Performance Evaluation and Application
Project report

Giovanni Demasi, 10656704

A.Y. 2022/23

1 Introduction

The purpose of this document is to explain how the final project of the course "Performance Evaluation and Application" has been faced, which techniques have been used and why. The document will briefly explain the project and then the solution will be explained and analysed in detail.

2 Project description

The project is about a three layer cloud architecture, composed by three layers of switches:

1. Access, 112 MB/s
2. Aggregation, 280 MB/s
3. Core, 1.12 GB/s

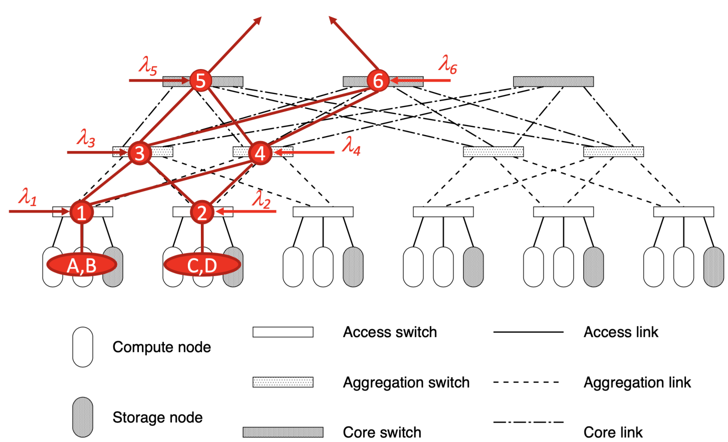


Figure 1: Datacenter

Services of switches can be considered exponentially distributed and data can be routed among different parallel redundant paths.

Two nodes per layer, and four type of traffic have to be considered:

1. A, 1 - 3 - 5
2. B, 1 - 4 - 5
3. C, 2 - 4 - 6
4. D, 2 - 3 - 6

Four .txt files are given. Data traces in files TraceD-A.txt to TraceD-D.txt shows the time instant when a 1MB block of data is received for each of the traffic types, expressed in milliseconds from the start of the logging.

Nodes are characterized by a finite capacity of 16 MB (at all the levels) and drops. Nodes are also characterized by background traffic, $\lambda 1$ to $\lambda 6$:

1. $\lambda 1$ = $\lambda 2$ = 40MB/s
2. $\lambda 3$ = $\lambda 4$ = 180MB/s
3. $\lambda 5$ = $\lambda 6$ = 600MB/s

Background traffic is routed to the two upstream nodes with equal probability and it can be considered a Poisson process.

The system is experiencing too many losses, for this reason the manager would like to add an extra link, with the same characteristics as the existing nodes. The aim of the project is to find which type of link (access, aggregation or core) should be added to reduce losses and how traffic of the four classes should be re-routed in order to take advantage of the new node.

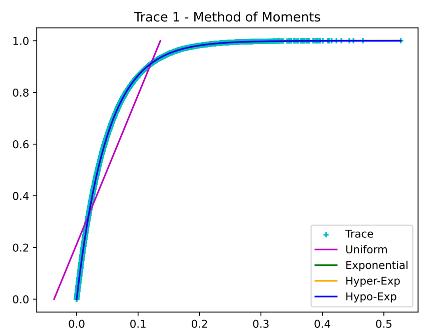
3 Project solution

3.1 Traces fitting

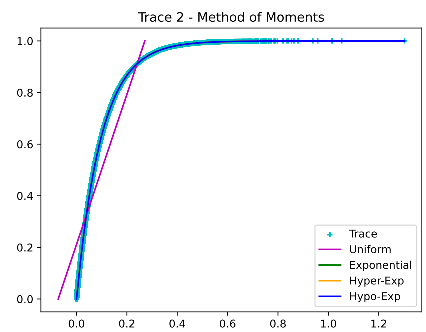
The project asks to consider four different traces which represents the time instant of when a 1MB block of data is received. One file for each traffic types is available and the arrival time instants are expressed in milliseconds. Since it is preferred not to use the repeater of JSIM, these traces needs to be fitted to a distribution in order to be used.

This has been done using a Python script. The first thing that has been done is the conversion of the arrival times from milliseconds to seconds, this for coherence with all the other parameters and to have JSIM results in seconds too. Then the inter-arrival times have been computed doing the subtraction between adjacent arrival times. It has been assumed that no arrival is present at time instant zero, this means that the number of inter-arrival times is equal to the number of arrival times minus one.

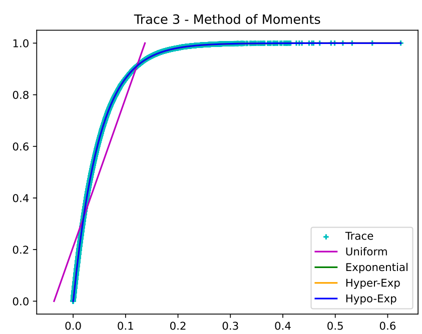
The successive step is the fitting of the distribution. A new array has been created, containing the sorted inter-arrival times which has been used to build the approximate cumulative distribution function (CDF) of the dataset. Then the fitting has been done considering the main distributions seen during the course (uniform, exponential, hyper-exponential and hypo-exponential). The fitting result, performed using the method of moments, is shown in the four plots below.



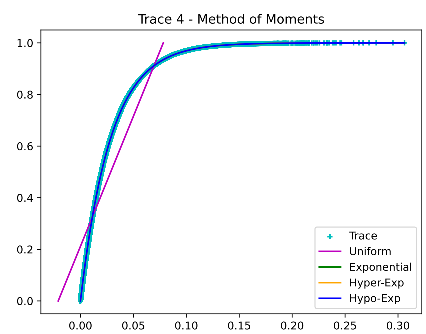
(a) TraceA.txt fitting



(b) TraceB.txt fitting



(c) TraceC.txt fitting



(d) TraceD.txt fitting

Figure 2: Plots of traces distributions fitting

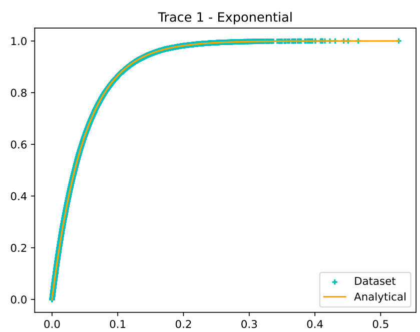
The uniform has been immediately discarded since it did not well represent the dataset distribution. From the plot all the other distribution seemed to be a valid choice but, computing the coefficient of variation (cv) of the different traces and considering the fitting parameters, the best choice seemed to be the exponential distribution. In particular, the fitting of the exponential distribution has been done by computing the fist moment of the inter-arrivals, which corresponds to the dataset mean.

$$E[X] = \frac{\sum_{i=1}^{N} x_i}{N}$$

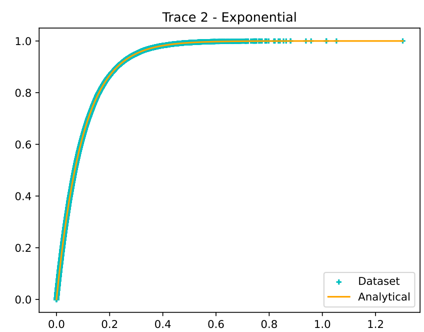
Then, the lambda has been computed by computing the inverse of the first moment (note that N is the number of samples).

$$\lambda = \frac{N}{\sum_{i=1}^{N} x_i}$$

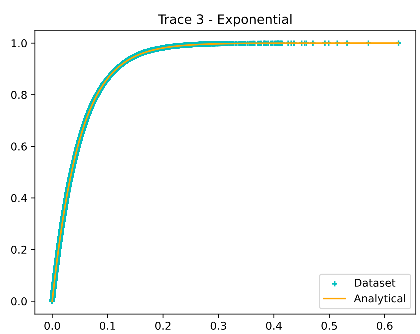
As a double-check, in order to be sure that the exponential was the right choice, the python script also contains the generation of the four analytical exponential distributions, which have been compared to the dataset distribution. The result was encouraging since the exponential distribution is a well representation of the dataset one, as it is shown in the four plots below.



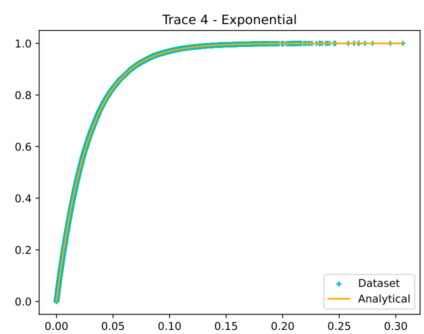
(a) TraceA.txt exp generation



(b) TraceB.txt exp generation



(c) TraceC.txt exp generation



(d) TraceD.txt exp generation

Figure 3: Comparison between analytical exp and dataset

In conclusion, the fitting of the four traces allowed to find the four exponential distributions, having the following parameters:

1. $\lambda 1$ = 20.0840 s^{-1}
2. $\lambda 2$ = 10.0661 s^{-1}
3. $\lambda 3$ = 19.8823 s^{-1}
4. $\lambda 4$ = 34.9285 s^{-1}

3.2 Datacenter analysis

After having done the fitting of the inter-arrivals distributions, the next step was the construction of the datacenter in JSIM graph software, in order to analyse the drop rate of the system and the drop rate of the different nodes to understand which layer needed an improvement (so to identify the bottleneck of the system).

The JSIM representation of the datacenter has been done following the project description. An open model has been created using 10 sources (four for the traffics and six for the background traffics), 1 sink and 6 queue station (one for each node of the datacenter).

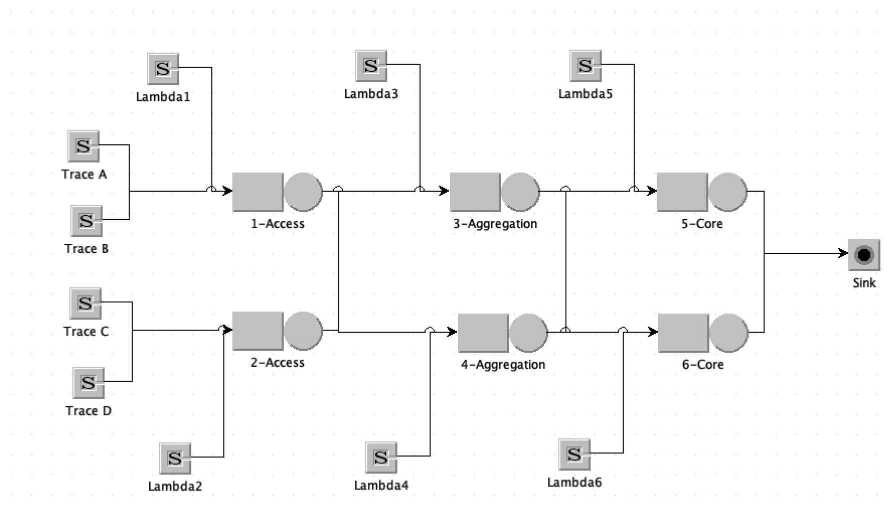


Figure 4: Original system configuration

The six background traffic have been configured using an exponential distribution (Poisson process) linked directly to the respective node. The four fitted distributions have been used for the four input traffics. Then each node has been configured with the respective service time distribution (for this step, for simplicity reasons, GB has been converted to MB multiplying by 1000 instead of 1024). Each node has been configured with a finite capacity of 16 MB (in JSIM it has been assumed that one job is equal to 1 MB) with a drop policy for the drop rule. Lastly, all the elements have been linked in the proper way. The routing of the switches up to the upstream nodes have been configured as random (equal routing probability) for the background traffic, instead probabilities (0/1) have been for the four traffic to respect the given routes.

After having configured all the elements using the right distributions and parameters, a simulation has been run. The result of the simulation showed that the system drop rate is 25.0634 MB/s on average, with a min value of 24.5410 MB/s and a max value of 25.6086 MB/s (0.99 confidence interval). Analysing the different nodes of the datacenter, it was evident that the one with most losses was the node 3 part of the aggregation layer with an average of 13.6832 MB/s. Another evidence is that, since the four input traffic distribution are different and they are not equally distributed among the nodes, some of the switches are more used than others. Below the analysis results are reported in detail.

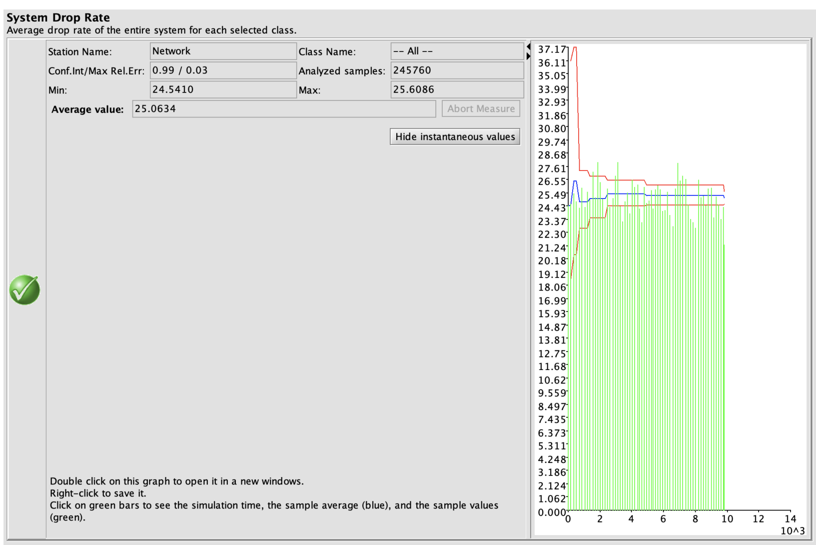


Figure 5: System drop rate analysis

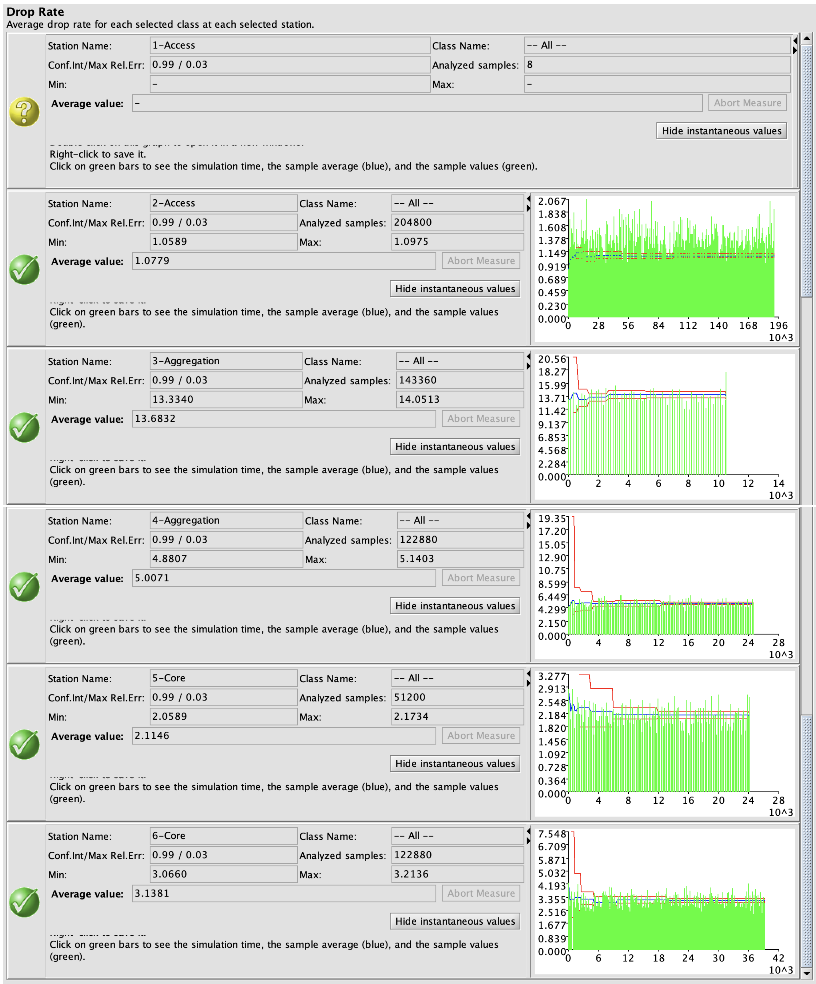


Figure 6: Nodes drop rate analysis

3.3 Datacenter improvement

Given the Datacenter analysis result, the most reasonable decision was to add a new node at the aggregation layer, to improve the bottleneck.

The first improvement has been the equally splitting of the background traffic, which has been splitted equally among the three aggregation nodes. This change lead to an improvement of the system drop rate of nearly 68%. The traffic leaving the two access nodes was redirected not only to nodes 3 and 4 but also on the new aggregation node with equal probability. So the routings have been splitted from the access node to the aggregation node between the original destination node and the new added one. Instead the four traffic leaving node 7 was still the same (A and B to 5 and C and D to 6).

This change still slightly improved the system performance but a new analysis showed that the the drop rate of node 6 was higher than the drop rate of node 5. Then, the last improvement has been the redirection of the traffic outgoing from the new node 7. Traffic A, B and C have been redirected to node 6 and traffics D has been redirected to node 5. This last improvement allowed to better balance the load between node 5 and 6, avoiding to have significantly differences, because now all the switches nearly manage the same amount of traffic avoiding as much losses as possible.

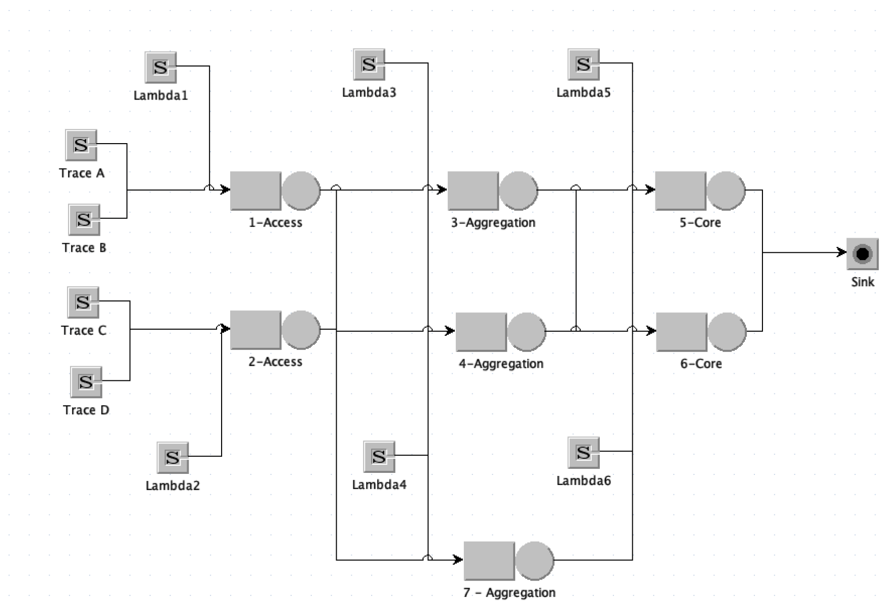


Figure 7: New system configuration

3.4 Outcome

The above mentioned improvements allowed to significantly reduce the system losses, which became 7.1782 MB/s, with a min value of 6.9817 MB/s and a max value of 7.3860 MB/s (0.99 confidence interval). This means an improvement of more than 71%. As desired, the two access nodes and the three aggregation nodes did not revealed significantly losses, as shown by the analysis below; this without affecting too much the drop rate of the cores node.

Figure 8: Nodes drop rate analysis

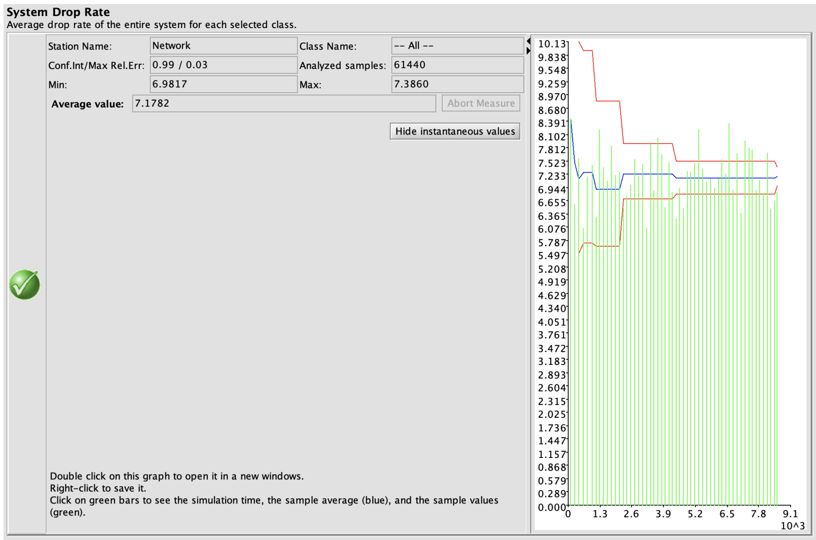


Figure 9: System drop rate analysis

3.5 Conclusion

In conclusion, we can say that since all the nodes of the datacenter are characterized by the same service time distribution and by the same finite capacity, the best solution to the problem was to add a node to the layer representing the bottleneck and balancing the load as much as possible by equally divide the amount of MB received.

Other alternatives have been considered for the solution of the problem but the explicated one represented the best choice that did not changed the system too much and that was able to significantly reduce the system losses by exploiting all the aggregation and core layer nodes in the best possible way, without making significantly changes unless a better traffic redirection.