acceptance reaches 1 when the maximum number of wavelengths is 32 (figure 19).

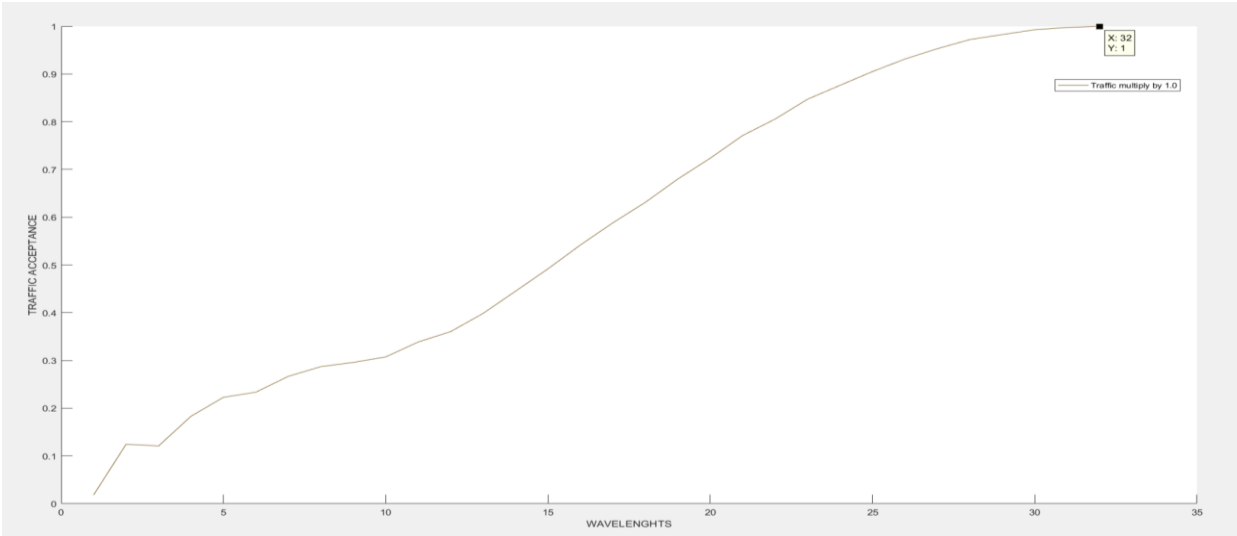

Figure 19: When traffic acceptance is 1

Figure 20 shows the maximum number of wavelengths use by each fabric switches when the maximum number of wavelengths is 32. The fabric switches with the lower ID has a lesser wavelengths usage than the fabric switches with the higher ID. The plot has a uniform distribution.

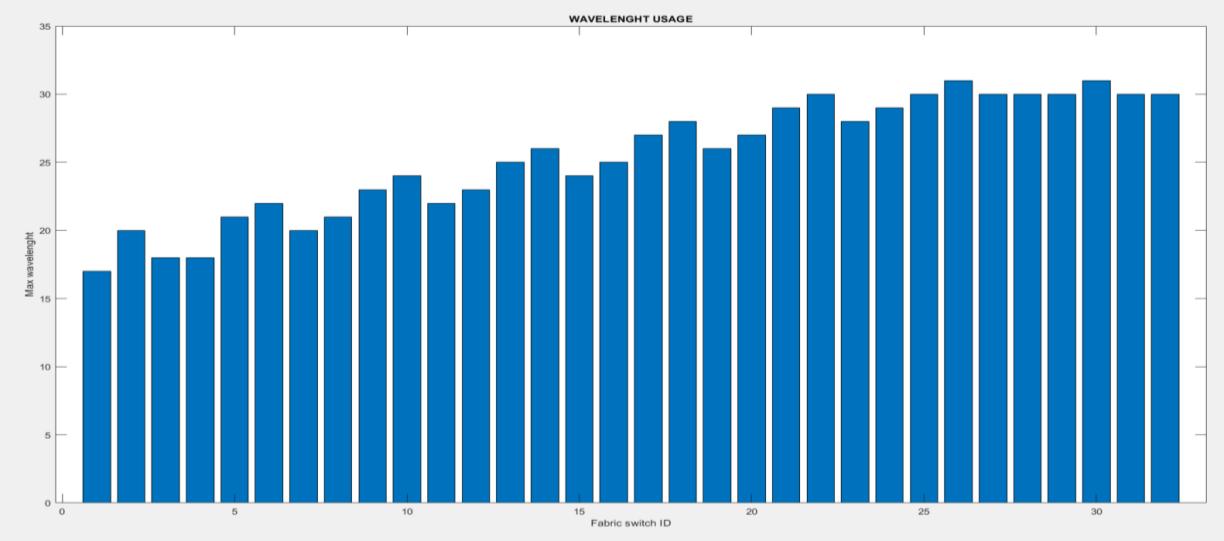

Figure 20: Maximum number of wavelengths use by fabric switches

4.2. "Case_A_Alternate.m".

A sample of how the wavelengths are assigned to the path using the fixed alternate routing and first fit technique is shown in figure 21.

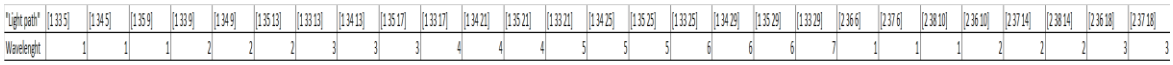

Figure 21: "Light path" with wavelength = 8

As explained before, the channel matrix determines the number of alternative paths assigned to a source-destination node pair. Just like the when a connection is about to be set up, the network looks for the wavelength available with the lowest index to assign to it. If there is no available wavelength, the connection is blocked. Each path is assigned a wavelength.

To be able to understand the network behaviour we calculate and simulate the traffic performance, it is shown in figure 22.

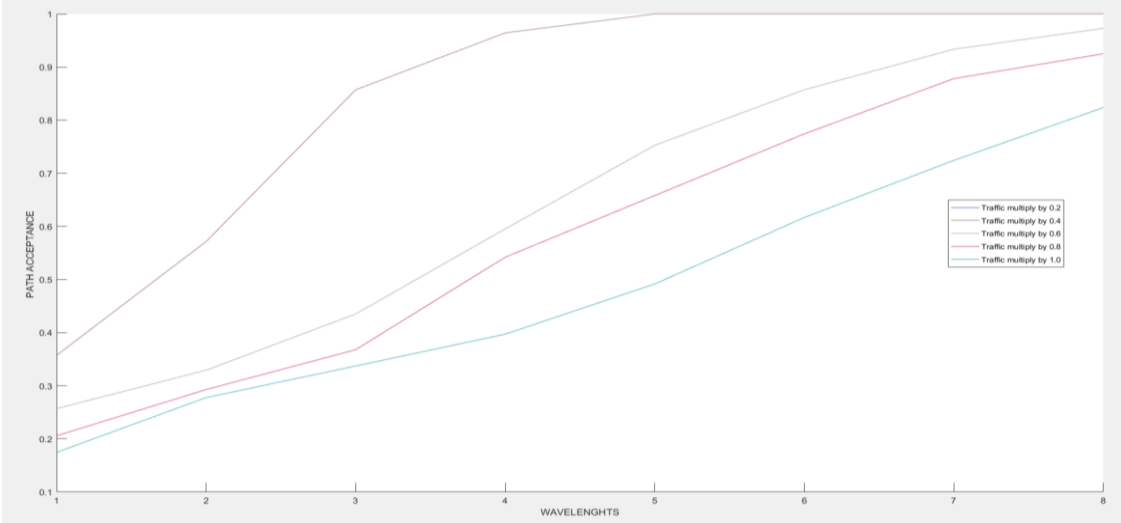

Figure 22: Traffic performance for Case A using fixed alternate routing

When the inter-cluster traffic matrix is reduced, the traffic acceptance increased.

The traffic acceptance reaches 1 when the maximum number of wavelengths is 12 (figure 23).

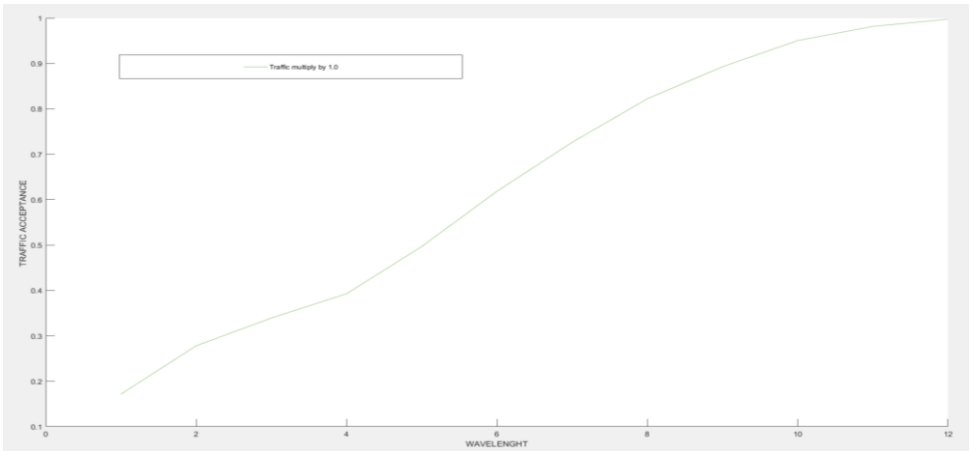

Figure 23: When traffic acceptance is 1

Figure 24 shows the maximum number of wavelengths use by fabric switches when the maximum number of wavelengths is 12. The fabric switches with the lower ID has a lesser wavelengths usage than the fabric switches with the higher ID. The plot has a little uniform distribution.

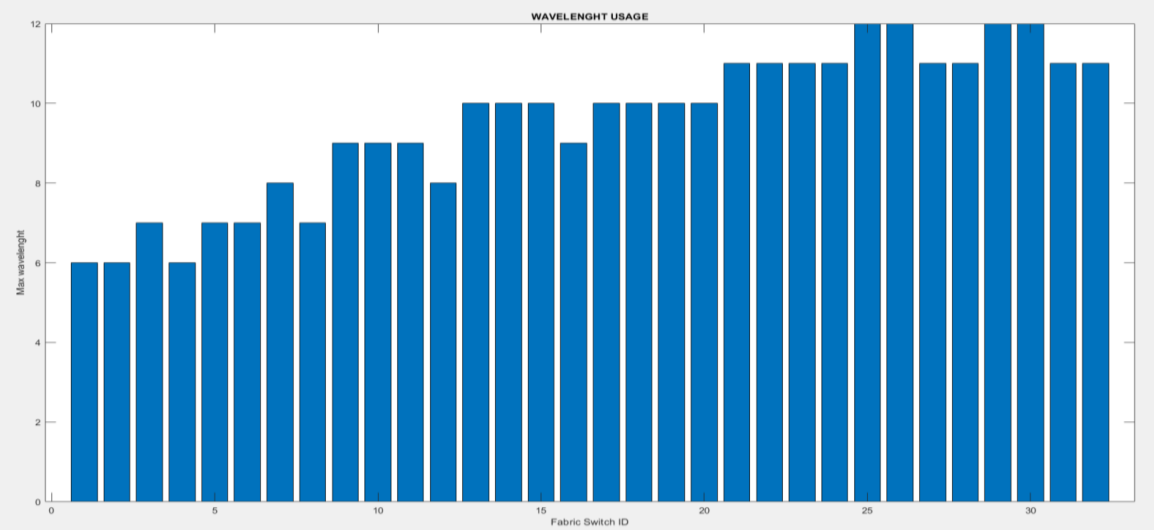

Figure 24: Maximum number of wavelengths use by fabric switches

4.3. "Case_B_Fixed.m".

The sample of how the wavelengths are assigned to the path using the fixed path routing and first fit technique is the same as shown in figure 17.

To be able to understand the network behaviour we calculate and simulate the traffic performance and it was the same as figure 18 because path found were the same as "Case_A_Fixed.m" so we decided to remove some of the links. The new topology with the removed links is shown in figure 25:

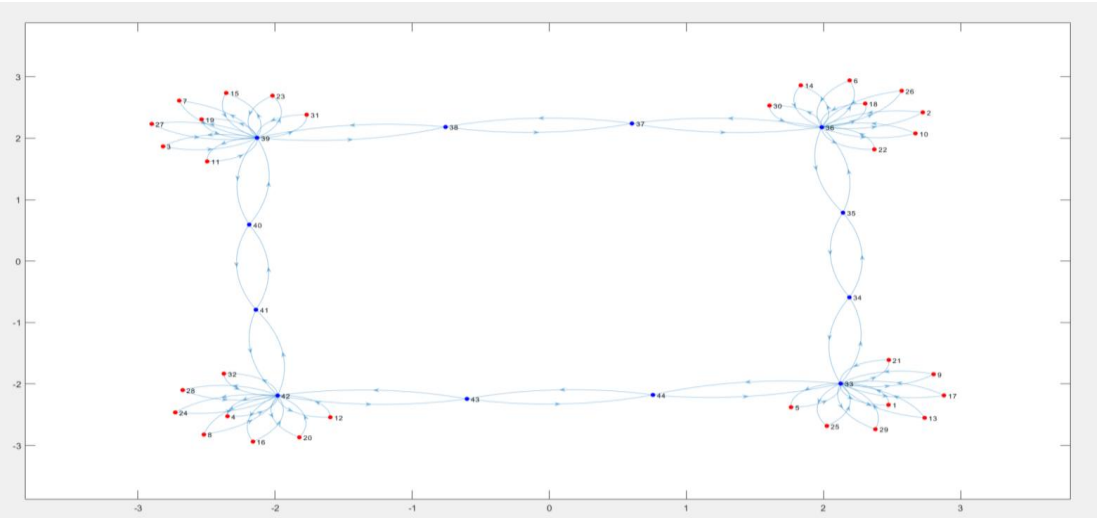

Figure 25: Case B topology with some links removed

The structural shape of figure 25 so that it can be visualize better is:

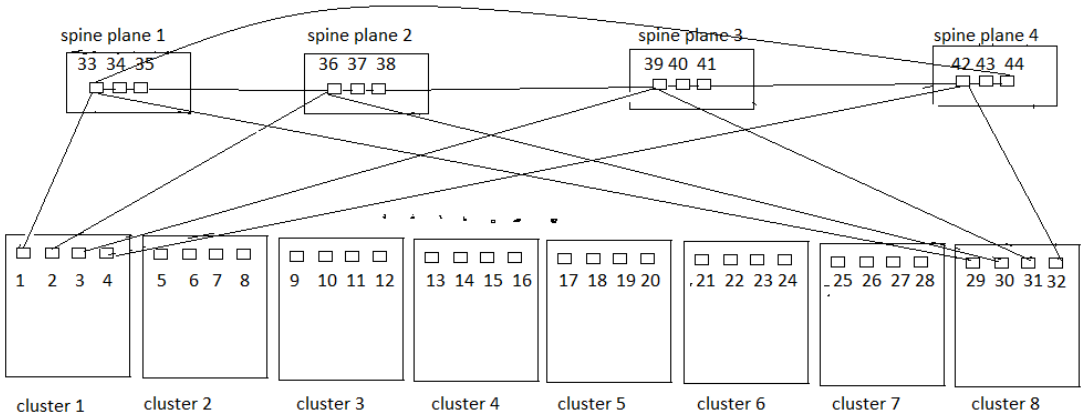

Figure 26: Case B topology with some links removed

Now in this modified Case B topology, the fabric switches instead of connecting to all the spine switches in a local spine plane, they are connected to only one spine switch in a spine plane. After removing the links, we simulate the traffic performance and it was still the same as figure 18.

The traffic matrix was then change and the traffic matrix in figure 12 (Inter-cluster Traffic Matrix between fabric switches) was used, the result is shown in figure 27. The traffic performance is worst.

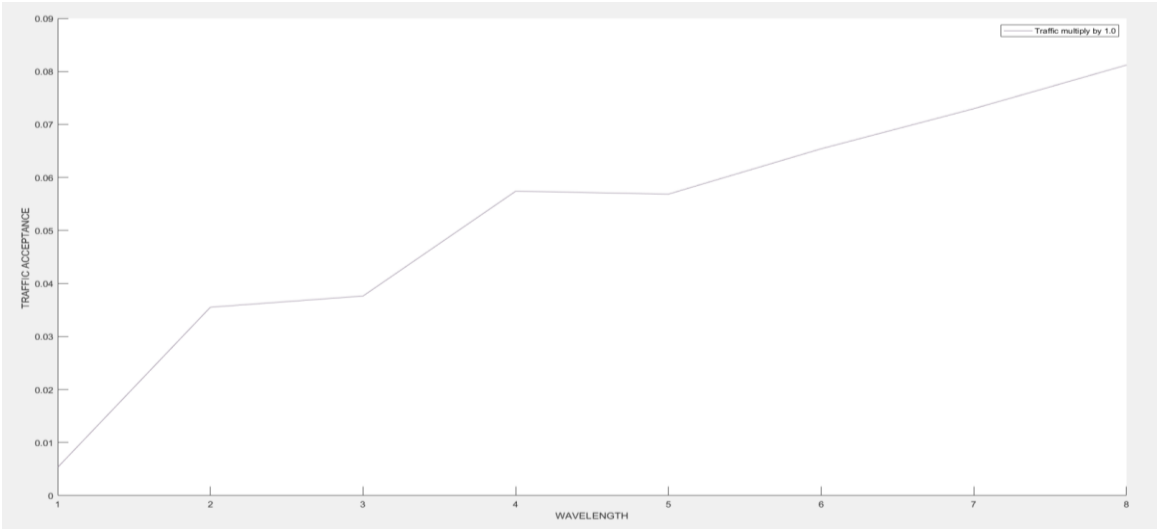

Figure 27: Traffic performance in Case B with Inter-cluster Traffic Matrix between fabric switches

In the modified Case B topology (some links removed), we simulate the traffic performance with the new traffic matrix and the result is shown in figure 28: the traffic acceptance decreased.

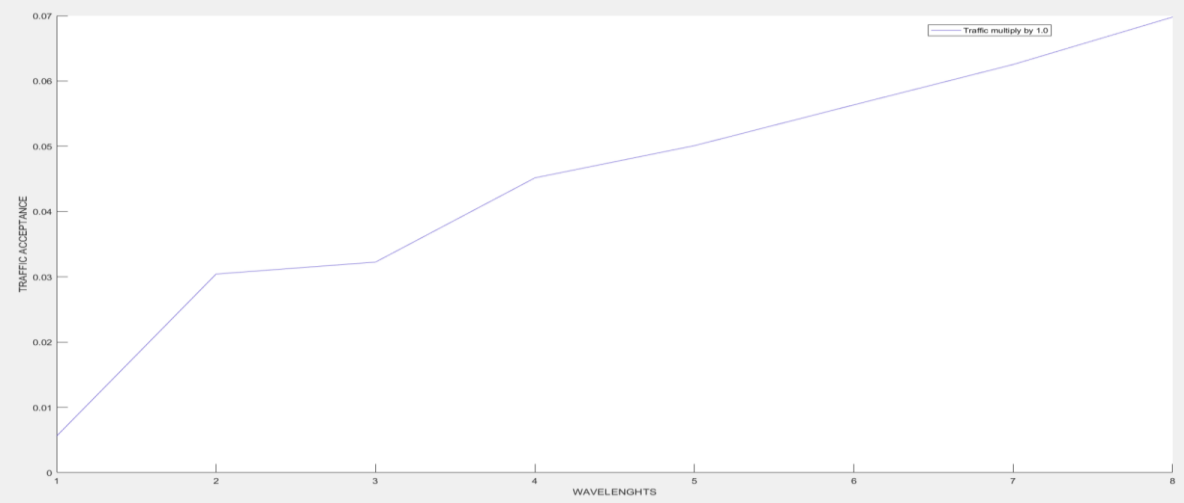

Figure 28: Traffic performance in modified Case B topology (some links removed) with Inter-cluster Traffic Matrix between fabric switches

4.4. "Case_B_Alternate.m".

The sample of how the wavelengths are assigned to the path using the fixed alternate routing and first fit technique is the same as shown in figure 21.

To be able to understand the network behaviour we calculate and simulate the traffic performance and it was the same as figure 22 because path found were the same as "Case_A_Alternate.m". Then we removed some links (modified Case B topology) and simulate again, the result is shown in figure 29. The traffic acceptance in figure 29 decrease a little in comparison with figure 22, apart from that when the matrix traffic decreases the traffic performance is still the same. That is because when the links were removed, the fixed alternate path behave as a fixed path routing keep only the principal paths and removing the alternative paths. Since the channel matrix only affects the fixed alternate routing and does not affect the fixed routing so the decrease in the traffic matrix does not have any partake in the traffic performance.

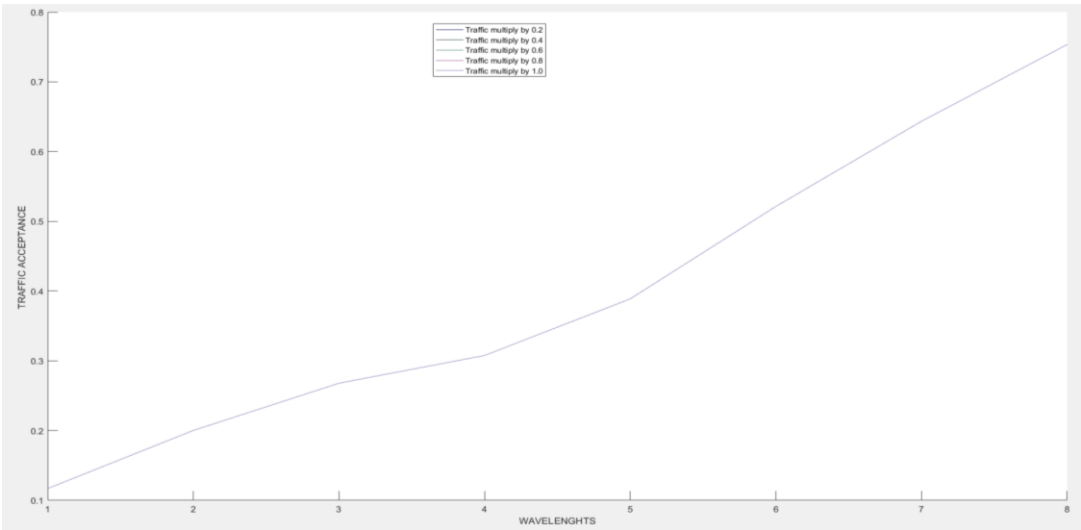

Figure 29 : Traffic performance in modified Case B topology (some links removed)

The traffic matrix was then change and the traffic matrix in figure 12 (Inter-cluster Traffic Matrix between fabric switches) was used, the result is shown in figure 30. The traffic performance is worst.

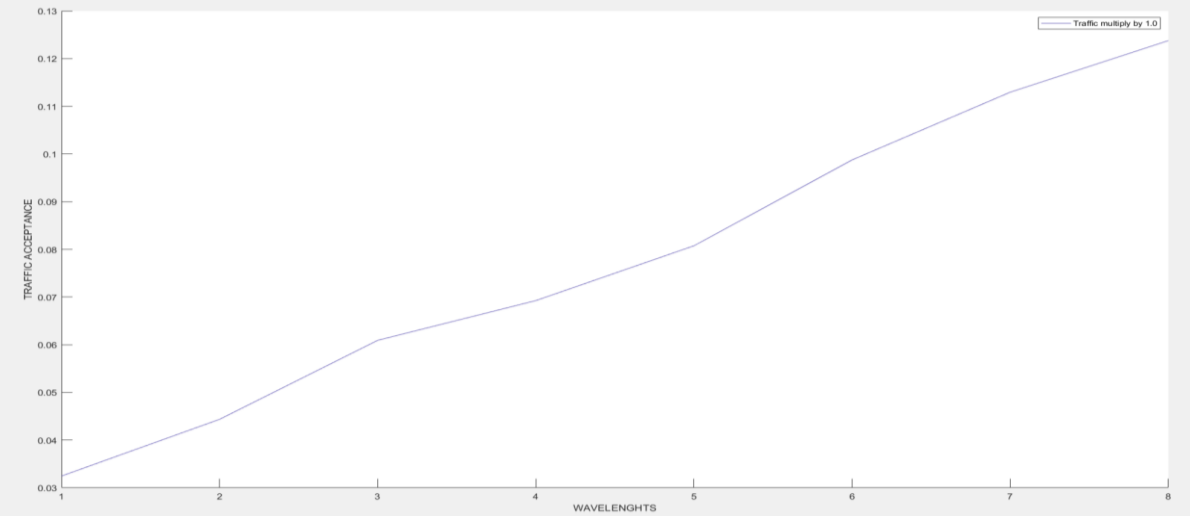

Figure 30: Traffic performance in Case B with Inter-cluster Traffic Matrix between fabric switches

In the modified Case B topology (some links removed), we simulate the traffic performance with the new traffic matrix and the result is shown in figure 31: the traffic acceptance decreased.

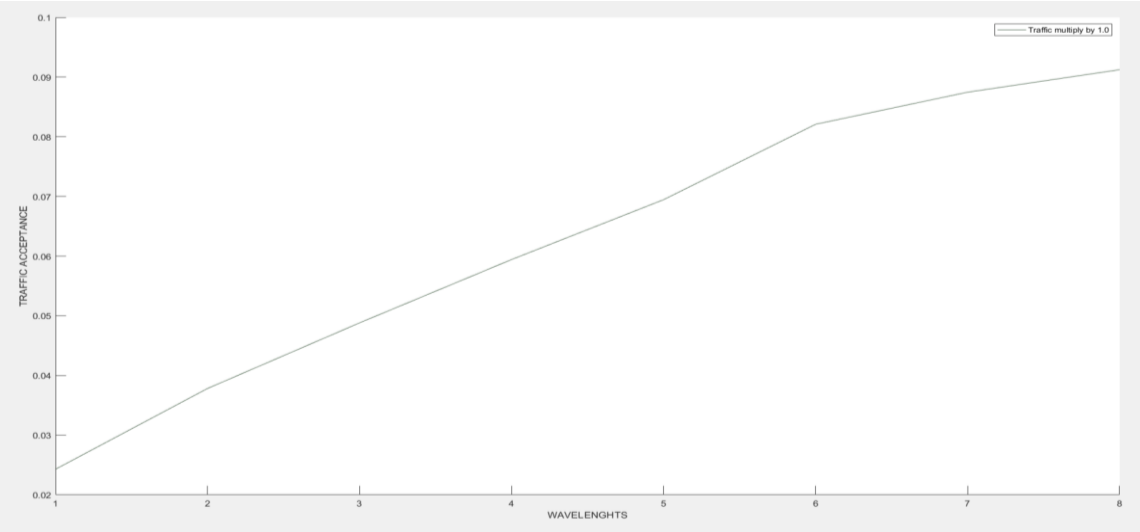

Figure 31:Traffic performance in modified Case B topology (some links removed) with Inter-cluster Traffic Matrix between fabric switches

5. Budget

The present project consists of both theoretical and practical research carried out with only one software tool which is MATLAB by an engineering student for 5 months with a total dedication of 660 hours (6h / day). Therefore, to estimate the budget, apart from the tools and equipment (PC) used, we will consider the estimated hourly price of a junior engineer.

Below is the breakdown of the project budget:

Concept	Quantity	Unit Price (€)	Cost (€)
Junior engineer	600 h	8€/h	4800 €
Laptop	1	565 €	565 €
MATLAB student license	1	35 €	35 €
Total			5400 €

Table 3: Project budget

6. Conclusions and future development:

From what we have seen till now, we have been able to see the traffic performances both in case A and in case B, and, we have been able to see that using the fixed alternate routing in the DC fabric is much better than the fixed path routing. The fixed alternate routing is better than fixed routing mainly because of the alternative paths which also happen to be the difference with the fixed path routing. Fixed-alternate routing provides simplicity of control for setting up and tearing down “light paths”, and it may also be used to provide some degree of fault tolerance upon link failures why fixed routing may be unable to handle fault situations in which one or more links in the network fail.

The connection of the spine planes through the ring topology in case B was intended to improve the topology but it failed. The traffic performances in case B is either equal to the traffic performances in case A or worst but never better than it. It is reasonable because all the paths in case A has only one hop but some of the paths in case B has more than one hop, it is known that a large number of hops between source and destination implies poor performances which makes the allocation of wavelengths difficult.

A future approach will be to interconnect the clusters through a ring topology. Since we are only interested in the intra-datacenter communications, we will use the ring topology to communicates between fabric switches and uses the spine planes only for inter-datacenter communications. We can’t guarantee that the traffic performances can be improved, this topology is much more complex than what we have been doing in this project, but it is possible.

Bibliography:

[1] Hui Zang, Jason P. Jue, Biswanath Mukherjee. “A Review of Routing and Wavelength Assignment Approaches for WavelengthRouted Optical WDM Networks”, SPIE/Baltzer Science Publishers 1388 6916/2000, January 2000.

[2] Ali Hammadi, Lofti Mhamdi. “A survey on architectures and energy efficiency in Data center Networks”, School of Electronic and Electrical Enginering, University of Leeds, United Kingdom ,12 December 2013.

[3] Krishna Kant. “Data center evolution, A tutorial on state of art, issues and challenges”,2009 published by Elsevier B.V, Intel Corporation, Hillboro, Oregon, USA.

[4] Christoforos Kachris, Ioannis Tomkos. “A Survey on Optical Interconnects for Data Centers”, IEEE communications surveys & tutorial, vol. 14, No. 4, fourth quarter 2012.

[5] Christoforos Kachris, Konstantinos Kanonakis, Ioannis Tomkos. “Optical Interconnection Networks in Data Centers: Recent Trends and Future Challenges”, IEEE Communications Magazine • September 2013

[6] Francisco Martinez de Rió, “Matlab Manuel de Usuario”, http://www4.ujaen.es/~fmartin/apuntesMATLAB.pdf, 2015.

[7] Alexey Andreyev.” Introducing data center fabric, the next-generation Facebook data center network”, https://code.fb.com/production-engineering/introducing-data-center-fabric-the-next-generation-facebook-data-center-network/, NOV 14, 2014.

Glossary

1. **DC fabric**: Data center fabric.
2. **ENIAC:** Electronic Numerical Integrator and Computer.
3. **WP ref:** Work packages reference.
4. **ToRs:** Top of Rack switches.
5. **POD:** Point of Delivery.

ANNEXES

Annex A:

"Case_A_Fixed.m" codes:

```
% Let's first "clean" the Workspace
clear all
close all
TORs=2;   %Maximum numbers of TOR switches in a POD -> [1...TORs]
FABRICs=4; %Maximum numbers of fabric switches in a POD -> [TORs+1....TORs+FABRICs]2
clusters=8; %Number of clusters in the network3
Spines_planes=4;  %Maximum numbers of spines_planes2
Spine_switches=3; %Maximum numbers of spine switches ->
[TORs+FABRICs+1...TORs+FABRICs+Spines]3
traffic_wavelength=50;   %traffic of the wavelength in Gbprs
increase_=[1:0.2:1];% Increase the traffic
max_Lambda=8;  %Number of Wavelength
count=1;
%% INTERNETWORKING OF A POD/CLUSTERS (LAYER 1 AND LAYER 2)
m=[(TORs*clusters)+1:(FABRICs+TORs)*clusters];  % vector of fabric switches
G = digraph;   % Create an empty Graph
l=1;
j=1;
for q=1:clusters
    for i=1:TORs
        a=zeros(1,FABRICs);
        a=a+j;
        s(j)={a};
        t(j)={m(l:l+FABRICs-1)};
        G = addedge(G,a,m(l:l+FABRICs-1));      % Add the conexions to the Graph
        G = addedge(G,m(l:l+FABRICs-1),a);      % Make the conexion be bidirectional
        j=j+1;
    end
    l=l+FABRICs;
end
figure(1)
p = plot(G);
p.NodeColor='black';
highlight(p,m,'NodeColor','r');
layout(p,'layered','Direction','up','Sources',[1:(TORs*clusters)]); % Representation of
the pod
matrix_link_1=full(adjacency(G));     %The physical interconexion saved in a matrix

%% INTERNETWORKING OF A CLUSTER TO A SPINE (LAYER 2 AND LAYER 3)
if (FABRICs~=Spines_planes)
    error('The number of spine planes must be the same as the number of fabric switches in
a cluster');
end
n=[((FABRICs+TORs)*clusters)+1:((FABRICs+TORs)*clusters)+(Spine_switches*Spines_planes)];
j=1;
for q=1:clusters
    l=1;
    for i=1:FABRICs
         a=zeros(1,Spine_switches);
         a=a+m(j);
         s(m(j))={a};
         test=[n(l):n(l+Spine_switches-1)];
         l=l+Spine_switches;
         t(m(j))={test};
         G = addedge(G,a,test);    % Add the conexions to the Graph
         G = addedge(G,test,a);    % Make the conexion be bidirectional
         j=j+1;
    end
end

figure(2)
p = plot(G);
p.NodeColor='black';
highlight(p,m,'NodeColor','r');
highlight(p,n,'NodeColor','b');
layout(p,'layered','Direction','up','Sources',[1:(TORs*clusters)],'Sinks',n); %
Representation of the pod
matrix_link_2=full(adjacency(G));
```

```
%%  ROUTING AND WAVELENGTH ASSIGNMENT IN THE SPINE PLANES
G_1=subgraph(G,[(TORs*clusters)+1:((FABRICs+TORs)*clusters)+(Spine_switches*Spines_planes)
]);
New_vec_fabric=[1:max(size(m))];
New_vec_spine=[max(size(m))+1:max(size(m))+(Spine_switches*Spines_planes)];
figure(3)
p = plot(G_1);
p.NodeColor='blue';
highlight(p,[1:(clusters*FABRICs)],'NodeColor','r');
layout(p,'layered','Direction','up','Sources',[1:(clusters*FABRICs)]);
matrix_link_3=full(adjacency(G_1));

% Generate a matrix traffic between Fabric switches
trafficperformance=zeros(max(size(increase_)),max_Lambda);
sum_trafficperformance=zeros(max(size(increase_)),max_Lambda);

for samples=1:1    % quantify the traffic performance
traffic=zeros(max(size(New_vec_fabric)),max(size(New_vec_fabric)));
for i=1:max(size(New_vec_fabric))
    for j=1:max(size(New_vec_fabric))
      if (i>=1)&& (i<=8) traffic_mice=ceil(random('normal',100,5)); %Normal distribution
with mean=100,std=5
      elseif (i>=9)&& (i<=16) traffic_mice=ceil(random('normal',80,5));
      elseif (i>=17)&& (i<=24) traffic_mice=ceil(random('normal',60,5));
      else  traffic_mice=ceil(random('normal',45,5));
      end
      if (i~=j)
           if (traffic(i,j)==0)
               traffic(i,j)=traffic_mice;       %traffic of a path
               traffic(j,i)=traffic_mice;
           end
       end
    end
end

% The traffic within the same cluster is null and traffic only exits were there is a
path..
for i=1:max(size(New_vec_fabric))
    for j=1:max(size(New_vec_fabric))
        ShortestPath=shortestpath(G_1,i,j);
        if (max(size(ShortestPath)))==0
        traffic(i,j)=0;
        end

    end
end
total_traffic=ceil(traffic/traffic_wavelength);

for as=1:max(size(increase_))
increase_traffic=increase_(as);
traffic_1=traffic*increase_traffic;
channels=ceil(traffic_1/traffic_wavelength);
traffic_performance=zeros(1,max_Lambda);
count=1;
for counter_lambda=1:max_Lambda
clear Path_cell;
clear traffic_path;
clear tempsave;
clear  Path2WV;
clear saveedge;
clear tempedge;

% USE techniques "FIXED PATH ROUTING" for the assignament of "routed lighthpath".
G_1.Edges.Weight = ones(1,size(G_1.Edges,1))';
Path_cell_index=1;

for i = 1:max(size(New_vec_fabric))
     for j = 1:max(size(New_vec_fabric))
             num_path=channels(i,j);
             if(num_path~=0)
                 [ShortestPath,d,edgepath]=shortestpath(G_1,i,j);
                 Path_cell(Path_cell_index) = {ShortestPath};
                 saveedge(Path_cell_index)={edgepath};
                 traffic_path(Path_cell_index)=traffic_1(i,j);
```

```
                channel_path(Path_cell_index)=channels(i,j);
                Path_cell_index=Path_cell_index+1;
            end

     end

end

%Wavelength assignment

%for the wavelength assignment, the "first fit" scheme is applied

lambda=[1:counter_lambda];

for c=1:max(size(Path_cell))
  Path2WV(c)={0};
end
i=1;
while( i<=(max(size(Path_cell))))
   break_out='FALSE';
   num_path=channel_path(i);
   o=zeros(1,num_path);
   for l=1:num_path
    if l==1
        if max(cell2mat(Path2WV(i)))==0
            o(l)=max(cell2mat(Path2WV(i)))+1;
        else
            o(l)=max(cell2mat(Path2WV(i)));
        end
    else
        if (o(l-1)+1<=counter_lambda)
           o(l)=o(l-1)+1;
        else
           clear Path2WV;
           if (isempty(Path_cell))==0
               Path_cell(i)=[];
               saveedge(i)=[];
               traffic_path(i)=[];
               channel_path(i)=[];
           end
           clear tempsave;
           clear tempedge;
           break_out='TRUE';
        end
    end
   end
   if  strcmp(break_out,'FALSE')==1
       Path2WV(i)={o};
       tempsave(i)=Path_cell(i);
       tempedge(i)=saveedge(i);
       a=cell2mat(Path_cell(i));
       b=cell2mat(saveedge(i));
       save_valor=0;

       for j = 1:max(size(tempsave))
           if strcmp(break_out,'TRUE')==1
                break;
           end
           ShortestPath=cell2mat(tempsave(j));
           ShortestPath_edge=cell2mat(tempedge(j));
           equal='FALSE';
           if ((max(size(a)))==(max(size(ShortestPath))))
               if (a==ShortestPath)
                   equal='TRUE';
               end
           end
           if strcmp(equal,'FALSE')==1
                if ((max(size(b)))==(max(size(ShortestPath_edge)))) ||
((max(size(b)))<(max(size(ShortestPath_edge))))
                    edgefound='FALSE';
                    for c=1:max(size(b))
                       for q=1:max(size(ShortestPath_edge))
                          if (b(c)==ShortestPath_edge(q))
                              edgefound='TRUE';
                              break;
```

```
                end
            end
            if strcmp(edgefound,'TRUE')==1
                break;
            end
        end
        if strcmp(edgefound,'TRUE')==1
            if  save_valor<=max(cell2mat(Path2WV(j)))
                    if ((max(cell2mat(Path2WV(j)))+1)<=counter_lambda)
                            o=zeros(1,num_path);
                            for l=1:num_path
                             if l==1
                                o(l)=max(cell2mat(Path2WV(j)))+1;
                             else
                                 o(l)=o(l-1)+1;
                             end
                            end
                    else
                        clear Path2WV;
                        Path_cell(i)=[];
                        saveedge(i)=[];
                        clear tempsave;
                        clear tempedge;
                        traffic_path(i)=[];
                        channel_path(i)=[];
                        break_out='TRUE';
                        break;
                    end
            end
        end
end
if ((max(size(b)))>(max(size(ShortestPath_edge))))
    edgefound='FALSE';
    for c=1:max(size(ShortestPath_edge))
        for q=1:max(size(b))
           if (ShortestPath_edge(c)==b(q))
               edgefound='TRUE';
               break;
           end
        end
        if strcmp(edgefound,'TRUE')==1
            break;
        end
    end
    if strcmp(edgefound,'TRUE')==1
         if  save_valor<=max(cell2mat(Path2WV(j)))
                    if ((max(cell2mat(Path2WV(j)))+1)<=counter_lambda)
                            o=zeros(1,num_path);
                            for l=1:num_path
                             if l==1
                                o(l)=max(cell2mat(Path2WV(j)))+1;
                             else
                                 o(l)=o(l-1)+1;
                             end
                            end
                    else
                        clear Path2WV;
                        Path_cell(i)=[];
                        saveedge(i)=[];
                        clear tempsave;
                        clear tempedge;
                        traffic_path(i)=[];
                        channel_path(i)=[];
                        break_out='TRUE';
                        break;
                    end
         end
    end
end
Path2WV(i)={o};
save_valor=min(cell2mat(Path2WV(i)));
if max(o)>counter_lambda
   clear Path2WV;
   Path_cell(i)=[];
   saveedge(i)=[];
```

```
                    clear tempsave;
                    clear tempedge;
                    traffic_path(i)=[];
                    channel_path(i)=[];
                    break_out='TRUE';
                    break;
                end
            end
        end
    end
    if (isempty(Path_cell))==1
       break_out='FALSE';
    end
    i=i+1;
    if strcmp(break_out,'TRUE')==1
        i=1;
        for c=1:max(size(Path_cell))
           Path2WV(c)={0};
        end
    end

end
%Calculate the "traffic performance" with the corresponding wavelength.
 traffic_performance(count)=sum(traffic_path)/sum(sum(traffic_1));
 count=count+1;
end

trafficperformance(as,:)=traffic_performance;
end
for sum_1=1:as
sum_trafficperformance(sum_1,:)=sum_trafficperformance(sum_1,:)+trafficperformance(sum_1,:
);
end
end

% Show the traffic performances in a plot
figure(4)
hold on;
for as=1:max(size(increase_))
increase_traffic=increase_(as);
spring=sprintf('Traffic multiply by %2.1f',increase_traffic);
color=rand(1,3)*0.8;
plot([1:max_Lambda],sum_trafficperformance(as,:)/1,'Displayname',spring,'Color',color);
end
ylabel('TRAFFIC ACCEPTANCE (%)');
xlabel('Wavelength');
legend('show');
hold off;

%show the max number of wavelength used in every origin node in the Graph
figure(5)
MAX_WV=zeros(1,max(size(New_vec_fabric)));
for c=1:max(size(New_vec_fabric))
    save=0;
    for i=1:max(size(Path_cell))
        a=cell2mat(Path_cell(i));
        if New_vec_fabric(c)==a(1)
           if cell2mat(Path2WV(i))>save
                save=max(cell2mat(Path2WV(i)));
           end
        end
        MAX_WV(c)=save;
    end
end
bar(New_vec_fabric,MAX_WV);
ylabel('Max wavelenght');
xlabel('Fabric Switch ID');
title('WAVELENGHT USAGE');
```

Annex B:

"Case_A_Alternate.m" codes:

```
% Let's first "clean" the Workspace
clear all
close all
TORs=2;   %Maximum numbers of TOR switches in a POD -> [1...TORs]
FABRICs=4; %Maximum numbers of fabric switches in a POD -> [TORs+1....TORs+FABRICs]2
clusters=8; %Number of clusters in the network3
Spines_planes=4;  %Maximum numbers of spines_planes2
Spine_switches=3; %Maximum numbers of spine switches ->
[TORs+FABRICs+1...TORs+FABRICs+Spines]3
traffic_wavelength=50;   %traffic of the wavelength in Gbprs
increase_=[1:0.2:1];    % Increase the traffic
max_Lambda=8;  %Number of Wavelength
count=1;
%% INTERNETWORKING OF A POD/CLUSTERS (LAYER 1 AND LAYER 2)
m=[(TORs*clusters)+1:(FABRICs+TORs)*clusters];  % vector of fabric switches
G = digraph;   % Create an empty Graph
l=1;
j=1;
for q=1:clusters
    for i=1:TORs
        a=zeros(1,FABRICs);
        a=a+j;
        s(j)={a};
        t(j)={m(l:l+FABRICs-1)};
        G = addedge(G,a,m(l:l+FABRICs-1));      % Add the conexions to the Graph
        G = addedge(G,m(l:l+FABRICs-1),a);      % Make the conexion be bidirectional
        j=j+1;
    end
    l=l+FABRICs;
end
figure(1)
p = plot(G);
p.NodeColor='black';
highlight(p,m,'NodeColor','r');
layout(p,'layered','Direction','up','Sources',[1:(TORs*clusters)]); % Representation of
the pod
matrix_link_1=full(adjacency(G));     %The physical interconexion saved in a matrix


%% INTERNETWORKING OF A CLUSTER TO A SPINE (LAYER 2 AND LAYER 3)
if (FABRICs~=Spines_planes)
    error('The number of spine planes must be the same as the number of fabric switches in
a cluster');
end
n=[((FABRICs+TORs)*clusters)+1:((FABRICs+TORs)*clusters)+(Spine_switches*Spines_planes)];
j=1;
for q=1:clusters
    l=1;
    for i=1:FABRICs
         a=zeros(1,Spine_switches);
         a=a+m(j);
         s(m(j))={a};
         test=[n(l):n(l+Spine_switches-1)];
         l=l+Spine_switches;
         t(m(j))={test};
         G = addedge(G,a,test);    % Add the conexions to the Graph
         G = addedge(G,test,a);    % Make the conexion be bidirectional
         j=j+1;
    end
end

figure(2)
p = plot(G);
p.NodeColor='black';
highlight(p,m,'NodeColor','r');
highlight(p,n,'NodeColor','b');
layout(p,'layered','Direction','up','Sources',[1:(TORs*clusters)],'Sinks',n); %
Representation of the pod
matrix_link_2=full(adjacency(G));
```

```
%%  ROUTING AND WAVELENGTH ASSIGNMENT IN THE SPINE PLANES
G_1=subgraph(G,[(TORs*clusters)+1:((FABRICs+TORs)*clusters)+(Spine_switches*Spines_planes)
]);
New_vec_fabric=[1:max(size(m))];
New_vec_spine=[max(size(m))+1:max(size(m))+(Spine_switches*Spines_planes)];
figure(3)
p = plot(G_1);
p.NodeColor='blue';
highlight(p,[1:(clusters*FABRICs)],'NodeColor','r');
layout(p,'layered','Direction','up','Sources',[1:(clusters*FABRICs)]);
matrix_link_3=full(adjacency(G_1));

% Generate a matrix traffic between Fabric switches
trafficperformance=zeros(max(size(increase_)),max_Lambda);
sum_trafficperformance=zeros(max(size(increase_)),max_Lambda);

for samples=1:1    % quantify the traffic
traffic=zeros(max(size(New_vec_fabric)),max(size(New_vec_fabric)));
for i=1:max(size(New_vec_fabric))
    for j=1:max(size(New_vec_fabric))
       if (i>=1)&& (i<=8) traffic_mice=ceil(random('normal',100,5));
       elseif (i>=9)&& (i<=16) traffic_mice=ceil(random('normal',80,5));
       elseif (i>=17)&& (i<=24) traffic_mice=ceil(random('normal',60,5));
       else  traffic_mice=ceil(random('normal',45,5));
       end
      if (i~=j)
           if (traffic(i,j)==0)
               traffic(i,j)=traffic_mice;       %traffic of a path
               traffic(j,i)=traffic_mice;
           end
       end
    end
end

% the traffic within the same cluster is null and traffic only exits were there is a
path..
for i=1:max(size(New_vec_fabric))
    for j=1:max(size(New_vec_fabric))
        ShortestPath=shortestpath(G_1,i,j);
        if (max(size(ShortestPath)))==0
        traffic(i,j)=0;
        end

    end
end

total_traffic=ceil(traffic/traffic_wavelength);

for as=1:max(size(increase_))
increase_traffic=increase_(as);
traffic_1=traffic*increase_traffic;
channels=ceil(traffic_1/traffic_wavelength);
traffic_performance=zeros(1,max_Lambda);
count=1;

for counter_lambda=11:max_Lambda
clear Path_cell;
clear traffic_path;
clear tempsave;
clear  Path2WV;

% USE techniques "FIXED ALTERNATE ROUTING" for the assignament of "routed lighthpath".
G_1.Edges.Weight = ones(1,size(G_1.Edges,1))';
Path_cell_index=1;
for i = 1:max(size(New_vec_fabric))
     for j = 1:max(size(New_vec_fabric))
             num_path=channels(i,j);
             q=1;
             while(q<=num_path)
                 [ShortestPath,d,edgepath]=shortestpath(G_1,i,j);
                 for l=1:(max(size(ShortestPath)))-1
                   idx=findedge(G_1,ShortestPath(l),ShortestPath(l+1));
                   G_1.Edges.Weight(idx)= G_1.Edges.Weight(idx)+1;
                 end
                 boolean='FALSE';
```

```
                if (max(size(ShortestPath)))~=0
                    for s=1:Path_cell_index-1
                      if ShortestPath==cell2mat(Path_cell(s))
                            boolean='TRUE';
                      end
                    end
                    if  strcmp(boolean,'FALSE')==1
                        Path_cell(Path_cell_index) = {ShortestPath};
                        traffic_path(Path_cell_index)=traffic_1(i,j);
                        saveedge(Path_cell_index)={edgepath};
                        Path_cell_index=Path_cell_index+1;

                    end
                end
                q=q+1;
            end


    end
end

%Wavelength assignment
%for the wavelength assignment, the "first fit" scheme is applied

lambda=[1:counter_lambda];
total_traffic_path=traffic_path;
i=1;
while( i<=(max(size(Path_cell))))
   break_out='FALSE';
   Path2WV(i)=min(lambda);
   tempsave(i)=Path_cell(i);
   a=cell2mat(Path_cell(i));
   save_valor=0;
   q=0;
   for j = 1:max(size(tempsave))
       if strcmp(break_out,'TRUE')==1
            break;
       end
       ShortestPath=cell2mat(tempsave(j));
       if (a==ShortestPath)
      else
         for l=1:(max(size(ShortestPath)))-1
                 if  ([a(l) a(l+1)]==[ShortestPath(l) ShortestPath(l+1)])
                       if  save_valor<=Path2WV(j)
                           if ((Path2WV(j)+1)<=counter_lambda)
                               Path2WV(i)=Path2WV(j)+1;
                               save_valor=Path2WV(i);
                           else
                              clear Path2WV;
                              Path_cell(i)=[];
                              clear tempsave;
                              traffic_path(i)=[];
                              break_out='TRUE';
                              break;
                           end
                       end
                end
        end
       end
   end
   i=i+1;
   if strcmp(break_out,'TRUE')==1
       i=1;
   end
end

 %traffic_performance(count)=((max(size(Path_cell))))/(Path_cell_index-1);
 traffic_performance(count)=sum(traffic_path)/sum(total_traffic_path);
 count=count+1;
end
for i = 1:max(size(Path_cell))
        color=rand(1,3)*0.8;
        highlight(p,cell2mat(Path_cell(i)),'EdgeColor',color);
end
trafficperformance(as,:)=traffic_performance;
```

```
end
for sum_1=1:as
sum_trafficperformance(sum_1,:)=sum_trafficperformance(sum_1,:)+trafficperformance(sum_1,:
);
end
end

% Show the traffic performances in a plot

figure(4)
hold on;
for as=1:max(size(increase_))
increase_traffic=increase_(as);
spring=sprintf('Traffic multiply by %2.1f',increase_traffic);
color=rand(1,3)*0.8;
plot([1:max_Lambda],sum_trafficperformance(as,:)/1,'Displayname',spring,'Color',color);
end
ylabel('TRAFFIC ACCEPTANCE (%)');
xlabel('LAMBDA');
legend('show');
hold off;

%show the max number of wavelengh used in every origin node in the Graph
figure(5)
MAX_WV=zeros(1,max(size(New_vec_fabric)));
for c=1:max(size(New_vec_fabric))
    save=0;
    for i=1:max(size(Path_cell))
        a=cell2mat(Path_cell(i));
        if New_vec_fabric(c)==a(1)
           if Path2WV(i)>save
                save=Path2WV(i);
           end
        end
        MAX_WV(c)=save;
    end
end
bar(New_vec_fabric,MAX_WV);
ylabel('Max wavelenght');
xlabel('Fabric Switch ID');
title('WAVELENGHT USAGE');
```

Annex C:

"Case_B_Fixed.m" codes:

```
% Let's first "clean" the Workspace
clear all
close all
TORs=2;   %Maximum numbers of TOR switches in a POD -> [1...TORs]
FABRICs=4; %Maximum numbers of fabric switches in a POD -> [TORs+1....TORs+FABRICs]
clusters=8; %Number of clusters in the network
Spines_planes=4;  %Maximum numbers of spines_planes2
Spine_switches=3; %Maximum numbers of spine switches ->
[TORs+FABRICs+1...TORs+FABRICs+Spines]
traffic_wavelength=50;   %bandwith of the wavelength in Gbprs
increase_=[1:0.2:1];    % Increase the traffic
max_Lambda=8;  %Number of Wavelength
count=1;
%% INTERNETWORKING OF A POD/CLUSTERS (LAYER 1 AND LAYER 2)
m=[(TORs*clusters)+1:(FABRICs+TORs)*clusters];  % vector of fabric switches
G = digraph;   % Create an empty Graph
l=1;
j=1;
for q=1:clusters
    for i=1:TORs
        a=zeros(1,FABRICs);
        a=a+j;
        s(j)={a};
        t(j)={m(l:l+FABRICs-1)};
        G = addedge(G,a,m(l:l+FABRICs-1));      % Add the conexions to the Graph
        G = addedge(G,m(l:l+FABRICs-1),a);      % Make the conexion be bidirectional
        j=j+1;
    end
    l=l+FABRICs;
end
figure(1)
p = plot(G);
p.NodeColor='black';
highlight(p,m,'NodeColor','r');
layout(p,'layered','Direction','up','Sources',[1:(TORs*clusters)]); % Representation of
the pod
matrix_link_1=full(adjacency(G));     %The physical interconexion saved in a matrix

%% INTERNETWORKING OF A CLUSTER TO A SPINE (LAYER 2 AND LAYER 3)
if (FABRICs~=Spines_planes)
    error('The number of spine planes must be the same as the number of fabric switches in
a cluster');
end
n=[((FABRICs+TORs)*clusters)+1:((FABRICs+TORs)*clusters)+(Spine_switches*Spines_planes)];
j=1;
for q=1:clusters
    l=1;
    for i=1:FABRICs
         a=zeros(1,Spine_switches);
         a=a+m(j);
         s(m(j))={a};
         test=[n(l):n(l+Spine_switches-1)];
         l=l+Spine_switches;
         t(m(j))={test};
         G = addedge(G,a,test);    % Add the conexions to the Graph
         G = addedge(G,test,a);    % Make the conexion be bidirectional
         j=j+1;
    end
end
G_intra=G;
% Connect the spine planes in a ring topology
for q=1:max(size(n))-1
   G = addedge(G,n(q),n(q+1));
   G = addedge(G,n(q+1),n(q));
end
G = addedge(G,n(max(size(n))),n(1));
G = addedge(G,n(1),n(max(size(n))));
figure(2)
p = plot(G);
p.NodeColor='black';
```

```
highlight(p,m,'NodeColor','r');
highlight(p,n,'NodeColor','b');
layout(p,'layered','Direction','up','Sources',[1:(TORs*clusters)],'Sinks',n); %
Representation of the pod
matrix_link_2=full(adjacency(G));

%%  ROUTING AND WAVELENGTH ASSIGNMENT IN THE SPINE PLANES
G_1=subgraph(G,[(TORs*clusters)+1:((FABRICs+TORs)*clusters)+(Spine_switches*Spines_planes)
]);
G_1_intra=subgraph(G_intra,[(TORs*clusters)+1:((FABRICs+TORs)*clusters)+(Spine_switches*Sp
ines_planes)]);

New_vec_fabric=[1:max(size(m))];
New_vec_spine=[max(size(m))+1:max(size(m))+(Spine_switches*Spines_planes)];

%Remove edges (links)to see the network behaviour
for i=1:max(size(New_vec_fabric))
    a=nearest(G_1,i,1); % find the neighbours nodes to node i with distance 1.
    l=1;
    if max(size(a))>=2
     while(l<max(size(a)))
         l=l+1;
         G_1=rmedge(G_1,i,a(l));
         G_1=rmedge(G_1,a(l),i);
     end
    end
end
figure(3)
p = plot(G_1);
p.NodeColor='blue';
highlight(p,[1:(clusters*FABRICs)],'NodeColor','r');
layout(p,'layered','Direction','up','Sources',[1:(clusters*FABRICs)],'Sinks',New_vec_spine
);
matrix_link_3=full(adjacency(G_1));

trafficperformance=zeros(max(size(increase_)),max_Lambda);
sum_trafficperformance=zeros(max(size(increase_)),max_Lambda);

for samples=1:1    % quantify the traffic performance
traffic=zeros(max(size(New_vec_fabric)),max(size(New_vec_fabric)));
for i=1:max(size(New_vec_fabric))
    for j=1:max(size(New_vec_fabric))
      if (i>=1)&& (i<=8) traffic_mice=ceil(random('normal',100,5)); %Normal distribution
with mean=100,std=5
      elseif (i>=9)&& (i<=16) traffic_mice=ceil(random('normal',80,5));
      elseif (i>=17)&& (i<=24) traffic_mice=ceil(random('normal',60,5));
      else  traffic_mice=ceil(random('normal',45,5));
      end
      if (i~=j)
           if (traffic(i,j)==0)
               traffic(i,j)=traffic_mice;       %traffic of a path
               traffic(j,i)=traffic_mice;
           end
       end
    end
end
%The traffic within the same cluster is permited and traffic only exits were there is a
path..
% this traffic matrix is equal to the traffic matrix in case A.
for i=1:max(size(New_vec_fabric))
    for j=1:max(size(New_vec_fabric))
        ShortestPath=shortestpath(G_1_intra,i,j);
        if (max(size(ShortestPath)))==0
        traffic(i,j)=0;
        end

    end
end
% The new traffic matrix
% for i=1:max(size(New_vec_fabric))
%     for j=1:max(size(New_vec_fabric))
%         ShortestPath=shortestpath(G_1,i,j);
%         if (max(size(ShortestPath)))==0
%         traffic(i,j)=0;
%         end
```

```
%
%     end
% end
% i=1;
% l=1;
% while(i<=max(size(New_vec_fabric))/FABRICs)
%
%     a=New_vec_fabric(l:l+FABRICs-1);
%     cluster_save(i,:)=a;
%    l=l+FABRICs;
% i=i+1;
% end
%
% for i=1:max(size(New_vec_fabric))
%        [e r]=find(cluster_save==i);
%     for j=1:max(size(New_vec_fabric))
%         if isempty(find(cluster_save(e,:)==j))
%         else
%            traffic(i,j)=0;
%            traffic(j,i)=0;
%         end
%
%     end
% end
%
for as=1:max(size(increase_))
increase_traffic=increase_(as);
traffic_1=traffic*increase_traffic;
channels=ceil(traffic_1/traffic_wavelength);
traffic_performance=zeros(1,max_Lambda);
count=1;

for counter_lambda=1:max_Lambda
clear Path_cell;
clear traffic_path;
clear tempsave;
clear  Path2WV;
clear saveedge;
clear tempedge;

% USE techniques "FIXED PATH ROUTING" for the assignament of "routed lighthpath".
G_1.Edges.Weight = ones(1,size(G_1.Edges,1))';
Path_cell_index=1;

for i = 1:max(size(New_vec_fabric))
     for j = 1:max(size(New_vec_fabric))
             num_path=channels(i,j);
             if(num_path~=0)
                 [ShortestPath,d,edgepath]=shortestpath(G_1,i,j);
                 Path_cell(Path_cell_index) = {ShortestPath};
                 saveedge(Path_cell_index)={edgepath};
                 traffic_path(Path_cell_index)=traffic_1(i,j);
                 channel_path(Path_cell_index)=channels(i,j);
                 Path_cell_index=Path_cell_index+1;
             end

     end

end

%Wavelength assignment
%for the wavelength assignment, the "first fit" scheme is applied
lambda=[1:counter_lambda];

for c=1:max(size(Path_cell))
  Path2WV(c)={0};
end
i=1;

while( i<=(max(size(Path_cell))))
   break_out='FALSE';

   num_path=channel_path(i);
   o=zeros(1,num_path);
   for l=1:num_path
```

```
    if l==1
        if max(cell2mat(Path2WV(i)))==0
            o(l)=max(cell2mat(Path2WV(i)))+1;
        else
            o(l)=max(cell2mat(Path2WV(i)));
        end
    else
        if (o(l-1)+1<=counter_lambda)
           o(l)=o(l-1)+1;
        else
           clear Path2WV;
           if (isempty(Path_cell))==0
               Path_cell(i)=[];
               saveedge(i)=[];
               traffic_path(i)=[];
               channel_path(i)=[];
           end
           clear tempsave;
           clear tempedge;
           break_out='TRUE';
        end
    end
   end
   if  strcmp(break_out,'FALSE')==1
       Path2WV(i)={o};
       tempsave(i)=Path_cell(i);
       tempedge(i)=saveedge(i);
       a=cell2mat(Path_cell(i));
       b=cell2mat(saveedge(i));
       save_valor=0;

       for j = 1:max(size(tempsave))
           if strcmp(break_out,'TRUE')==1
                break;
           end
           ShortestPath=cell2mat(tempsave(j));
           ShortestPath_edge=cell2mat(tempedge(j));
           equal='FALSE';
           if ((max(size(a)))==(max(size(ShortestPath))))
               if (a==ShortestPath)
                   equal='TRUE';
               end
           end
           if strcmp(equal,'FALSE')==1
                if ((max(size(b)))==(max(size(ShortestPath_edge)))) ||
((max(size(b)))<(max(size(ShortestPath_edge))))
                    edgefound='FALSE';
                    for c=1:max(size(b))
                       for q=1:max(size(ShortestPath_edge))
                          if (b(c)==ShortestPath_edge(q))
                              edgefound='TRUE';
                              break;
                          end
                       end
                       if strcmp(edgefound,'TRUE')==1
                           break;
                       end
                    end
                    if strcmp(edgefound,'TRUE')==1
                        if  save_valor<=max(cell2mat(Path2WV(j)))
                                    if ((max(cell2mat(Path2WV(j)))+1)<=counter_lambda)
                                           o=zeros(1,num_path);
                                           for l=1:num_path
                                            if l==1
                                               o(l)=max(cell2mat(Path2WV(j)))+1;
                                            else
                                                o(l)=o(l-1)+1;
                                            end
                                           end
                                    else
                                       clear Path2WV;
                                       Path_cell(i)=[];
                                       saveedge(i)=[];
                                       clear tempsave;
                                       clear tempedge;
```

```
                                    traffic_path(i)=[];
                                    channel_path(i)=[];
                                    break_out='TRUE';
                                    break;
                                end
                            end
                        end
                    end
                    if ((max(size(b)))>(max(size(ShortestPath_edge))))
                        edgefound='FALSE';
                        for c=1:max(size(ShortestPath_edge))
                            for q=1:max(size(b))
                               if (ShortestPath_edge(c)==b(q))
                                   edgefound='TRUE';
                                   break;
                               end
                            end
                            if strcmp(edgefound,'TRUE')==1
                                break;
                            end
                        end
                        if strcmp(edgefound,'TRUE')==1
                             if  save_valor<=max(cell2mat(Path2WV(j)))
                                        if ((max(cell2mat(Path2WV(j)))+1)<=counter_lambda)
                                               o=zeros(1,num_path);
                                               for l=1:num_path
                                                if l==1
                                                   o(l)=max(cell2mat(Path2WV(j)))+1;
                                                else
                                                    o(l)=o(l-1)+1;
                                                end
                                               end
                                        else
                                           clear Path2WV;
                                           Path_cell(i)=[];
                                           saveedge(i)=[];
                                           clear tempsave;
                                           clear tempedge;
                                           traffic_path(i)=[];
                                           channel_path(i)=[];
                                           break_out='TRUE';
                                           break;
                                        end
                                end
                        end
                    end
                    Path2WV(i)={o};
                    save_valor=min(cell2mat(Path2WV(i)));
                    if max(o)>counter_lambda
                       clear Path2WV;
                       Path_cell(i)=[];
                       saveedge(i)=[];
                       clear tempsave;
                       clear tempedge;
                       traffic_path(i)=[];
                       channel_path(i)=[];
                       break_out='TRUE';
                       break;
                    end
                end
            end
        end
    end
    if (isempty(Path_cell))==1
       break_out='FALSE';
    end
    i=i+1;
    if strcmp(break_out,'TRUE')==1
        i=1;
        for c=1:max(size(Path_cell))
           Path2WV(c)={0};
        end
    end

end
%Calculate the "traffic performance" with the corresponding wavelength.
```

```
 traffic_performance(count)=sum(traffic_path)/sum(sum(traffic_1));
 count=count+1;

end
trafficperformance(as,:)=traffic_performance;
end
for sum_1=1:as
sum_trafficperformance(sum_1,:)=sum_trafficperformance(sum_1,:)+trafficperformance(sum_1,:
);
end
end


% Show the traffic performances in a plot
figure(4)
hold on;
for as=1:max(size(increase_))
increase_traffic=increase_(as);
spring=sprintf('Traffic multiply by %2.1f',increase_traffic);
color=rand(1,3)*0.8;
plot([1:max_Lambda],sum_trafficperformance(as,:)/1,'Displayname',spring,'Color',color);
end
ylabel('TRAFFIC ACCEPTANCE (%)');
xlabel('LAMBDA');
legend('show');
hold off;
```

Annex D:

"Case_B_Alternate.m" codes:

```
% Let's first "clean" the Workspace
clear all
close all
TORs=2;   %Maximum numbers of TOR switches in a POD -> [1...TORs]
FABRICs=4; %Maximum numbers of fabric switches in a POD -> [TORs+1....TORs+FABRICs]
clusters=8; %Number of clusters in the network
Spines_planes=4;  %Maximum numbers of spines_planes2
Spine_switches=3; %Maximum numbers of spine switches ->
[TORs+FABRICs+1...TORs+FABRICs+Spines]
traffic_wavelength=50;   %bandwith of the wavelength in Gbprs
increase_=[1:0.2:1];    % Increase the traffic
max_Lambda=12;  %Number of Wavelength
count=1;
%% INTERNETWORKING OF A POD/CLUSTERS (LAYER 1 AND LAYER 2)
m=[(TORs*clusters)+1:(FABRICs+TORs)*clusters];  % vector of fabric switches
G = digraph;   % Create an empty Graph
l=1;
j=1;
for q=1:clusters
    for i=1:TORs
        a=zeros(1,FABRICs);
        a=a+j;
        s(j)={a};
        t(j)={m(l:l+FABRICs-1)};
        G = addedge(G,a,m(l:l+FABRICs-1));      % Add the conexions to the Graph
        G = addedge(G,m(l:l+FABRICs-1),a);      % Make the conexion be bidirectional
        j=j+1;
    end
    l=l+FABRICs;
end
figure(1)
p = plot(G);
p.NodeColor='black';
highlight(p,m,'NodeColor','r');
layout(p,'layered','Direction','up','Sources',[1:(TORs*clusters)]); % Representation of
the pod
matrix_link_1=full(adjacency(G));     %The physical interconexion saved in a matrix

%% INTERNETWORKING OF A CLUSTER TO A SPINE (LAYER 2 AND LAYER 3)
if (FABRICs~=Spines_planes)
    error('The number of spine planes must be the same as the number of fabric switches in
a cluster');
end
n=[((FABRICs+TORs)*clusters)+1:((FABRICs+TORs)*clusters)+(Spine_switches*Spines_planes)];
j=1;
for q=1:clusters
    l=1;
    for i=1:FABRICs
         a=zeros(1,Spine_switches);
         a=a+m(j);
         s(m(j))={a};
         test=[n(l):n(l+Spine_switches-1)];
         l=l+Spine_switches;
         t(m(j))={test};
         G = addedge(G,a,test);    % Add the conexions to the Graph
         G = addedge(G,test,a);    % Make the conexion be bidirectional
         j=j+1;
    end
end
G_intra=G;
% Connect the spine planes in a ring topology
for q=1:max(size(n))-1
   G = addedge(G,n(q),n(q+1));
   G = addedge(G,n(q+1),n(q));
end
G = addedge(G,n(max(size(n))),n(1));
G = addedge(G,n(1),n(max(size(n))));
figure(2)
```

```
p = plot(G);
p.NodeColor='black';
highlight(p,m,'NodeColor','r');
highlight(p,n,'NodeColor','b');
layout(p,'layered','Direction','up','Sources',[1:(TORs*clusters)],'Sinks',n); %
Representation of the pod
matrix_link_2=full(adjacency(G));

%%  ROUTING AND WAVELENGTH ASSIGNMENT IN THE SPINE PLANES
G_1=subgraph(G,[(TORs*clusters)+1:((FABRICs+TORs)*clusters)+(Spine_switches*Spines_planes)
]);
G_1_intra=subgraph(G_intra,[(TORs*clusters)+1:((FABRICs+TORs)*clusters)+(Spine_switches*Sp
ines_planes)]);

New_vec_fabric=[1:max(size(m))];
New_vec_spine=[max(size(m))+1:max(size(m))+(Spine_switches*Spines_planes)];

%Remove edges (links)to see the network behaviour
for i=1:max(size(New_vec_fabric))
    a=nearest(G_1,i,1); % find the neighbours nodes to node i with distance 1.
    l=1;
    if max(size(a))>=2
     while(l<max(size(a)))
         l=l+1;
         G_1=rmedge(G_1,i,a(l));
         G_1=rmedge(G_1,a(l),i);
     end
    end
end

figure(3)
p = plot(G_1);
p.NodeColor='blue';
highlight(p,[1:(clusters*FABRICs)],'NodeColor','r');
layout(p,'layered','Direction','up','Sources',[1:(clusters*FABRICs)],'Sinks',New_vec_spine
);

matrix_link_3=full(adjacency(G_1));
trafficperformance=zeros(max(size(increase_)),max_Lambda);
sum_trafficperformance=zeros(max(size(increase_)),max_Lambda);

for samples=1:1    % quantify the traffic performance
traffic=zeros(max(size(New_vec_fabric)),max(size(New_vec_fabric)));

for i=1:max(size(New_vec_fabric))
    for j=1:max(size(New_vec_fabric))
      if (i>=1)&& (i<=8) traffic_mice=ceil(random('normal',100,5)); %Normal distribution
with mean=100,std=5
      elseif (i>=9)&& (i<=16) traffic_mice=ceil(random('normal',80,5));
      elseif (i>=17)&& (i<=24) traffic_mice=ceil(random('normal',60,5));
      else  traffic_mice=ceil(random('normal',45,5));
      end
      if (i~=j)
           if (traffic(i,j)==0)
               traffic(i,j)=traffic_mice;       %traffic of a path
               traffic(j,i)=traffic_mice;
           end
       end
    end
end
%The traffic within the same cluster is permited and traffic only exits were there is a
path..
% this traffic matrix is equal to the traffic matrix in case A.
for i=1:max(size(New_vec_fabric))
    for j=1:max(size(New_vec_fabric))
        ShortestPath=shortestpath(G_1_intra,i,j);
        if (max(size(ShortestPath)))==0
        traffic(i,j)=0;
        end

    end
end
% for i=1:max(size(New_vec_fabric))
%     for j=1:max(size(New_vec_fabric))
%         ShortestPath=shortestpath(G_1,i,j);
```

```
%         if (max(size(ShortestPath)))==0
%         traffic(i,j)=0;
%         end
%
%     end
% end
% i=1;
% l=1;
% while(i<=max(size(New_vec_fabric))/FABRICs)
%
%     a=New_vec_fabric(l:l+FABRICs-1);
%     cluster_save(i,:)=a;
%    l=l+FABRICs;
% i=i+1;
% end
%
% for i=1:max(size(New_vec_fabric))
%        [e r]=find(cluster_save==i);
%     for j=1:max(size(New_vec_fabric))
%         if isempty(find(cluster_save(e,:)==j))
%         else
%            traffic(i,j)=0;
%            traffic(j,i)=0;
%         end
%
%     end
% end

for as=1:max(size(increase_))
increase_traffic=increase_(as);
traffic_1=traffic*increase_traffic;
channels=ceil(traffic_1/traffic_wavelength);
traffic_performance=zeros(1,max_Lambda);
count=1;

for counter_lambda=1:max_Lambda
clear Path_cell;
clear traffic_path;
clear tempsave;
clear  Path2WV;
clear saveedge;
clear tempedge;


G_1.Edges.Weight = ones(1,size(G_1.Edges,1))';
Path_cell_index=1;
for i = 1:max(size(New_vec_fabric))
     for j = 1:max(size(New_vec_fabric))
             num_path=channels(i,j);
             q=1;
             while(q<=num_path)
                 [ShortestPath,d,edgepath]=shortestpath(G_1,i,j);
                 for l=1:(max(size(edgepath)))
                     G_1.Edges.Weight(edgepath(l))= G_1.Edges.Weight(edgepath(l))+1;
                 end
                 boolean='FALSE';
                 if (max(size(ShortestPath)))~=0
                     for s=1:Path_cell_index-1
                         if (max(size(ShortestPath)))==(max(size(cell2mat(Path_cell(s)))))
                           if ShortestPath==cell2mat(Path_cell(s))
                                 boolean='TRUE';
                           end
                         end
                     end
                     if  strcmp(boolean,'FALSE')==1
                         Path_cell(Path_cell_index) = {ShortestPath};
                         traffic_path(Path_cell_index)=traffic_1(i,j);
                         saveedge(Path_cell_index)={edgepath};
                         Path_cell_index=Path_cell_index+1;
                         channel_path(Path_cell_index)=channels(i,j);
                     end
                 end
                 q=q+1;
             end
```

```
     end
end

%Wavelength assignment
%for the wavelength assignment, the "first fit" scheme is applied

total_traffic_path=traffic_path;
lambda=[1:counter_lambda];

for c=1:max(size(Path_cell))
  Path2WV(c)={0};
end
i=1;
while( i<=(max(size(Path_cell))))
   break_out='FALSE';


   o=zeros(1,1);
   for l=1:1
    if l==1
        if max(cell2mat(Path2WV(i)))==0
            o(l)=max(cell2mat(Path2WV(i)))+1;
        else
            o(l)=max(cell2mat(Path2WV(i)));
        end
    else
        if (o(l-1)+1<=counter_lambda)
           o(l)=o(l-1)+1;
        else
           clear Path2WV;
           if (isempty(Path_cell))==0
               Path_cell(i)=[];
               saveedge(i)=[];
               traffic_path(i)=[];
               channel_path(i)=[];
           end
           clear tempsave;
           clear tempedge;
           break_out='TRUE';
        end
    end
   end
   if  strcmp(break_out,'FALSE')==1
       Path2WV(i)={o};
       tempsave(i)=Path_cell(i);
       tempedge(i)=saveedge(i);
       a=cell2mat(Path_cell(i));
       b=cell2mat(saveedge(i));
       save_valor=0;

       for j = 1:max(size(tempsave))
           if strcmp(break_out,'TRUE')==1
                break;
           end
           ShortestPath=cell2mat(tempsave(j));
           ShortestPath_edge=cell2mat(tempedge(j));
           equal='FALSE';
           if ((max(size(a)))==(max(size(ShortestPath))))
               if (a==ShortestPath)
                   equal='TRUE';
               end
           end
           if strcmp(equal,'FALSE')==1
                if ((max(size(b)))==(max(size(ShortestPath_edge)))) ||
((max(size(b)))<(max(size(ShortestPath_edge))))
                    edgefound='FALSE';
                    for c=1:max(size(b))
                       for q=1:max(size(ShortestPath_edge))
                          if (b(c)==ShortestPath_edge(q))
                              edgefound='TRUE';
                              break;
                          end
                       end
                       if strcmp(edgefound,'TRUE')==1
```

```
                break;
            end
        end
        if strcmp(edgefound,'TRUE')==1
            if  save_valor<=max(cell2mat(Path2WV(j)))
                    if ((max(cell2mat(Path2WV(j)))+1)<=counter_lambda)
                            o=zeros(1,1);
                            for l=1:1
                             if l==1
                                o(l)=max(cell2mat(Path2WV(j)))+1;
                             else
                                 o(l)=o(l-1)+1;
                             end
                            end
                    else
                       clear Path2WV;
                       Path_cell(i)=[];
                       saveedge(i)=[];
                       clear tempsave;
                       clear tempedge;
                       traffic_path(i)=[];
                       channel_path(i)=[];
                       break_out='TRUE';
                       break;
                    end
            end
        end
end
if ((max(size(b)))>(max(size(ShortestPath_edge))))
    edgefound='FALSE';
    for c=1:max(size(ShortestPath_edge))
        for q=1:max(size(b))
           if (ShortestPath_edge(c)==b(q))
               edgefound='TRUE';
               break;
           end
        end
        if strcmp(edgefound,'TRUE')==1
            break;
        end
    end
    if strcmp(edgefound,'TRUE')==1
         if  save_valor<=max(cell2mat(Path2WV(j)))
                    if ((max(cell2mat(Path2WV(j)))+1)<=counter_lambda)
                            o=zeros(1,1);
                            for l=1:1
                             if l==1
                                o(l)=max(cell2mat(Path2WV(j)))+1;
                             else
                                 o(l)=o(l-1)+1;
                             end
                            end
                    else
                       clear Path2WV;
                       Path_cell(i)=[];
                       saveedge(i)=[];
                       clear tempsave;
                       clear tempedge;
                       traffic_path(i)=[];
                       channel_path(i)=[];
                       break_out='TRUE';
                       break;
                    end
            end
    end
end
Path2WV(i)={o};
save_valor=min(cell2mat(Path2WV(i)));
if max(o)>counter_lambda
   clear Path2WV;
   Path_cell(i)=[];
   saveedge(i)=[];
   clear tempsave;
   clear tempedge;
   traffic_path(i)=[];
```

```
                    channel_path(i)=[];
                    break_out='TRUE';
                    break;
                end
            end
        end
    end
    if (isempty(Path_cell))==1
       break_out='FALSE';
    end
    i=i+1;
    if strcmp(break_out,'TRUE')==1
        i=1;
        for c=1:max(size(Path_cell))
           Path2WV(c)={0};
        end
    end

end
%Calculate the "traffic performance" with the corresponding wavelength.

 traffic_performance(count)=sum(traffic_path)/sum(total_traffic_path);
 count=count+1;

end
trafficperformance(as,:)=traffic_performance;
end
for sum_1=1:as
sum_trafficperformance(sum_1,:)=sum_trafficperformance(sum_1,:)+trafficperformance(sum_1,:
);
end
end

% Show the traffic performances in a plot
figure(4)
hold on;
for as=1:max(size(increase_))
increase_traffic=increase_(as);
spring=sprintf('Traffic multiply by %2.1f',increase_traffic);
color=rand(1,3)*0.8;
plot([1:max_Lambda],sum_trafficperformance(as,:)/1,'Displayname',spring,'Color',color);
end
ylabel('TRAFFIC ACCEPTANCE (%)');
xlabel('LAMBDA');
legend('show');
hold off;
```