Performance Evaluation of CRAN 1

Università di Pisa

Giacomo Pellicci, Nicola Mota, Alexander De Roberto

February, 2019

Contents

1 Modeling

1.1 Introduction

In these paragraphs we will explain how we modeled the CRAN 1 system as asked in the specifications.

- An **Application Server** (**AS** from now on), which generates packets to be transmitted to various cells. The size is variable according to a certain distribution that will be explained later on. These packages will then be forwarded instantly to the **BBU**.

  Each packet will have one and only one cell as its destination. The destination cell will be chosen uniformly among those available.
  The **packet size** s is taken from two different distributions depending on the scenario to be analyzed. It can be exponential or lognormal.
  Packets are generated every t seconds, where t is an exponential RV.

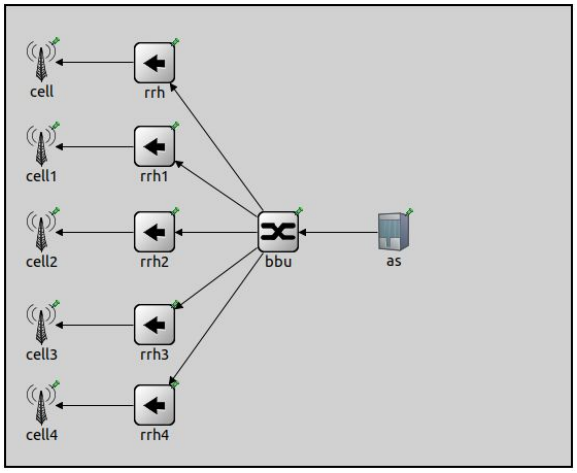
- A **Central Processing Unit** (**BBU** from now on), which will forward the packet received from the AS to the target cell. The BBU has N links, one for each RRH, at fixed speed **C** bytes/s . The BBU can only transmit on one of those links at a time, so when the link is busy packets must be queued and served using a FIFO policy.
  In addition to that, depending on the scenario, the BBU can perform compression operation on data packets before forwarding them to RRHs, thus reducing their size by **X%**. By the way this operation will require a certain amount of time $S = X \cdot 70ms$. The BBU can only compress 2 packets in parallel. In case both compressing processes are busy, incoming data packets are queued and served using a FIFO policy. Queues are assumed to be infinite.

- **N Remote Radios** (**RRH** from now on), which will receive packets from the BBU through a dedicated link. If compressed packets are received, the RRH will perform the decompression operation. In the end the RRH will forward the packet to its own cell. Both decompression operation and forwarding are assumed to be instantaneous.

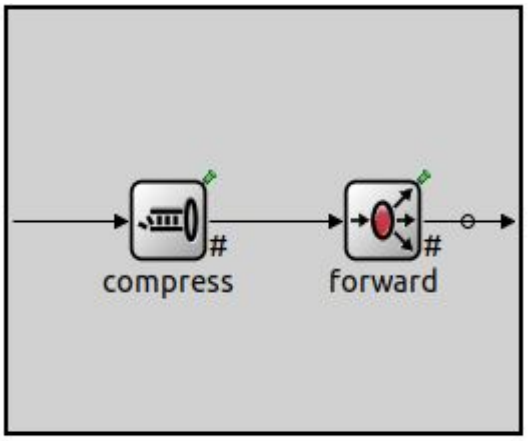
- **N Cells**, which will receive the packet forwarded by their RRHs. Cells are exploited to compute the end to end delay in the simulation.

1.2 Network topology

In order to build the model and run simulation we exploited OMNeT++ 5.4.1.
Cells, which are the final recipients of the packets, were used to compute the end to end delay for each received packet. Moreover, we asked the BBU to compute some other statistics like the number of jobs in the queue, both for the compression queue and for the forwarding queue.
In the high-level .ned file we can see the network topology, i.e. by setting $n = 5$ we obtain the following:



Network topology



Internal view of BBU

Of course the BBU has been developed as a *Compound Module*, divided in two *submodules*, one for the compression processes, the other for the simple forwarding operation via the RRH link to the target cell.
Compression is switched ON/OFF by setting the boolean *COMPRESS* inside the *compressUnit.ned* file during the configuration. In case the compression is switched off the *compressUnit* will simply forward the packet to the *forwardUnit*.
All the *.ned* files can be found in */src* together with the *.h* and *.cc* files.

2 Simulation

2.1 Introduction

The specification asked to measure at least the end to end delay for various value of S, which is the time spent in the compression process by a generic packet. In particular two scenario have to be evaluated, the one with the RV s exponentially distributed and another one where s is lognormal. While during validation N will be larger than 1, during simulation it will be kept equal to 1, just because the end to end delay and other statistics are not affected by the number of Cells/RRHs allowing us to avoid inserting a fictitious entity to collect statistics. The time between the generation of two packets is called t and it's an exponential distributed RV. The line capacity C of the link between each BBU and RRH is fixed and equal for each link. So we have to use many different RNG, due to the RVs t and s, the destination of the packet which must be picked from a uniform distribution and another degree of randomness about which of the two compressing processes should be choose in case both are idle.

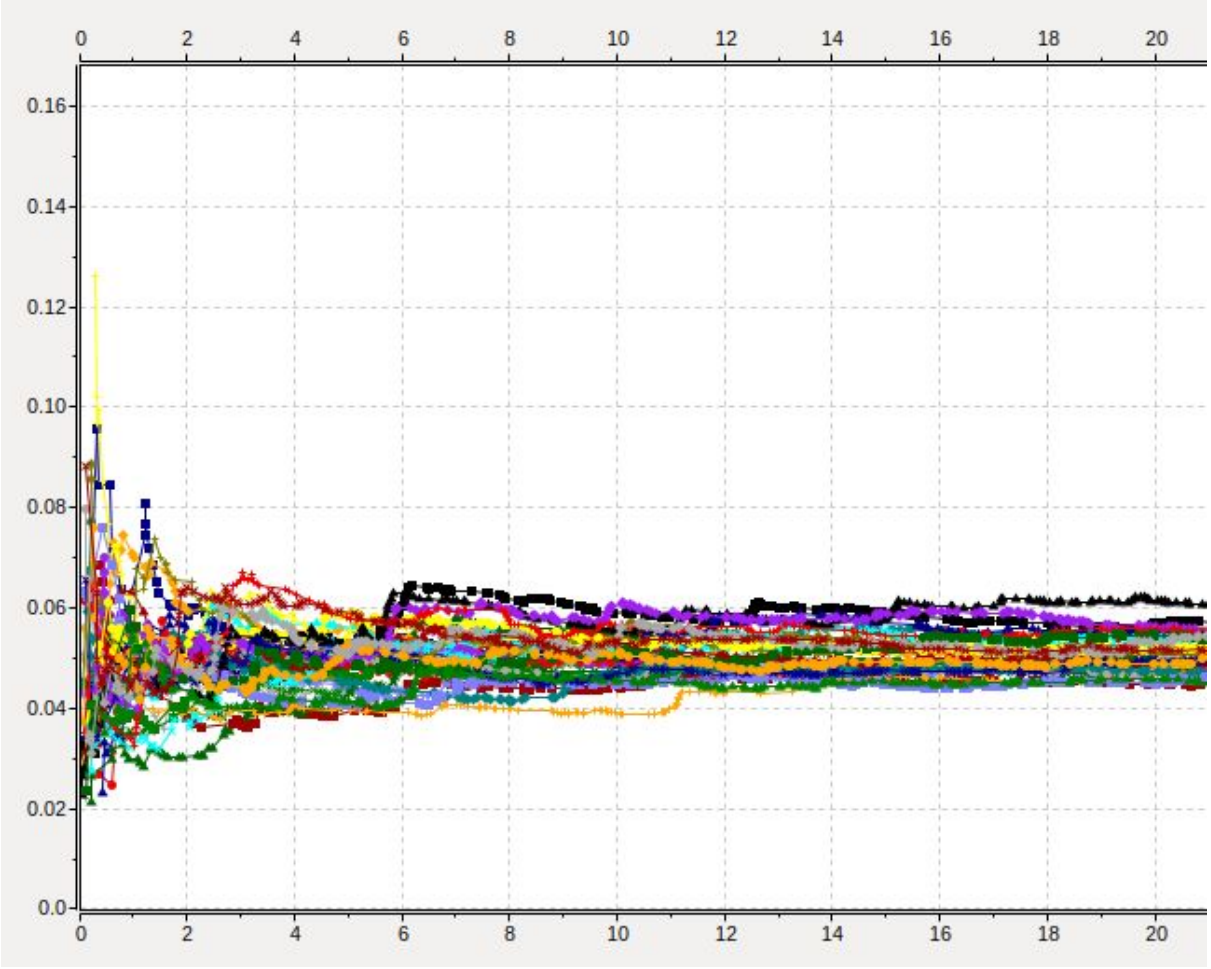
So in the end we have:

- Number of Cells/RRHs, $N = 1$
- Mean size of packets, $E[s] = 1KB$
- Line capacity, $C = 30KB/s$
- Compression ratio, $X = \{10, 20, 30, 40, 50\}$
- Number of repetitions, $REP = 35$
- Simulation Time, $ST = 1h$
- Warmup period, $WP = 4s$

The parameter to be used for the exponential and lognormal distributions will be computed later on according to system stability conditions.

2.2 Warm-up Period Estimation

In order to collect meaningful results a warm-up period must be added to the simulation. This way the simulation will start to record data from a certain point in time where we are pretty confident that the system has reached a steady state.
For both scenario (exponential and lognormal distribution of size s) we plotted the graph of the end to end delay. For each repetition we applied a sliding moving average and then we identified the time it took to stabilize around a certain value. Finally we chose the worst value of that time (the greatest one) as the warm-up period.
The worst case behaved like that:



So we picked **6s** as warm-up period.

3 Validation

3.1 Introduction

Before running a simulation with all the parameters we are interested in, we will test our model in simpler scenario, in order to validate it. In particular we want to check if the model correctly reproduces the expected behavior of the system. During validation we used the scenario in which the size is exponentially distributed, because it's easier to be analyzed. We will consider our model reliable if all those tests will be successful.

3.2 1st test: No compression

During this test we turned off the compression, so the end to end delay will be equal to the response time of the *forwardUnit*. In particular the *forwardUnit* can be modeled as an M/M/1 queuing system, which parameters are:

- Arrival rate $\lambda = \frac{1}{E[t]}$
- Mean service time $\frac{1}{\mu} = \frac{E[s]}{C} \Leftrightarrow \mu = \frac{C}{E[s]}$

From the theory we have that the stability condition of such a system is:

$$\rho < 1$$

$$\frac{\lambda}{\mu} = \frac{E[s]}{C \cdot E[t]} = \rho < 1$$

$$C > \frac{E[s]}{E[t]}$$

Taking $E[s] = 1KB$, $E[t] = 50ms$, we have:

$$C > 20KB/s$$

Obviously if we choose a too big C, the forwarding time will be much much smaller than the compression time S, which goes from 7ms (when X = 0.1) to 35 ms (when X = 0.5). Therefore, we have chosen $C = 30KB/s$, this way the forwarding time will be

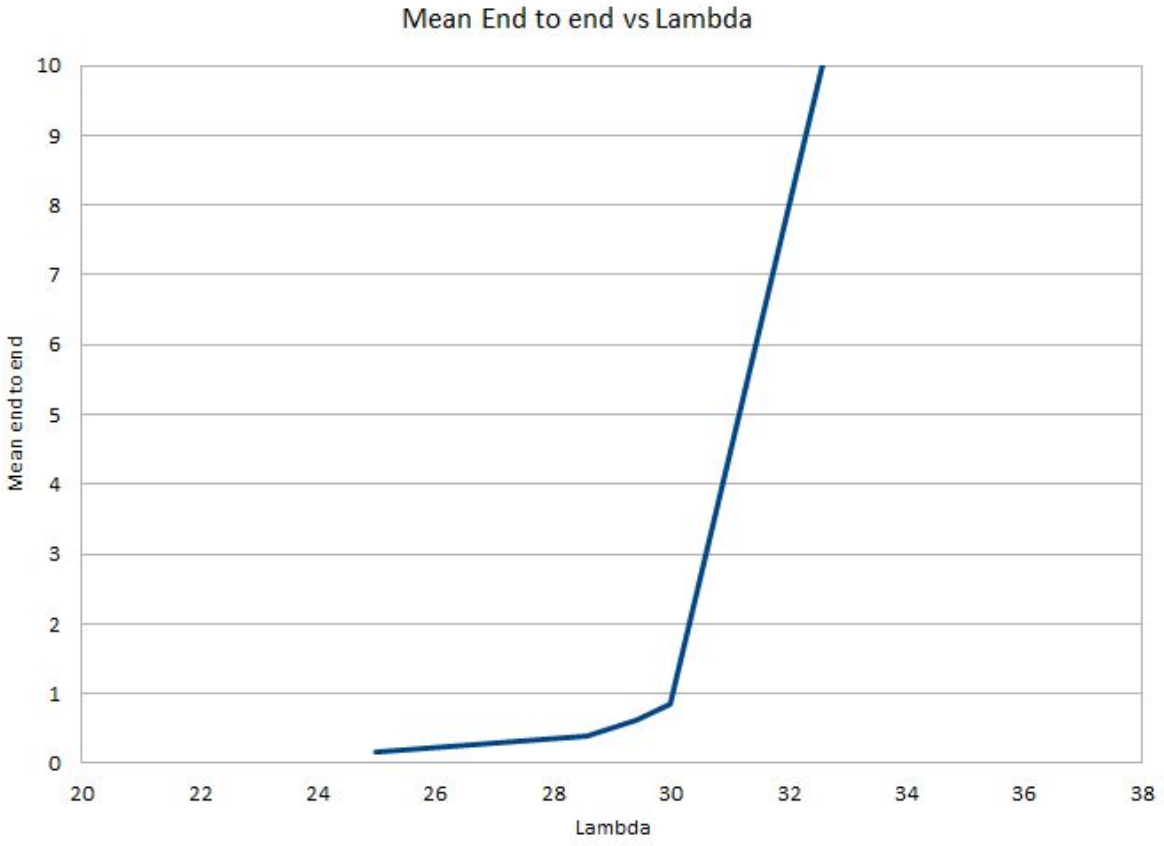
on average $\frac{E[s]}{C} = \frac{1KB}{30KB/s} \sim 33ms$, which is a time comparable with S. As counterproof, if we choose $C = 30MB/s$ the forwarding time will be $\frac{1MB}{30KB/s} \sim 0,033ms$ which is much smaller than S, this means that compressing packets will always be disadvantageous.

Therefore, we choose $C = 30KB/s$ so that $\rho = \frac{2}{3}$.

Moreover, analyzing the parameter $E[t] = \frac{1}{\lambda}$ we obtain that the limit condition for the stability with fixed parameter $E[s]$, C it's:

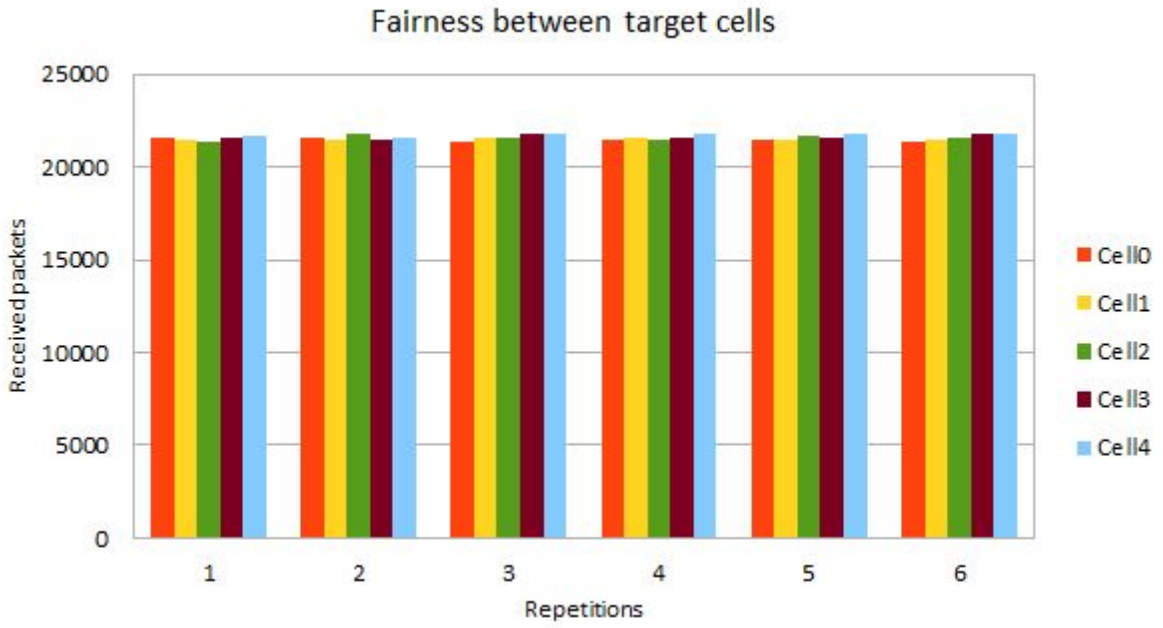
$$E[t] > 0.03333 \Leftrightarrow \lambda < 30.003$$

As discovered during the test the system follow exactly this condition, being stable for value of $\lambda < 30.003$, becoming unstable otherwise.
We can see here a graph of the mean end to end delay as function of λ:



We clearly see that instability kicks in when λ approaches the limit value $\lambda_{max} = 30.003$.

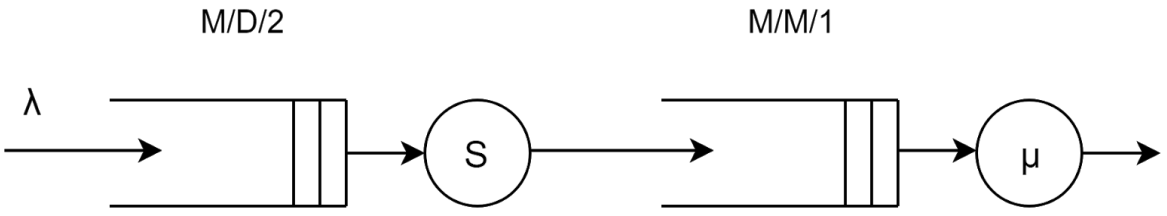
Moreover, just to be sure that the model is correct we computed for different repetitions the number of packets received on each cell, and then checked that those number are uniformly distributed.
After 1h of simulation we obtained the following results:



Given that the percentage difference between the maximum and minimum number of received packets is ~1%, we can state that the packets are uniformly distributed.

3.3 2nd test: Compression, fixed parameter

In order to validate the system we need to test it also when the compression is enabled. We can summarize the system as a M/D/2 linked to a M/M/1



Where for the M/D/2 the service time is deterministic and constant, equal to $S = X \cdot 70ms$. The M/D/2 system can be considered stable if:

$$\lambda < 2\mu \Rightarrow 1 > 2 \cdot 70ms \cdot \frac{X}{E[t]}$$

$$E[t] > \frac{X \cdot 70ms}{2}$$

So this should be solved for each value of X that will be used. By the way the highest X, which is 50, will be the limit condition, so:

$$E[t] > \frac{0.5 \cdot 70ms}{2} = 0.0175\ s \ \Leftrightarrow\ \lambda < 57.1428$$

Once the steady state is reached, the throughput of the M/D/2 will be equal to its arrival rate ($\gamma = \lambda$) so we can use λ as arrival rate for the M/M/1.

Of course the M/D/2 has the compression purpose, so the output packets will have their own size reduced by X%.
We can say that $newSize = (1 - X) \cdot size$. Let's compute $E[newSize]$:
From the theory we know that:

$$E[X + Y] = E[X] + E[Y]$$

So our case is similar:

$$E[newSize] = E[1 \cdot size - X \cdot size] = E[size] - X \cdot E[size] = (1 - X) \cdot E[size]$$

Recalling what was said in the previous paragraph regarding the M/M/1 system, we have that:

$$\mu = \frac{C}{(1-X) \cdot E[s]} \ \Leftrightarrow\ \frac{1}{\mu} = \frac{(1-X) \cdot E[s]}{C}$$

From this we can derive the stability condition for a M/M/1 system:

$$\frac{\lambda}{\mu} = \rho < 1$$

$$\rho = \frac{(1-X) \cdot E[s]}{C \cdot E[t]} < 1$$

Following the choice taken in the previous paragraph, we fix $E[s] = 1KB$ and $C = 30KB/s$ we have:

$$\rho < 1 \Rightarrow \frac{(1-X) \cdot E[s]}{C \cdot E[t]} < 1 \Rightarrow E[t] > \frac{(1-X) \cdot E[s]}{C}$$

For $X = 0$ we find the same ρ of the previous paragraph.
While X is varied between 10 and 50 (10, 20, 30, 40, 50) we have different condition. Once more the strict condition is the one for the smallest X, which is 10 so:

$$E[t] > 0.03 \Leftrightarrow \lambda < 33.333$$

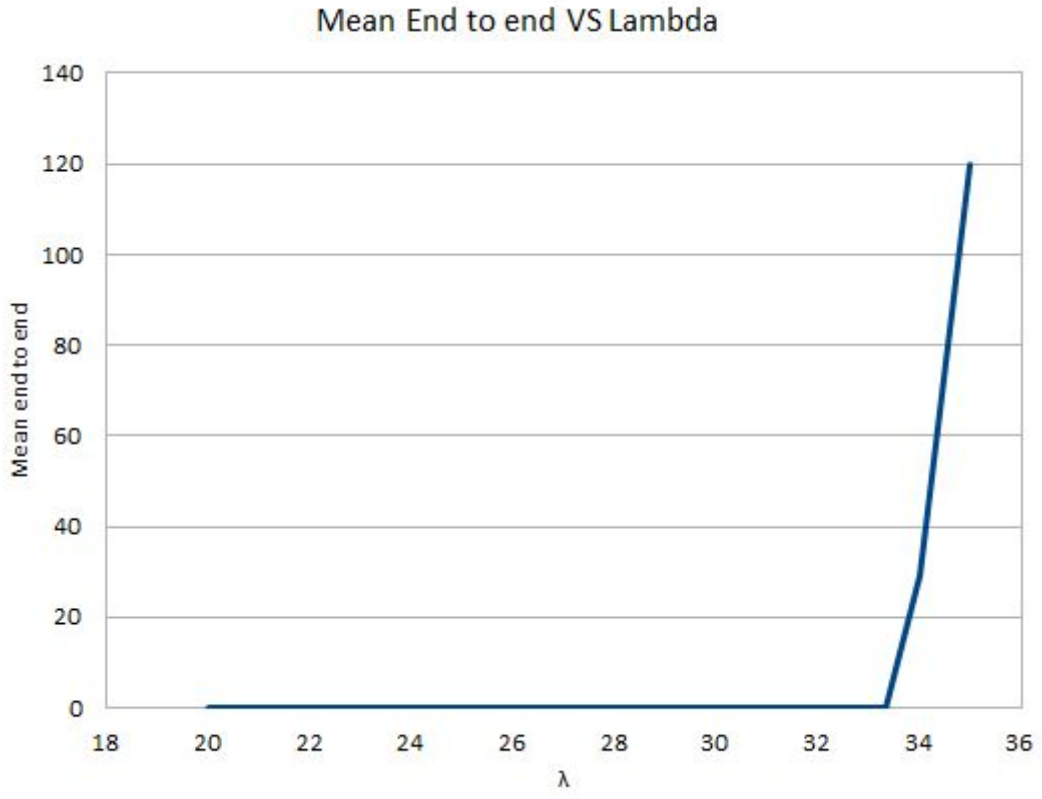
So pairing the two condition that we have found:

$$E[t] > 0.0175\, s \Leftrightarrow \lambda < 57.1428$$
$$E[t] > 0.03\, s \Leftrightarrow \lambda < 33.333$$

So the new condition is:

$$E[t] > 0.03 \Leftrightarrow \lambda < 33.333$$

While this relations hold for s and t taken from exponential distribution, we analyzed a simplified scenario, where we removed randomness for the size and the interarrival time of the packets. We can show in fact that in our model the stability condition is the same found via analytical computations, which confirms our test. The results, for $X = 0.1$, are the following:



For $X = 0.1$ you have that the stability condition is $\lambda < 33.3$ in the graph we see exactly this behavior.
We can clearly see that the end to end delay is stable, to the correct value which is $S + \frac{(1-X)\cdot 1KB}{C} = 37ms$ but suddenly, once the system becomes unstable it grows indefinitely. The behavior of the model is congruent with what is expected, so the test is considered passed.

4 Simulation: Exponential scenario

4.1 Introduction

In this scenario we will consider the size of the packet as a RV which is exponentially distributed, with mean $E[s] = 1KB$ as computed in the validation chapter. Moreover, $C = 30KB/s$. While keeping C and $E[s]$ fixed, we moved the parameters X and $\lambda = \frac{1}{E[t]}$ across a range of values, which are:

$$E[t] = \{35ms,\ 40ms,\ 50ms,\ 75ms,\ 100ms,\ 1000ms\}$$
$$X = \{0,\ 10,\ 20,\ 30,\ 40,\ 50\}$$

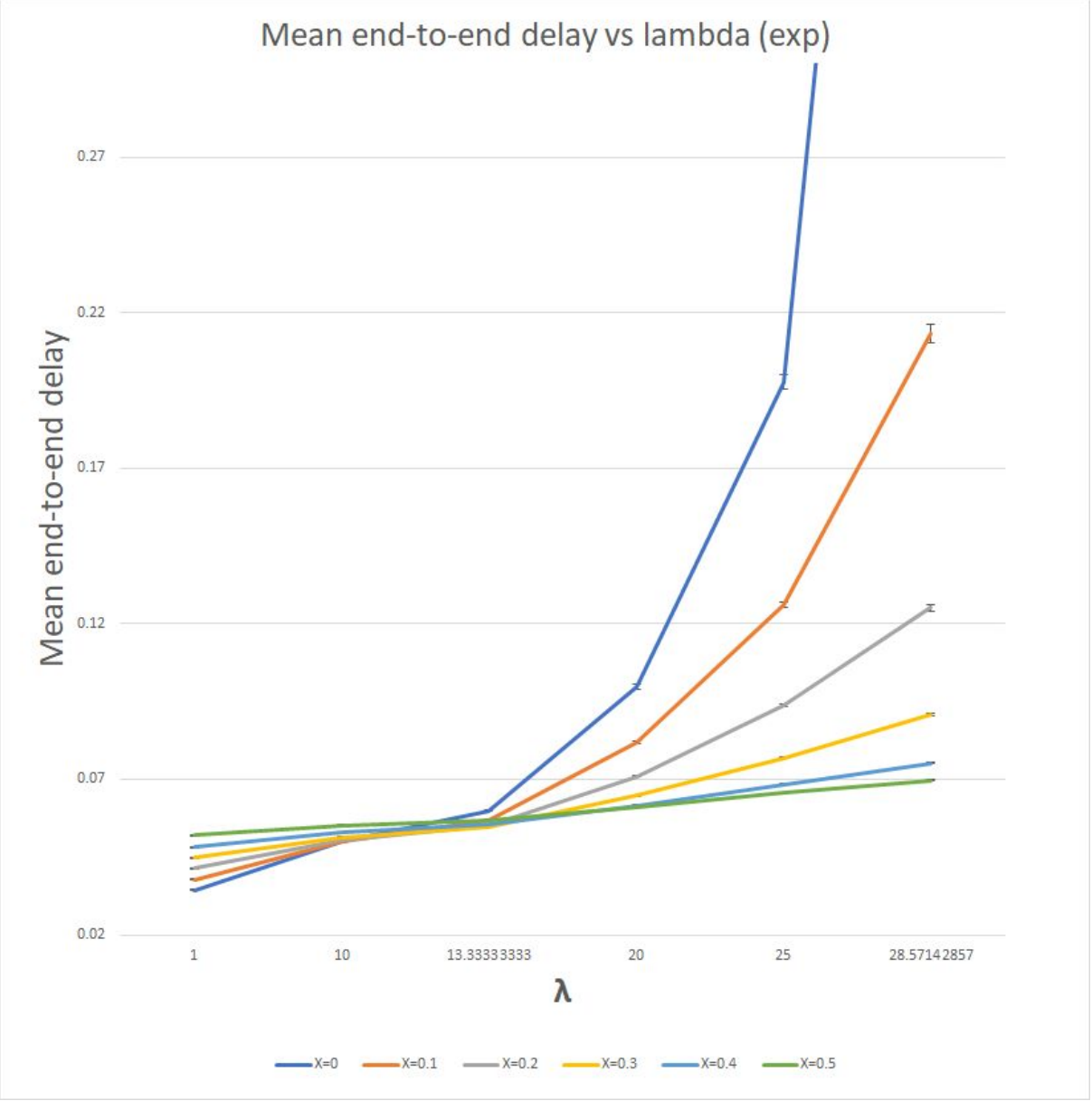
This way we tried to evaluate the system under different working condition, scaling from few packets per second up to many packets per second. The main performance index we decided to analyze are *Mean end to end delay* and the *Mean throughput*.
All the statistics are plotted with 95% confidence interval for the mean value. We have considered 35 repetitions for each experiment, so that the sample was large enough ($n \geq 30$) making reasonable to use the following formula to compute the confidence interval:

$$\left[\overline{X} - \frac{S}{\sqrt{n}} \cdot z_{\frac{\alpha}{2}} , \overline{X} - \frac{S}{\sqrt{n}} \cdot z_{\frac{\alpha}{2}} \right]$$

Where $z_{\frac{\alpha}{2}}$ is the $\left(1 - \frac{\alpha}{2}\right)$ percentile of the Standard Normal and S is the sample standard deviation.

4.2 Mean end to end delay

First of all, we have analyzed the mean end to end delay under different workload(λ), for each value of X. For each repetition we computed the mean end to end, then we found out for the specific couple of λ, X the sample mean end to end delay and it's 95% confidence interval. The overall result is shown here:



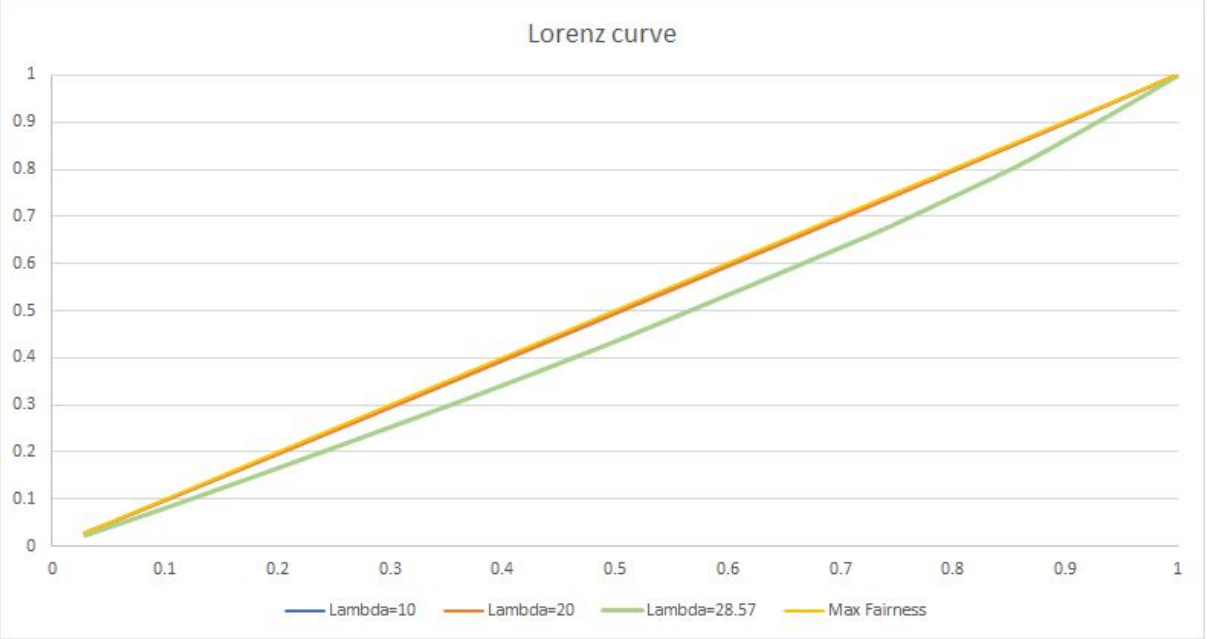
As we can see different behaviours are shown for different parameters:

- For $\lambda \leq 10$ we can see that forwarding directly packets without compressing them is the preferable method, because its mean end to end delay is slightly lesser than the others.

- For $\lambda \geq 13$, we have the opposite, compressing the packets help the system to keep the mean end to end delay low. In particular we can see, for the case in which $X = 50$, that the mean end to end delay is increasing approximately linearly with the packet arrival rate λ. Moreover, for values $13.33 \leq \lambda \leq 20$ the compression factor should be picked accordingly to the smaller mean end to end delay. For $\lambda > 20$ the best choice is to compressi with $X = 50$.

Larger values of λ are not used just because we didn't want to show the unstable condition, which are not the ones used in real applications.

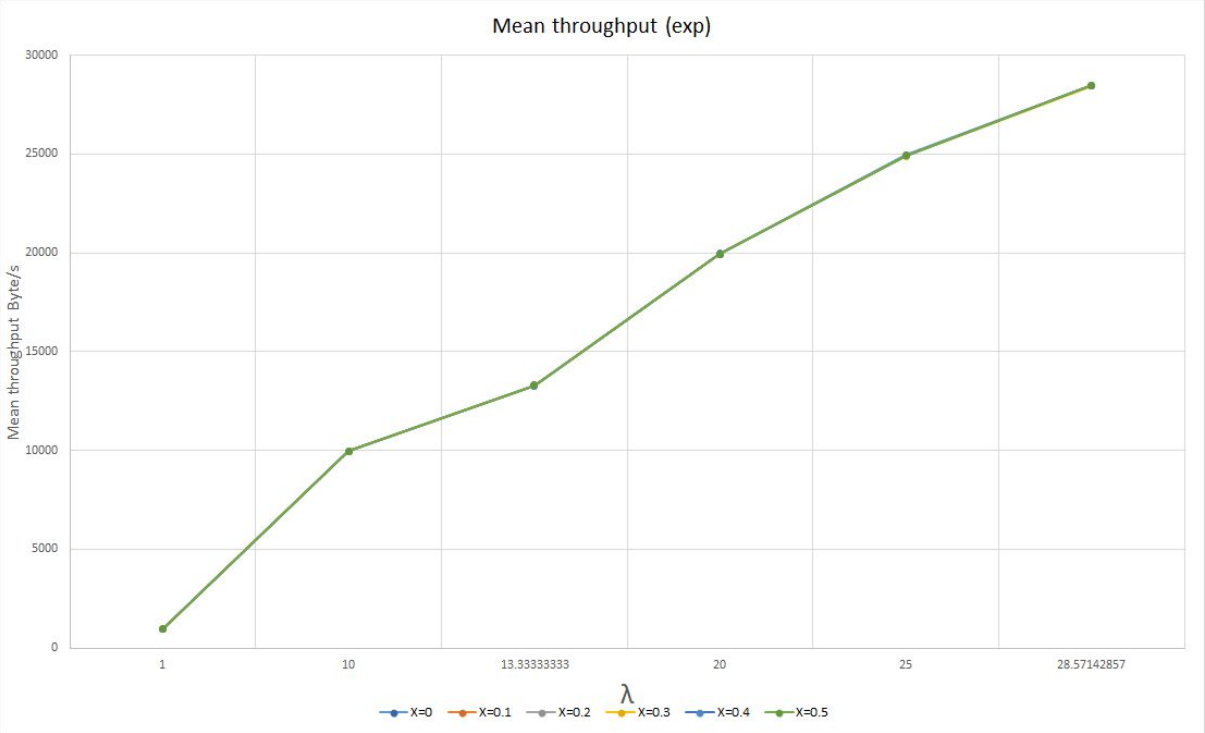
Moreover, we computed the Lorenz Curve for the mean end to end delay that show up differences between the mean response time experienced from packets over all the repetitions of each experiment. The curve is as follows:



As λ increases the fairness decreases instead. This means that for higher arrival rate, and so system workload we have that the mean end to end delay experienced from a packet is more variable. This happens because in general the queue will contain some packets, so the end to end delay is conditioned also by the size of those packet in the queue. This makes the mean end to end delay less fair with respect to the case where the queue are usually empty, in which the delay depends only on the packet size and the line capacity.

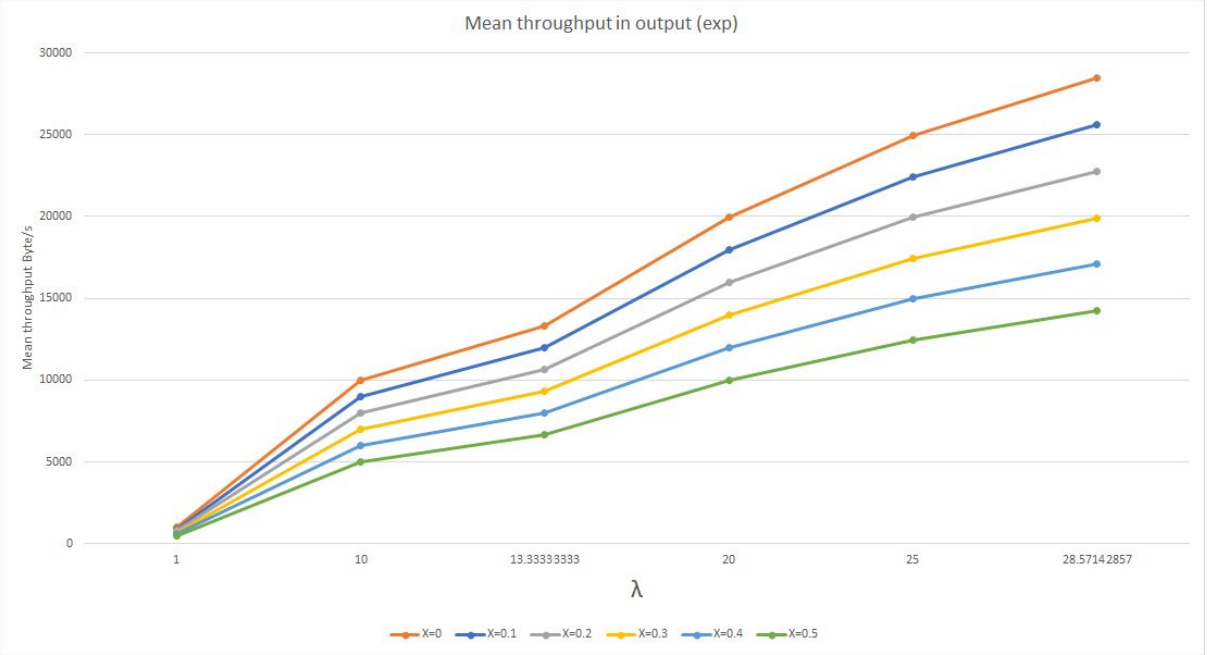
4.3 Mean throughput

The mean throughput has been analyzed for each value of X and λ. The results, with a 95% confidence interval is as follows:



This result is not that much surprising, because considering the definition of throughput as "number of jobs served per unit time", so even if compression is performed, the amount of byte processed per unit time (i.e. *Byte/s*) is still the same for each value of X.

If interested in the throughput seen at the output of the BBU, considered as "number of jobs forwarded on the link per unit time" we have:



Which is coherent with what is expected, because compression reduces the size of the forwarded packet, hence the throughput will be scaled according to the compression ratio.

5 Simulation: Lognormal scenario

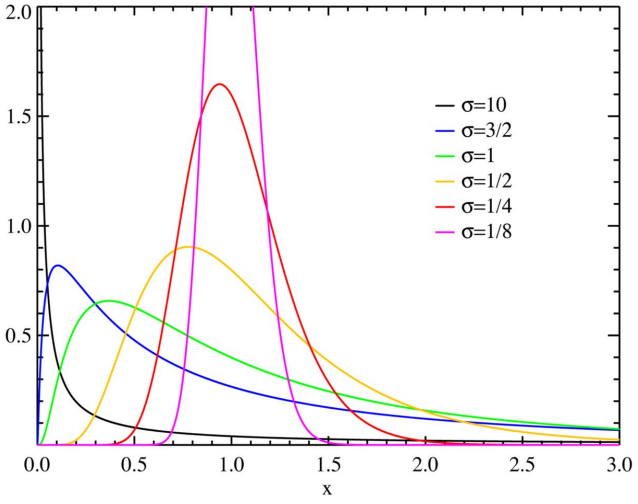
5.1 Introduction

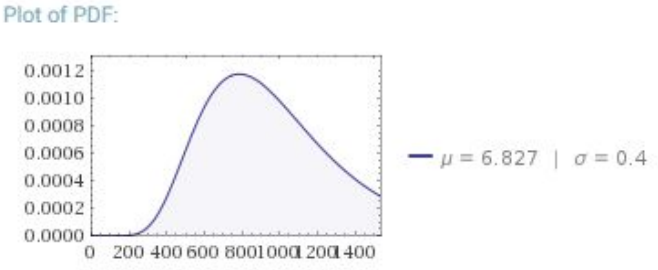
In this scenario we will consider the size of the packet as a RV which is lognormal. We will assume as **mean** $E[s] = 1KB$, which is the same as the exponential scenario.

By definition if s is lognormal, then the RV $X = ln(s)$ has a normal distribution with **mean** μ and **variance** σ^2.
Moreover, we have that:

$$E[s] = e^{\mu+\frac{\sigma^2}{2}}$$

We decided to set $E[s] = 1KB$ and $\sigma = 0.4$, this way solving the previous equation we obtained $\mu = 6.827$.



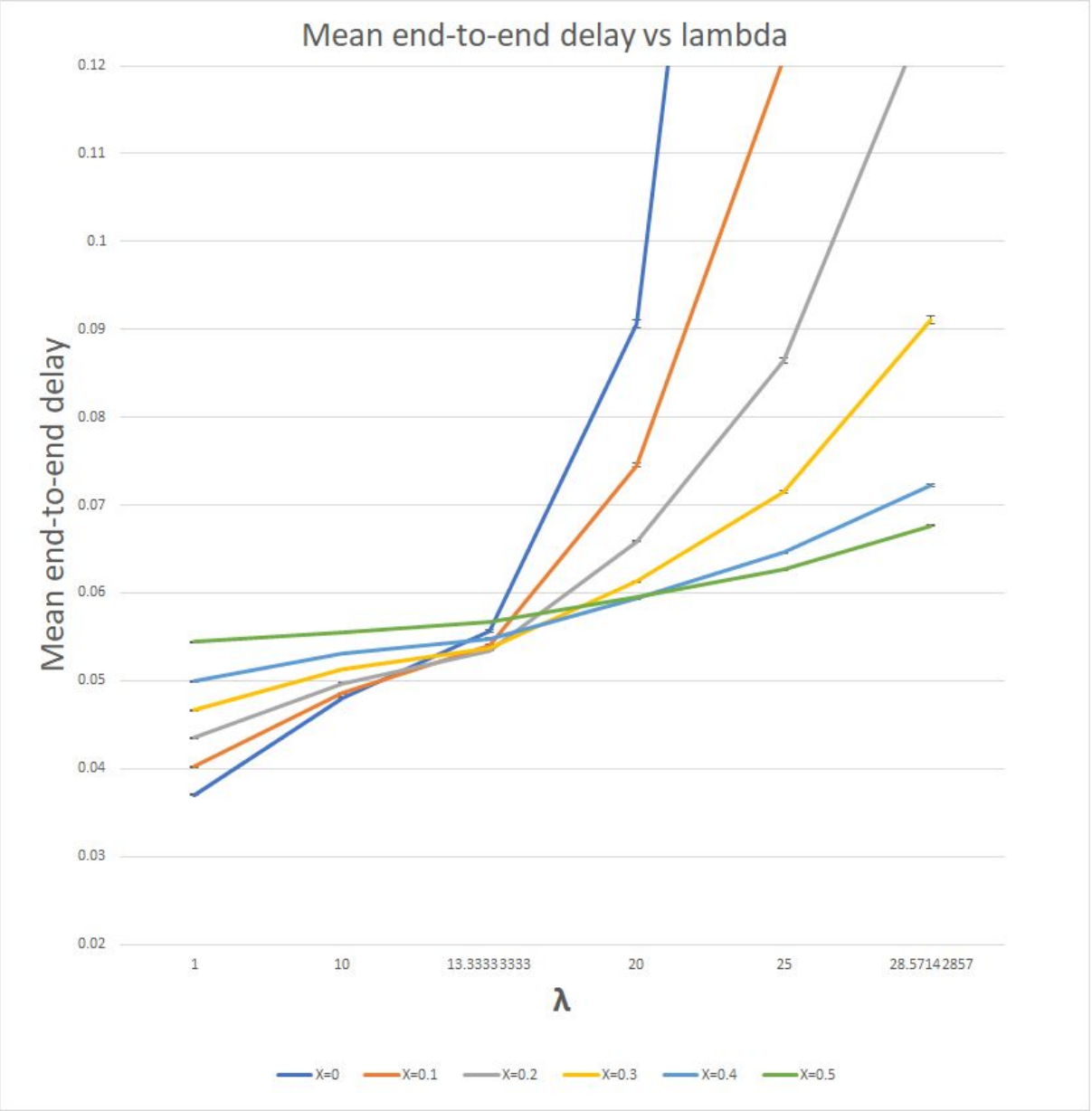


This is the PDF of the lognormal distribution that has been used.

5.2 Mean end to end delay

As in the previous scenario we analyzed the mean end to end delay under different values of the parameters λ, X. We have computed the mean end to end delay for each repetition for each possible combination of λ and X. Then we computed the sample mean of all of the 35

repetition, showing the result with a 95% confidence interval. The following graph shows the result that has been collected:



The graph reveal a particular behavior with regard to the mean end to end delay:

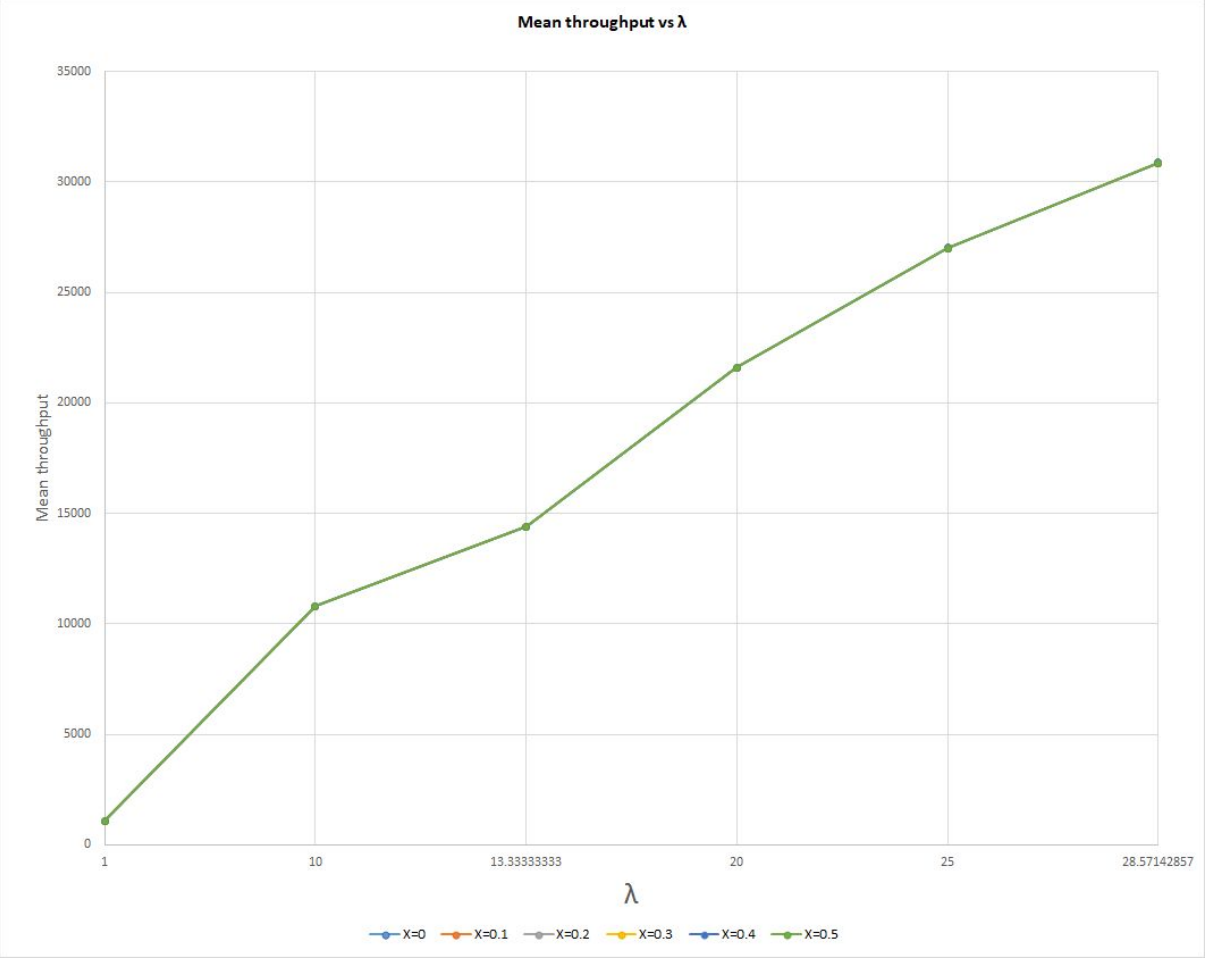
- For $\lambda \leq 10$ packets that has been compressed experience a higher mean end to end delay, making the no compression method preferable.
- For $\lambda \geq 13$, as the packet arrival rate increases, we find out that compressing the packets is the best option, in particular we can see that as λ increases, the more the packet is compressed the lower the mean end to end delay will be with regard to other compression configurations. By the way this behavior is not well defined.
    - For $13 \leq \lambda \leq 20$ compression is the best method, by the way we obtained that, thanks to the small confidence interval, the compression used (X) should be

picked according to the rate λ because the mean end to end delay start to grow faster when compression is low, while being better for lower values of λ.

- $\lambda \geq 20$ it's clear that beyond that value the lowest mean end to end delay is achieved thanks to the highest compression (50%).

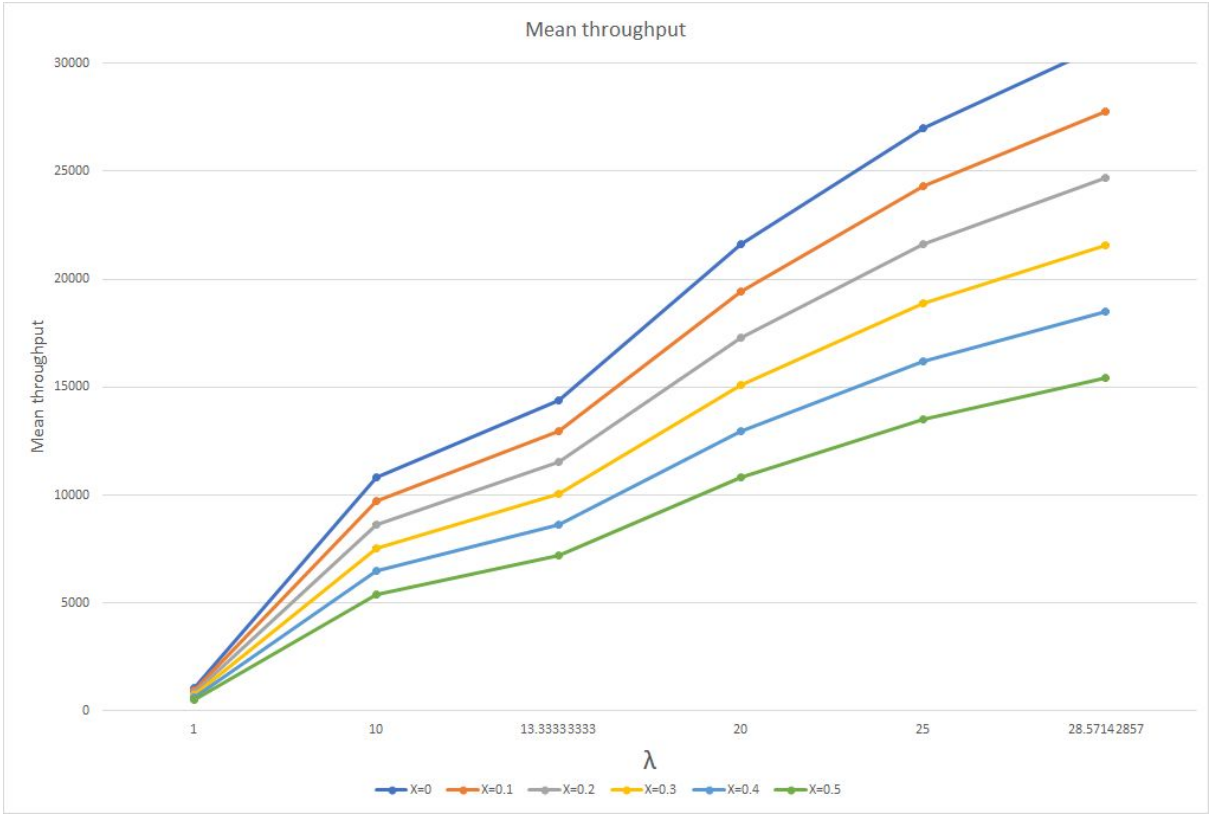
5.3 Mean throughput

The mean throughput has been analyzed at 95% confidence interval, moving across a set of value for both λ and X.



This graph shows the behavior of the mean throughput as the "mean number of jobs served per unit time". This means that it's not important if the compression (for any value of X) is performed or not, the amount of byte processed per unit time (i.e. *Byte/s*) is still the same and it depends on the size of the packets.

The definition of throughput can also be understood as “number of jobs forwarded on the link per unit time”. According to this definition, the following graph shows the throughput at the output of the BBU:



Obviously we can see that the more the compression ratio increases the more the throughput decreases. This happens because the compression reduces the size of the packets.