Report:

Open and Virtualized Networks

Hamza Rehman

s266061

April 9, 2021

Introduction

The course Open and Virtualized Networks (OVN) uses Object-Oriented Programming paradigm to manage networks from the physical layer up to the network layer. For this purpose, network elements and subsystems are virtualized, relying on physical and operational mathematical/statistical models. The course proposes the fundamental concepts of the optical network elements (transceivers, fiber propagation, amplifiers and switches) with the purpose to enable the use of the APIs virtualizing their operational modes.

This document mainly deals with the description of laboratory exercise from 8 till 11, with elaborative and concise description of all the working nuts and bolts of the concerned laboratory and it's outcome. This document is divided into different sections, each dealing with one laboratory and carried forward into subsequent laboratory.

Recap

The network that is constructed in Laboratory 3 is being carried till laboratory 11 with modifications in each subsequent laboratory and addition of new methods. The exercise mainly deals with software abstraction of a network, whose information, like node, position and the node it is connected to is given in .json file. And using it we defined **Node** class and **Line** class. Node class attributes contains name, position, list of connected nodes, and dictionary named successive. Line class attributes contains line label, length, by using them we were able to construct the lines of our virtual network. For making the connection less prone to loss of information we used amplifier after every 80 KM. All the possible paths present in the network can be evaluated by using method *find_paths()*. We have a method in Network class named *set_weighted_paths()*, that generates a dataframe that contains all the

possible paths in network which are obtained as a consequence of using *find_paths()* method along with their latencies[1] and SNR's[2]. This dataframe is being stored in one of the attribute of Network class named *route_space()*. Once the network is established, we can obtain the graphical representation of the network using *draw()* method. This graphical representation of network was displayed using **matplotlib**, Python library.

We have two methods that help us to choose the best path for transmission of a signal from source to destination node, first *find_best_latency()* , while selecting the path for transmission we express our interest that whether we want best path in terms of lowest latency or of highest SNR, this method is for choosing the path with lowest latency while the method *find_best_snr()* is for selecting the path with highest SNR. If the priority of latency or SNR is not expressed, then the software will select a path that has highest SNR by default.

Node, Line and Network class, all have a method named *propagate()*, but each with different functionality. When the signal needs to propagate, at first, propagate method of **Network** class is called which extract path from **Lightpath** object that is passed as an argument to propagate method. Then it calls the propagate method in **Node** class of the first node present in the path which is the source node. This recursion continues until we get to the destination node. All these activities were performed using single channel and moving onto the next laboratories we were advised to use multiple channels for signal propagation and in case of using multiple channels, first we had to check if that particular channel is free or not. Till now, we were mainly propagating the signal using the above mentioned classes

1. Total time delay due to signal propagation through any network element along the path.

2. Signal-to-Noise Ratio

and methods but in order to obtain a fruitful connection in any given two nodes we define a class named **Connection** that given any two nodes i.e. input node and output node, it stores the signal power, latency and SNR of signal. After building the connection from input and output node we use a method *stream()* which is defined in **Network** class, which takes an argument the best which can be latency or SNR and a list of connections. We optimize the connection by using method *optimization()*, that predicts the signal power to be used for lossless communication. The attribute amplifier is introduced which expresses number of optical amplifiers. Moreover, the gain and noise figure are fixed for each amplifier to 20 dB and 5 dB respectively. ASE[3] is generated using a cascade of amplifiers which introduces noise figure. Now as we have an amplifier at every 80 KM that increase the gain of signal, it also increases the noise gain along with the signal and it is nonlinear noise. The method *nli_generation()* is used to evaluate this noise.

3. Amplified Spontaneous Emission

Laboratory 8

This laboratory gives more accurate depiction of noise propagation through an optical fibre line. In order to calculate more precise abstraction of noise, first we have to calculate GSNR[4] propagating through a Lightpath[5] with optimal signal power. A new concept that we encountered in this laboratory was the concept of transceiver.[6] In this laboratory we used three different transceivers technologies for our laboratory exercise namely,

- Fixed-rate Transceiver
- Flex-rate Transceiver
- Shannon Transceiver

Fixed-rate transceiver is the simplest one, it's transmitter is capable of transmitting with one modulation format. This technology cannot be efficient for exploiting the capabilities of the network. With Fixed-rate transceiver we maybe able to transmit only 100Gbps per channel per lightpath. PM-QPSK modulation is an example of fixed-rate transceiver.

Flex-rate transceiver or flexible rate transceiver uses more complex modulation format and transmit more data per channel. The range can vary from 0 to 400Gbps depending upon GSNR. And for all these different transmitting rates we have different transceivers, PM-QPSK (100Gbps), PM-8-QAM (200Gbps) and PM-16QAM (400Gbps) modulations.

Shannon transceiver does not exist and practically it cannot be implemented, it is more of an ideal case for transmitting data over the channel using ideal Gaussian modulation format.

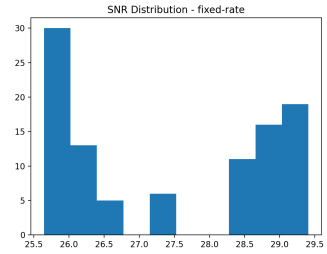
---

4. Gradient Signal-to-Noise Ratio

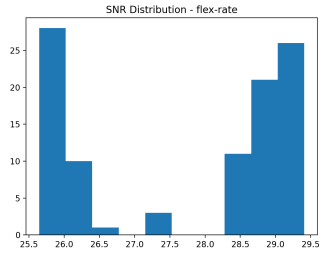
5. a transparent route or path over the network at a given wavelength

6. a device that is able to both transmit and receive information through a transmission medium

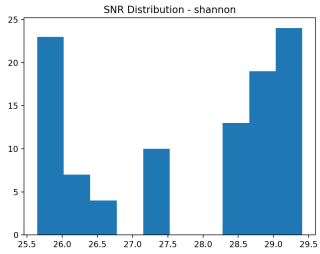
While implementing the mathematical formula for calculating fixed-rate or flex-rate or Shannon we have to calculate the bitrate supported by the lightpath deploying the connection, based on the lightpath GSNR.



(a) SNR fixed-rate

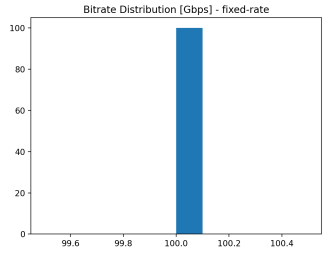


(b) SNR flex-rate

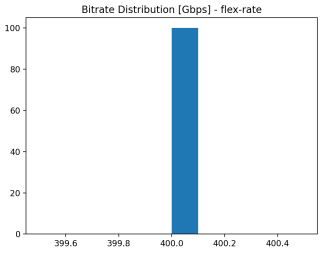


(c) SNR Shannon

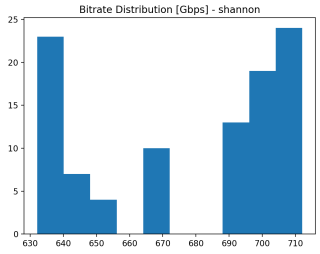
Figure 1: SNR distribution of transceivers



(a) Bitrate fixed-rate

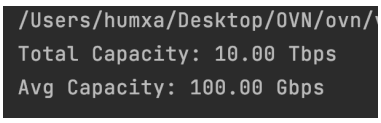


(b) Bitrate flex-rate

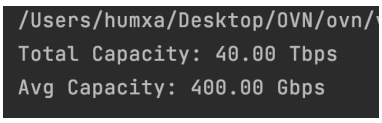


(c) Bitrate Shannon

Figure 2: Bitrate distribution of transceivers



(a) Capacity fixed-rate



(b) Capacity flex-rate

(c) Capacity Shannon

Figure 3: Capacities of transceivers

SNR distribution vary predominantly depending upon the shuffle(node_labels) for each transceiver but bit of similarity of distribution lies between flex-rate and Shannon in the region 28 dB to 30 dB. Whereas the behavior between 25 dB to 26 dB is almost the same for each transceiver. As can be seen in figure 3(a), 3(b) and 3(c) the total capacities

are 10 Tbps, 40 Tbps and 67.42 Tbps for fixed-rate, flex-rate and Shannon respectively and their average capacities are also calculated using the piece-wise formulas given in laboratory 8 document. The average capacities are 100 Gbps for fixed-rate, 400 Gbps for flex-rate and 674.22 Gbps for Shannon. As can be seen the total and average of capacities of Shannon is higher than both the fixed-rate and flex-rate is due to the fact that Shannon is an ideal case of a transceiver and practical implementation of such a transceiver is not possible due to many reasons.

Laboratory 9

Laboratory 9 introduces us to a new concept of traffic matrix. A traffic matrix T is defined as a matrix with a row and a column for each node of the network. Each element $T_{i,j}$ represents the bitrate request in Gbps between the nodes i, j. If $T_{i,j} = 0$ then no connection request is issued between i, j. If $T_{i,j} = \text{Inf}$, then it is intended as pure connection request where whatever available bitrate from the lightpath is accepted. The traffic matrix usually contains one request per node-pair. That is the case when a network's snapshot under full load has to be captured and analyzed to determine its power consumption and other performance parameters. Also we are requesting a specific bitrate for a connection request rather than accepting available rate. For this we mark the connection request as "blocked" if the requested rate is not available or cannot be satisfied. In order to do this we modify the lightpath allocation method, so, that if a path is found but the available bitrate is less than the requested rate, it searches for other lightpath so that the sum of their capacity is at least equal to the requested. If the requested rate is reached one or more lightpaths

are allocated for that connection request, otherwise it is marked as blocked. In the end all these bitrates are stored in *bitrate* attribute of the class **Connection**. And similarly the method *calculate_capacity()* is modified to sum up all these sums from previous laboratory. When the requested lightpath has been assigned we update the routing space so as to make it compatible with our changes in the program. For updating the routing space we checked all lightpaths and when we are sure that connection can be allocated, because we have to find out the lightpaths to fulfil residual rate request, then in the end we update the routing space otherwise not.

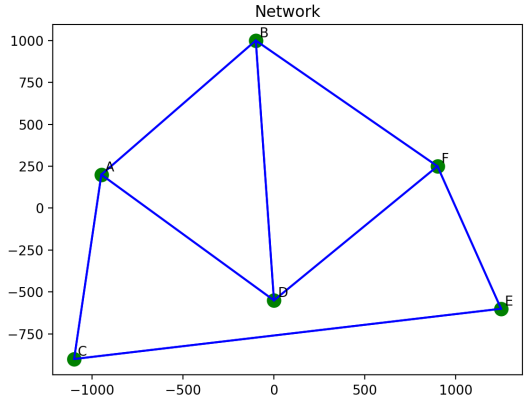
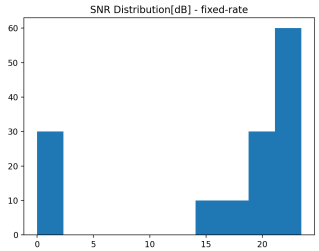
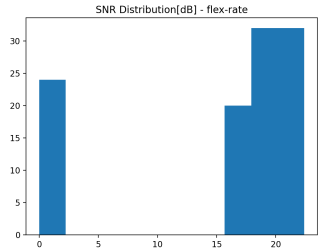


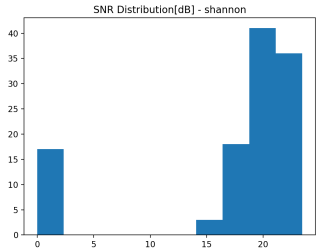
Figure 4: Network in consideration



(a) SNR fixed-rate

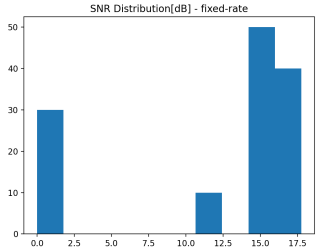


(b) SNR flex-rate

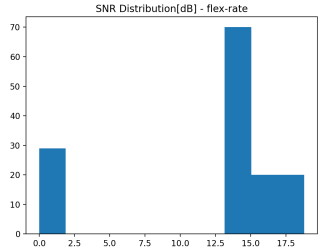


(c) SNR Shannon

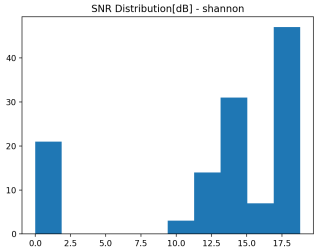
Figure 5: SNR distribution with noise figure 7



(a) SNR fixed-rate

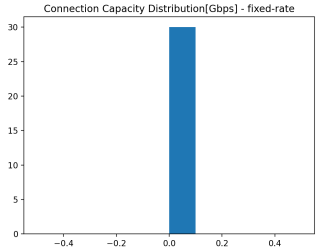


(b) SNR flex-rate

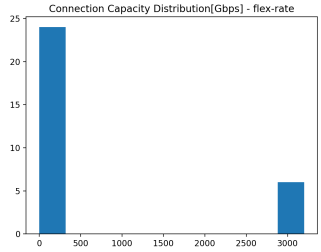


(c) SNR Shannon

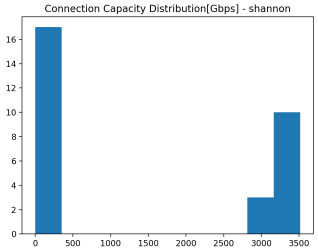
Figure 6: SNR distribution with noise figure 14



(a) Connection Capacity fixed-rate

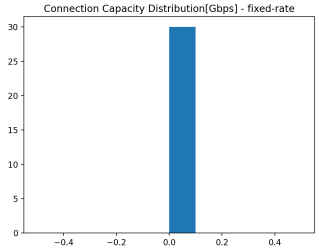


(b) Connection Capacity flex-rate

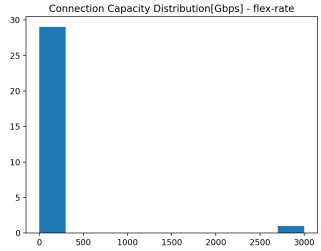


(c) Connection Capacity Shannon

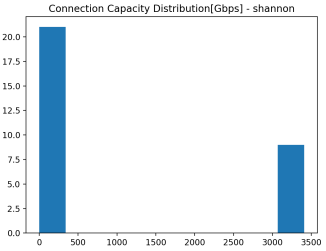
Figure 7: Connection capacity distribution with noise figure 7



(a) Connection Capacity fixed-rate

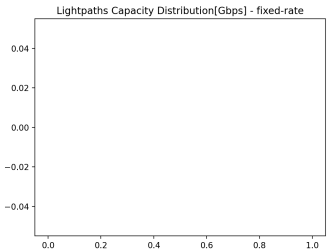


(b) Connection Capacity flex-rate

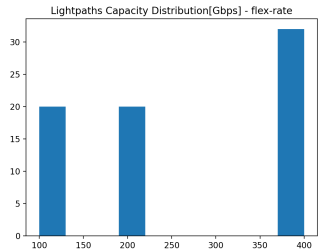


(c) Connection Capacity Shannon

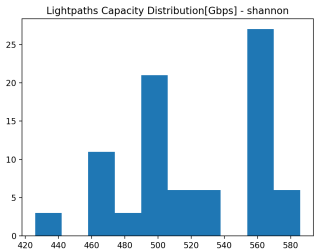
Figure 8: Connection capacity distribution with noise figure 14



(a) Lightpaths Capacity fixed-rate

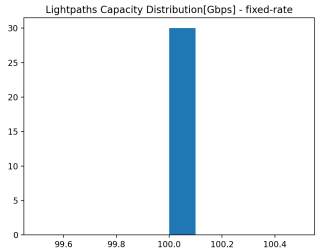


(b) Lightpaths Capacity flex-rate

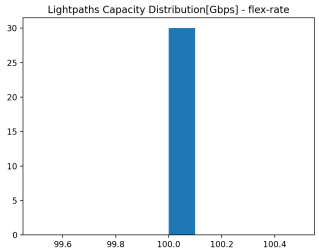


(c) Lightpaths Capacity Shannon

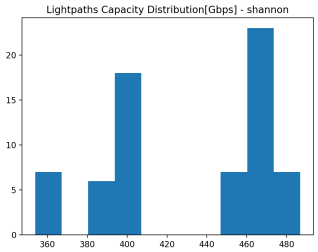
Figure 9: Lightpaths capacity distribution with noise figure 7



(a) Lightpaths Capacity fixed-rate

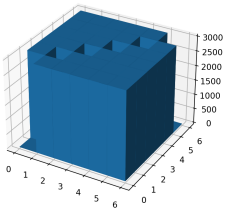


(b) Lightpaths Capacity flex-rate

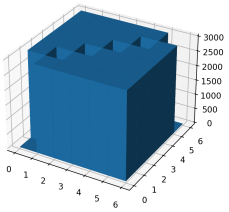


(c) Lightpaths Capacity Shannon

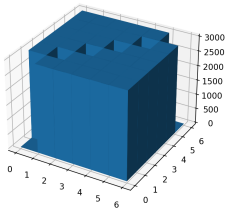
Figure 10: Lightpaths capacity distribution with noise figure 14



(a) Traffic matrix fixed-rate

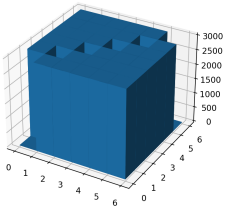


(b) Traffic matrix flex-rate

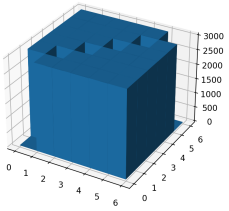


(c) Traffic matrix Shannon

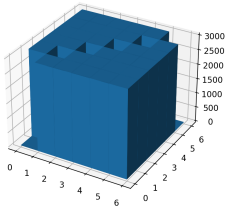
Figure 11: 3D Traffic matrix with noise figure 7



(a) Traffic matrix fixed-rate

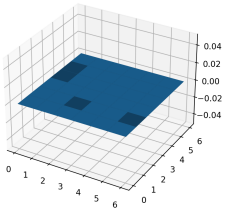


(b) Traffic matrix flex-rate

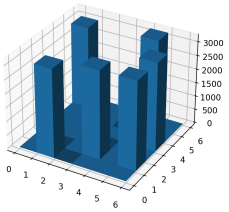


(c) Traffic matrix Shannon

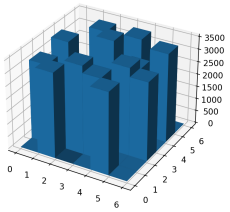
Figure 12: 3D Traffic matrix with noise figure 14



(a) Traffic matrix fixed-rate

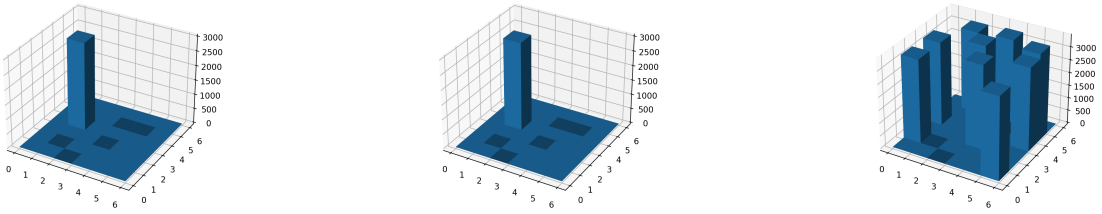


(b) Traffic matrix flex-rate



(c) Traffic matrix Shannon

Figure 13: 3D Traffic matrix with noise figure 7



(a) Traffic matrix fixed-rate (b) Traffic matrix flex-rate (c) Traffic matrix Shannon

Figure 14: 3D Traffic matrix with noise figure 14

Depicting from the graphs we can clearly see that the graphs with a noise figure of 7, as we increased it to 14 various distortions occurs. All parameters of the transceivers average SNR, total capacity of Connections, total capacity of Lightpaths and average capacity of the network tend to decrease, and in case of fixed-rate transceivers the parameters almost tend to zero apart from average SNR, when we change the noise figure from 7 to 14. This increase in noise figure also effect on traffic matrix and the values for connection request are all almost 0 which means no connection has been established. The 3D figures for traffic matrix are shown above along with the ones in which the noise figures have been increased.

Laboratory 10

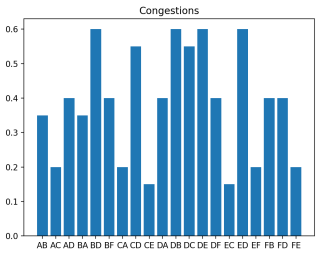
This laboratory simulates network performance for our optical network. For simulation purpose we use the acclaimed "Monte-Carlo Simulation" technique. Monte Carlo simulation, is a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results.[7] The underlying concept is to use randomness to solve problems that might be deterministic in principle. In other words Monte-Carlo simulation is a method where we simulate a lot of times a phenomenon where something is randomly

7. definition from Wikipedia

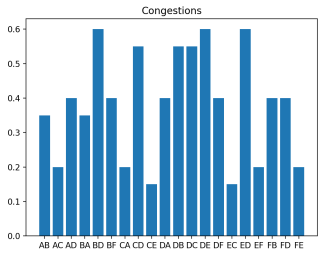
happening. In this laboratory more precisely we derived the average bitrate of network by performing Monte-Carlo simulation on lightpath allocation. The rate that we obtain from our network depends upon the sequence of requested lightpath allocation within the network. It should be kept in mind that the sequence which we ask for allocation lightpath request, taken from the traffic matrix is important for final allocation of the lightpath distribution. Subsequently the number of channels are increased from 10 to 20. In our program the Monte Carlo simulation is performed on traffic matrix with 100 realizations [8] and for this whole simulation "Shannon" transceiver technology is being used. The allocated sequences for each realization may generate different performances and results, we have to go on with our simulation until we collect sufficient amount of observations and at the end we average the results of all these realizations of our Monte- Carlo method in order to get an overview of the performance of our network. Monte-Carlo simulation is performed on sequence of the connection requested not exactly on the traffic matrix, we keep the traffic matrix uniform in the beginning. After performing Monte-Carlo simulation on our network, we have gathered sufficient amount of data on which we can calculate the average congestion of the network lines. The congestion of a line is defined as the percentage of allocated channel to the total available channel. Analyzing the congestion plots we look for the most congested line and we decrease the noise figure of amplifier by 3 dB for that line only and redo the Monte-Carlo simulation on the upgraded network and check for improvements in congestion and overall network capacity.

---

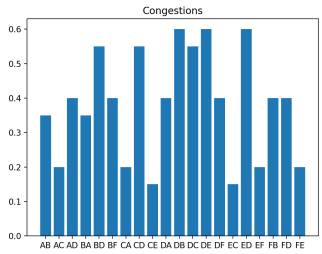
8. a method of finding explicit networks (configurations and element values) from prescribed characteristics



(a) 10 Monte-Carlo Simulations

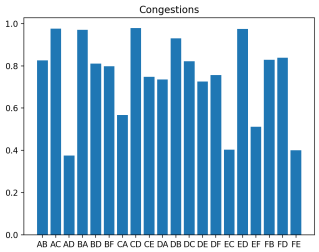


(b) 20 Monte-Carlo Simulations

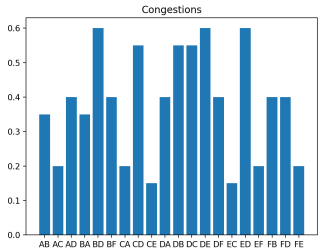


(c) 30 Monte-Carlo Simulations

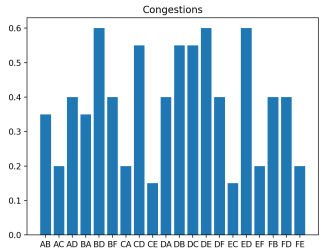
Figure 15: Line Congestions



(a) 50 Monte-Carlo Simulations



(b) 100 Monte-Carlo Simulations



(c) 500 Monte-Carlo Simulations

Figure 16: Line Congestions

Simulations	Line upgrade	Avg.Traffic	Lightpath Bitrate	Lightpath SNR
10	BD	72.92	639.64	26.00
20	BD	72.26	639.51	25.99
30	DB	72.26	639.51	25.99
50	BD	72.26	639.51	25.99
100	DB	72.26	639.51	25.99
500	BD	72.26	639.51	25.99

Table 1: Monte-Carlo Simulation Analytics

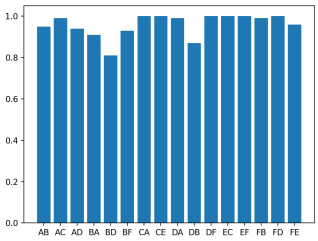
As directed in lab manual the number of channels were set to 20 and different number of Monte-Carlo simulation were carried out, every time the simulation ended the line was upgraded and a new simulation was carried out. The number of simulations that I performed were 10, 20, 30, 100 and 500. All congestion graphs are shown above. The lines that were mainly used to be upgraded to were BD and DB, it was sort of flipping in between these two.

The average total traffic starting at 72.92 Tbps for 10 simulations and increasing further on the number of simulations, it stuck at 72.26 Tbps and remained constant. As, for average Lightpath bitrate it started off at 639.94 Gbps and kept constant at 639.51 Gbps for the rest of the simulations and for average Lightpath SNR it started at 26.00 dB and remained constant at 25.99 dB for the rest of the simulations.

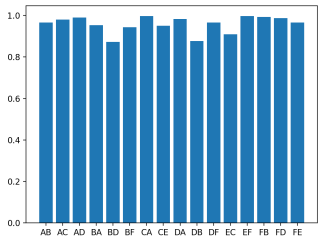
Laboratory 11

In previous laboratory we used 20 channels in total and 100 realizations but it took a lot of time for simulation to complete and yield the final outcome, but in this laboratory we have decreased the number of channels as well as the number of realizations to 10 and 50 respectively. Almost 50% decrease in the value of both attributes. It not only helped in the processing time but also we were able to obtain more observations in analyzing the network. As we increase the **rate_multiplier** the performance of the whole network degrades, the average rate per lightpath increases, as well as the congestion increases. If we increase the number of channels it helps with congestion increases. The **upgrade_method** decreases the noise and we notice a slight improvement in performance of the network. If we degrade the noise figure by 3 dB it assumes an increase in noise but theoretically it should worsen the network performance but as it is completely random simulation and every time we have different sequences of lightpath allocated, the results may vary. If we use **beta** value of LEAF the performance of the network improves by a wide margin.

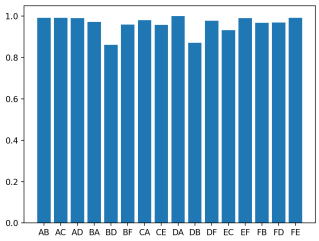
The performance of the whole network degraded as the rate_multiplier was increased, the performance of the network can be analyzed by the following data obtained as number



(a) 10 Monte-Carlo Simulations



(b) 30 Monte-Carlo Simulations



(c) 50 Monte-Carlo Simulations

Figure 17: Line Congestions

Simulations	Line upgrade	Avg.Traffic	Lightpath Bitrate	Lightpath SNR
10	CA	37.98	498.15	19.20
30	CA	39.62	499.58	19.28
50	DF	37.98	498.28	19.24

Table 2: Monte-Carlo Simulation Analytics

of Monte-Carlo simulations were increased.

Bibliography

[1] Prof. Vittorio Curri, and Prof. Cristina Rottondi. *Lecture Slides*. DET, Politecnico di Torino.

[2] Science Direct,
`https://www.sciencedirect.com/`

[3] Wikipedia,
`https://en.wikipedia.org/`

[4] IEEE Xplore,
`https://ieeexplore.ieee.org/`