Computer Engineering

Performance Evaluation of Computer Systems and Networks

Aerocom System 2

Project Documentation

Alessio Di Ricco

Edoardo Geraci

Mariella Melechì

2022/2023

Table of Contents

1 Introduction

1.1 Objectives:

The objective of the project is to evaluate the performance of two different implementations of a communication protocol, used by *Aircrafts* to communicate with a *Control Tower* via *Data Links* having a transmission capacity varying linearly in time. The two implementations are:

- **Unmonitored**: The Aircraft *keeps the same serving Data Link* for the whole simulation;
- **Monitored**: The Aircraft constantly monitors the service time of Data Link and *before each transmission it selects the Data Link* with the highest actual capacity. Monitoring Data Link service time gives *X% malus to capacity*.

The evaluation will focus primarily on the performance indexes regarding response time, waiting time and queue length.

2 Model

2.1 Components:

- **Aircraft (AC)**: Main module of the analysis composed by:
  - **Packet Generator**: Sub Module which generates fixed length packets with exponential generation distribution. These packets will be handled by the Link Selector which is to be described later.
  - **Link Selector**: Sub Module which manages a FIFO queue, and is responsible for sending packets. At the start of a simulation, if the unmonitored operation mode is chosen, a random Data Link will be selected. If, instead, monitored mode is chosen, the Link Selector picks the Data Link with the highest capacity whenever a new message needs to be sent. While operating in monitored mode, the act of monitoring the various Data Link generates a penalty of X% to the overall capacity.
- **Data Link (DL)**: A DL is the link between ACs and the CT. There is a fixed number of DLs available to all the ACs. Each DL has a capacity that varies linearly with time t.
- **Control Tower (CT)**: Receives packets and drops them. The Control Tower simulation is simplified as it is not the main focus of the analysis
- **Packet**: Created by the packet generator, it's sent to the CT.

2.2 Assumptions:

- DLs used by ACs to communicate with the CT are considered **ideal**; each channel can be used by any number of ACs at the same time without concurrency or interference.
- ACs are independent and the handling of messages by one AC does not interfere with the handling of all the others.
- The capacity of a DL is the number of bytes per second that the DL is able to transmit. Knowing the size of each packet, which is fixed, and the capacity at the time of communication we can then compute the service time easily:
  - sPCK: size of the packet [byte]
  - cDL: capacity of the DL [byte/second]
  - ST: Service Time = $\frac{sPCK}{cDL}$ $\frac{[byte]}{[byte/second]}$
- We have 3 different RNGs, one for the interarrival time, one for the capacity setting time and one for the DL selected in the unmonitored mode.

Parameter	Values
Capacity malus	X
Capacity time generation mode	Exponential / Lognormal
Data link capacity interval	[minCapacity, maxCapacity]
Setting capacity time	t
Number of Aircrafts	nAC
Number of Data Links	nDL
Operation mode	monitored / unmonitored
Packet generation exponential mean	kMean
Packet Size	sPCK

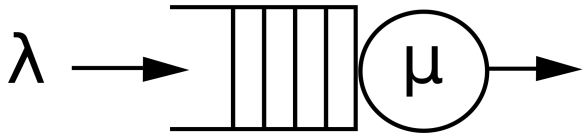
3. Model Validation

3.1 Memory analysis

The program's memory safety was validated with the help of Valgrind, a programming tool for memory debugging, memory leak detection, and profiling, which didn't encounter any issue.

3.2 Analysis of the stability condition

Given the nature of our system, each AC can be simplified as an M|M|1 system, with an interarrival rate equal to $\lambda = \frac{1}{kMean}$ and a service rate equal to $\mu = \frac{1}{ServiceTime}$. As in every M|M|1 system, the stability condition is $\rho < 1$ or $\lambda < \mu \Rightarrow kMean > ServiceTime$.



We tested the system with different values of *kMean*. The results are reported in the table:

kMean	Service Time	Mean Response Time	Mean Queue Length
0.003	0.008407	31.294345	$\sim\infty$
0.009	0.008567	0.387522	43.141522
0.02	0.008606	0.021300	1.301352
0.05	0.008626	0.018495	0.729679

In red we have the experiments in which the stability criterion isn't met. As expected, the queue length becomes really high in a relatively low interval of time and clearly shows a linear growth to infinity (fig 3.2.1).

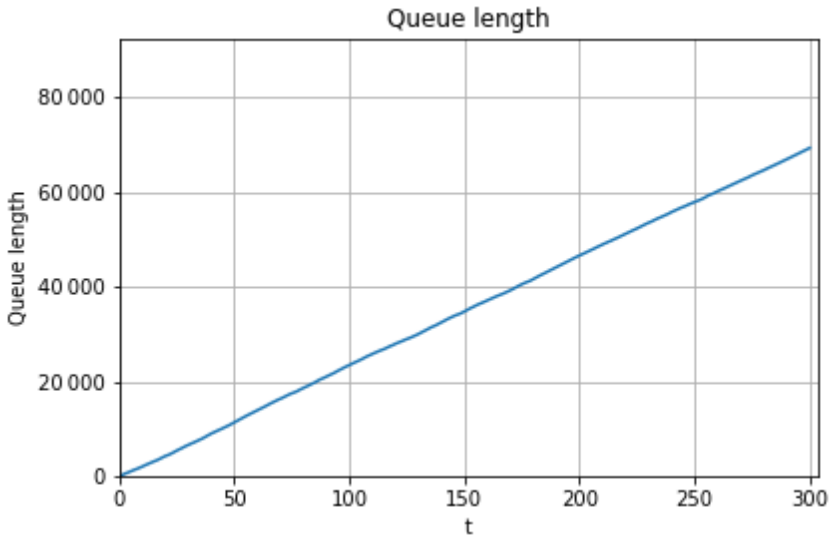


fig 3.2.1 with $kMean = 0.003$

After these considerations, $kMean = 0.05$ was chosen as it is a value that won't create any limit condition.

3.3 Degeneracy test

We tested the system in some limit cases:

- simulation with no *ACs*
- simulation with no *DLs*.

As expected, in both scenarios all statistics values were 0 (if computed at all, since the behavior of the system is to drop packets if $\#DL = 0$ and to just vary linearly the capacity and wait if $\#AC = 0$).

3.4 Packet loss

During system simulations the number of packets sent by the *AC* is exactly equal to the number of packets received by the *CT* as it regularly should.

3.5 Number of *ACs*

Since the DL is modeled as an ideal channel, no differences were measured by varying the number of simulated ACs. Thus, from this point onwards, each and every simulation will include only one AC.

3.6 Continuity test

After having chosen the kMean, a number of simulations were performed to prove that, after reaching stability, no critical conditions or unexpected behaviors are encountered.

kMean	{0.050, 0.051, 0.052, 0.053, 0.054, 0.055, 0.056, 0.057, 0.058, 0.059}

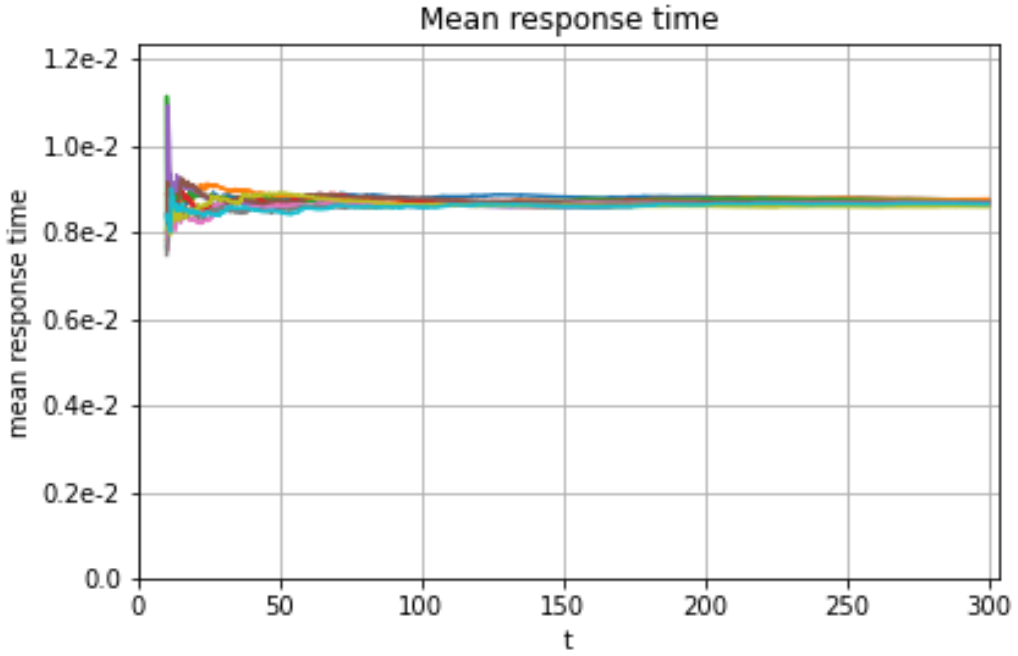


fig 3.6.1

3.7 Little's Law

Since the system, as previously stated, can be approximated as an *M|M|1* system when running in unmonitored mode, the validity of the model can be proven comparing the theoretical values of an *M|M|1* system and the measured values of the modeled system.

The values are reported in the table ahead.

Theoretical values	Measured values
$\mu = 105$	$\mu = 99.3$
$E[t_s] = \frac{1}{\mu} = 0.00954$	$E[t_s] = \frac{1}{\mu} = 0.010069$
$\lambda = \frac{1}{kMean} = 20$	$\lambda = \frac{1}{kMean} = 20$
$\rho = \frac{\lambda}{\mu} = 0.1905$	$\rho = \frac{\lambda}{\mu} = 0.2014$
$E[N] = \frac{\rho}{1-\rho} = 0.2353$	$E[N] = \frac{\rho}{1-\rho} = 0.2522$
$E[N_q] = E[N] - \rho = 0.0448$	$E[N_q] = E[N] - \rho = 0.051$
$E[R] = \frac{E[N]}{\lambda} = 0.01177$	$E[R] = \frac{E[N]}{\lambda} = 0.01147$
$E[W] = E[R] - E[t_s] = 0.00223$	$E[W] = E[R] - E[t_s] = 0.001409$

The differences between the theoretical and measured values are due to the fact that in the first one, the linear capacity variation of each DL does not get considered, contrary to what happens in the simulation.

The system in monitored mode is actually really similar to the one above, except for what regards μ which becomes $\mu - X\%\mu$. It is still possible to model the system as an M|M|1. The comparison between theoretical and measured values, with a 10% malus is in the table ahead. The computations for the first test were done with the theoretical capacity $C = \frac{maxCapacity + minCapacity}{2}$ while in the second test the theoretical capacity was set to $C = maxCapacity$.

This is because of the logic behind the monitoring: since the system chooses the best capacity possible each time it has to send a packet, chances are that the capacity chosen in the simulation is almost the maximum capacity possible for the *DLs*.

Theoretical values	Measured values
$\mu = 135$	$\mu = 125$
$E[t_s] = \frac{1}{\mu} = 0.00740$	$E[t_s] = \frac{1}{\mu} = 0.008008$
$\lambda = \frac{1}{kMean} = 20$	$\lambda = \frac{1}{kMean} = 20$
$\rho = \frac{\lambda}{\mu} = 0.1481$	$\rho = \frac{\lambda}{\mu} = 0.16$
$E[N] = \frac{\rho}{1-\rho} = 0.17391$	$E[N] = \frac{\rho}{1-\rho} = 0.19047$
$E[N_q] = E[N] - \rho = 0.02581$	$E[N_q] = E[N] - \rho = 0.0305$
$E[R] = \frac{E[N]}{\lambda} = 0.008695$	$E[R] = \frac{E[N]}{\lambda} = 0.008761$
$E[W] = E[R] - E[t_s] = 0.001296$	$E[W] = E[R] - E[t_s] = 0.000753$

4 Warm-up Analysis

In order to define the warm-up period, two main factors need to be considered: **queue length** and **response time**.

The system was tested in both monitored mode (shown in **fig. 4.1**) and unmonitored (shown in **fig. 4.2**) , and in both *lognormal* and *exponential* generation of capacity setting time. The plots were obtained from simulations which consisted of 10 repetitions each. This number of repetitions is a good enough indicator of warm-up behavior.

We can see that after 10s the behavior is stable, so we can set our warm-up time like so.

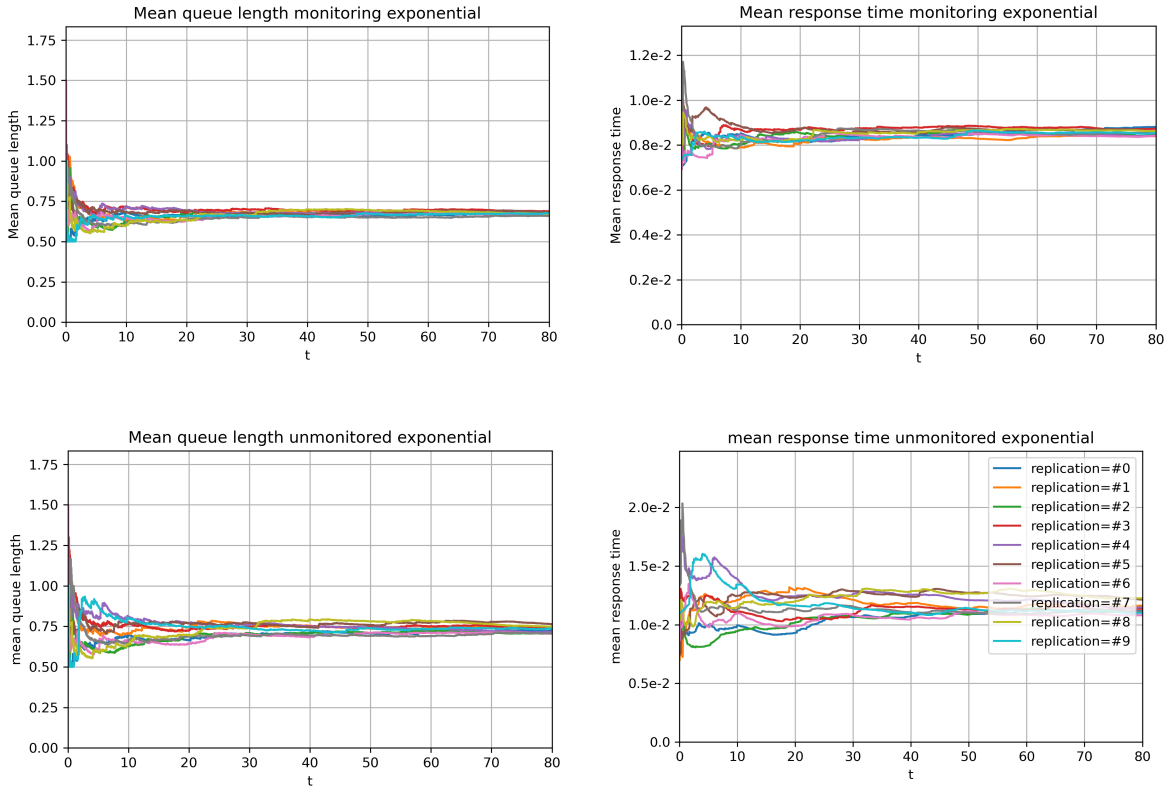
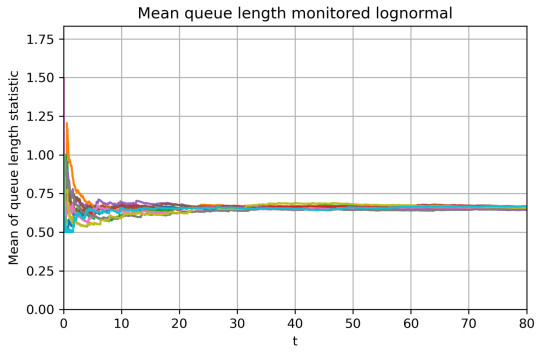
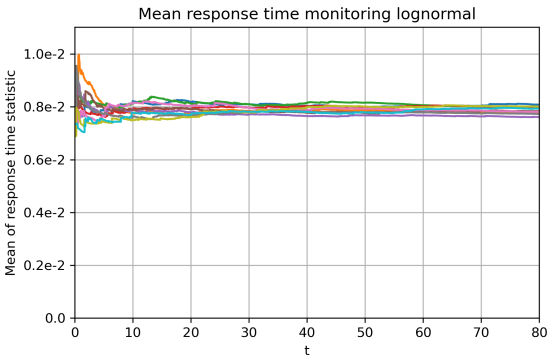
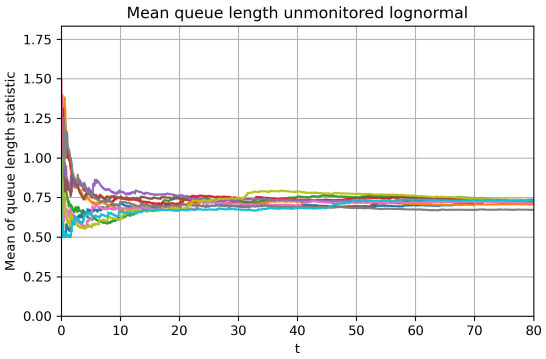


fig 4.1


Mean queue length monitored lognormal
Mean of queue length statistic
t


Mean response time monitoring lognormal
Mean of response time statistic
t


Mean queue length unmonitored lognormal
Mean of queue length statistic
t

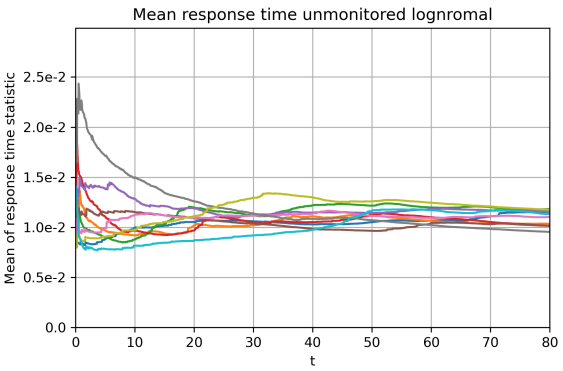

Mean response time unmonitored lognromal
Mean of response time statistic
t

fig 4.2

5 Simulation Length Analysis

The behavior of the system was tested with different simulation lengths. The mean value and the standard deviation of *queue length* and *response time* vary when the simulation length gets increased, up to ~300s. The amount of samples collected in this interval of time seems to be enough to represent the system variability.

Below are reported the *queue length* and *response time* behavior related to the duration of the simulation in monitored mode, with exponential time generation

fig 5.1

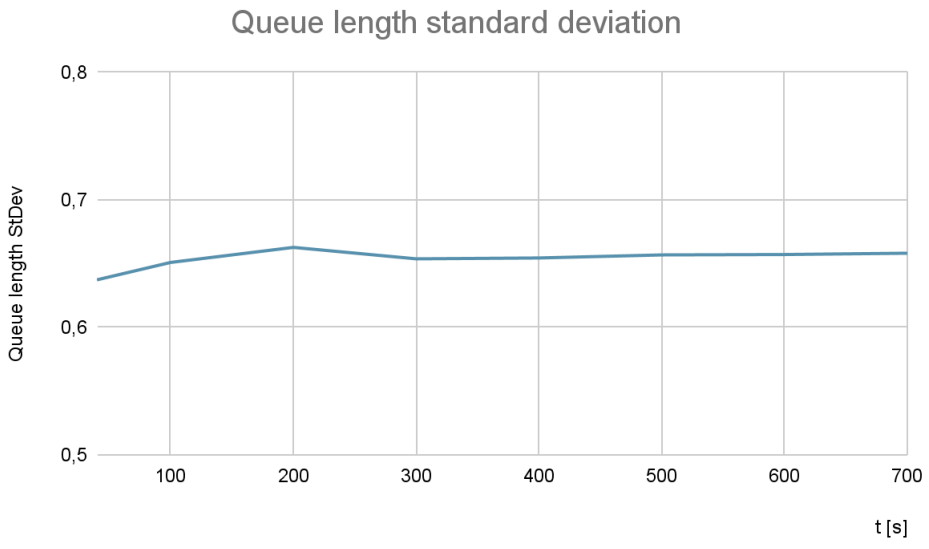


fig 5.2

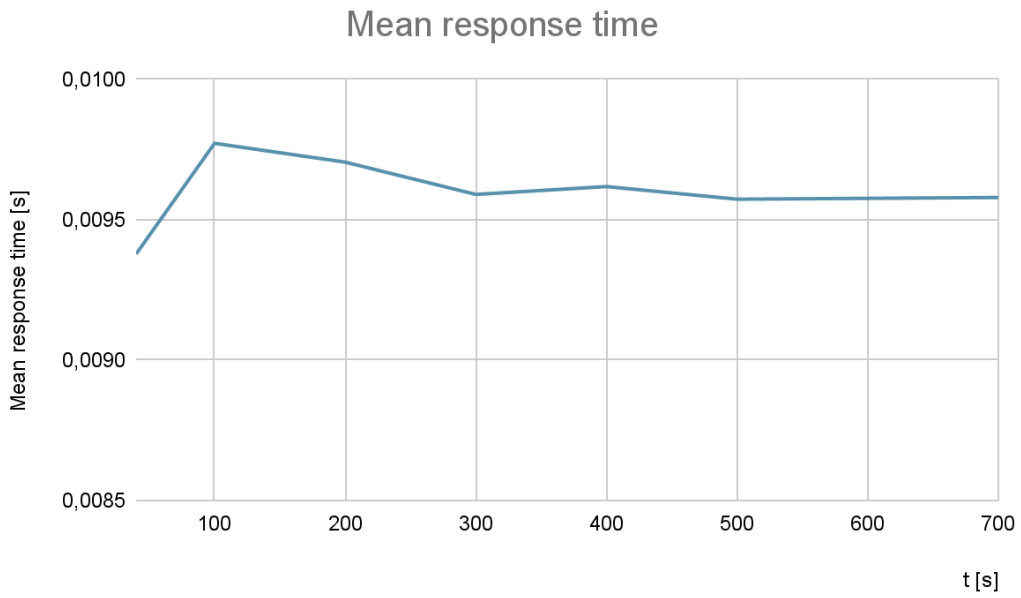

Mean response time
Mean response time [s]
0,0100
0,0095
0,0090
0,0085
100
200
300
400
500
600
700
t [s]

fig 5.3

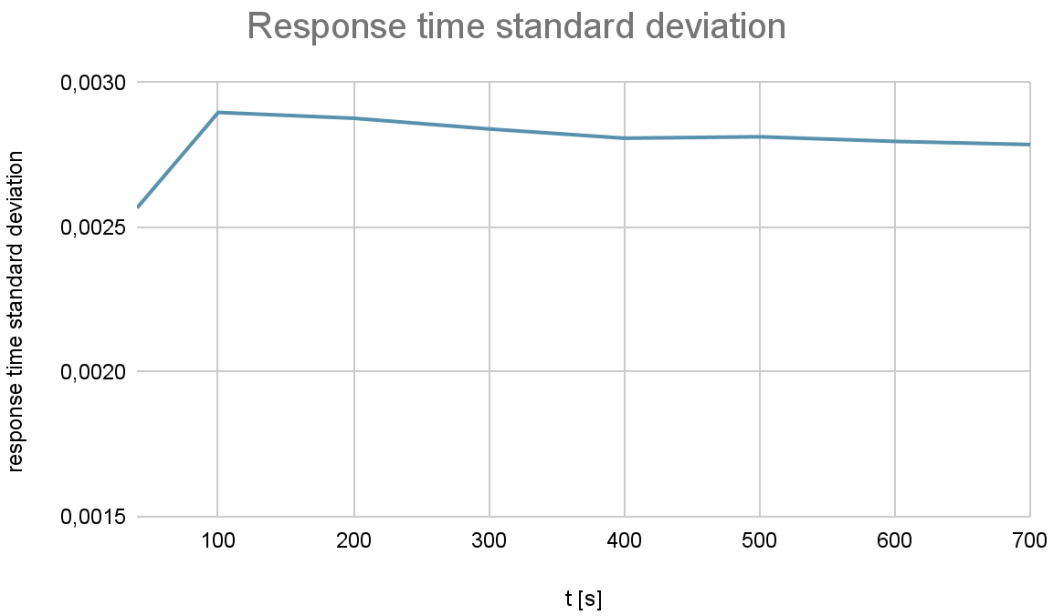

Response time standard deviation
response time standard deviation
0,0030
0,0025
0,0020
0,0015
100
200
300
400
500
600
700
t [s]

fig 5.4

6 Experiments

6.1 Behavior of the response time related to nDL

The response time of the system was tested by using various combinations of malus and *nDL*.

As can be seen in fig 6.1.1 there is a break-even point influenced by both the parameters. If the *X%* parameter gets increased, larger values of nDL are needed in order to obtain better response times in monitored mode rather than unmonitored mode

The behavior of the system, with arbitrary *X%* tends to stabilize from $nDL = 15$. In all further analysis, a value of $nDL = 15$ was chosen in order to avoid misunderstandings of the system performance possibly caused by a low nDL value.

Data was reported with a lognormal distribution of t as there is no substantial difference with the behavior of the system with exponential generation distribution.

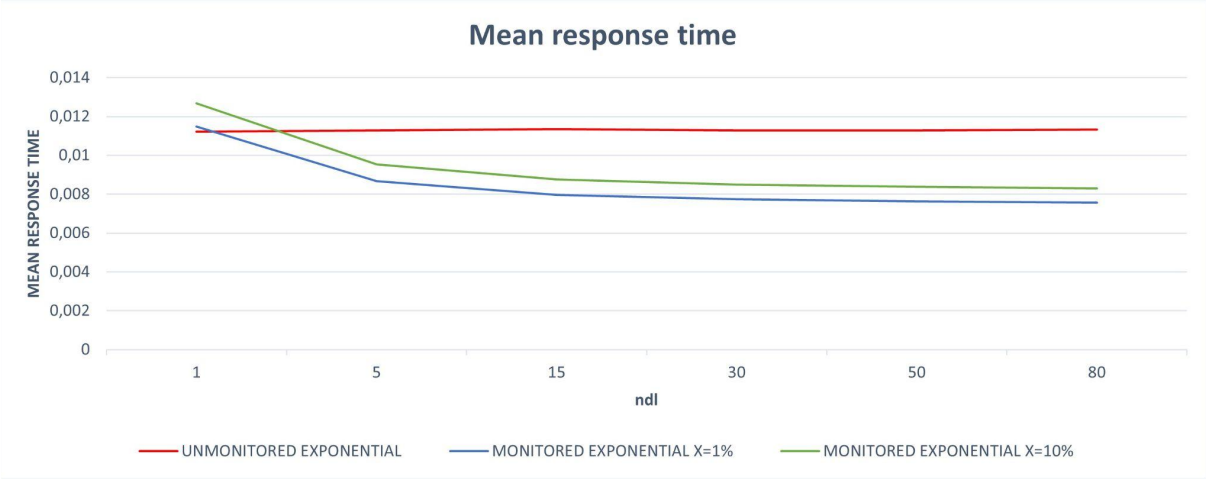


fig 6.1.1

the 99% CI is not shown since it would be impossible to see it

6.2 Relation between service time and waiting time

The relation between E[t_s] and E[W] can be studied by varying the values of *X%*. As can be seen in fig 6.2.1, incrementing *X%* causes both E[t_s] and E[W] to increase. This is an expected result larger values of *X%* causes a decrease in capacity, which in turn causes an increment in E[t_s] and E[W]. The impact of the E[t_s] on E[R], reported as a percentage over the bars in the graph, decreases as the *X%* value increases. This is due to the high increase of E[W], which takes an increasing share of time as the malus increases. The decreasing trend of E[t_s] also causes an increase in $E[N_q]$.

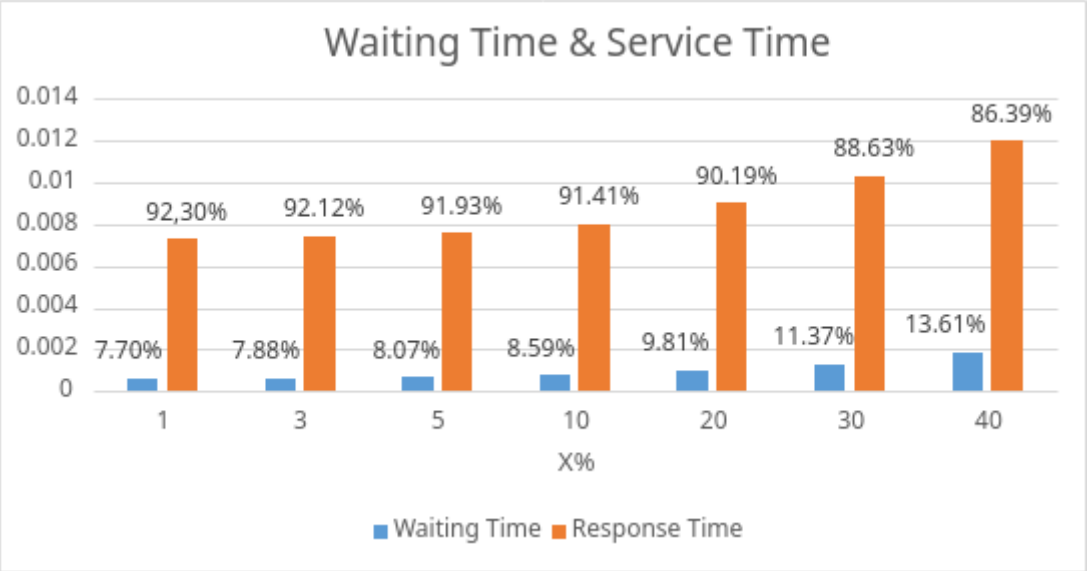


fig 6.2.1

the 99% CI is not shown since it would be impossible to see it

6.3 Analysis varying the malus

As can clearly be seen in fig 6.3.1 and fig 6.3.2 the system response time increases as *X%* increases, as expected. There is a break even point for *X%* that sets the limit where using the monitored version of the system becomes pointless: with a malus greater than the 30% of the transmission capacity, the response time of the monitored system is worse than the one of the unmonitored. These simulations were computed with $nDL = 15$. With a lower value of *nDL*, the limit for the malus is even lower as reported in par 6.1.

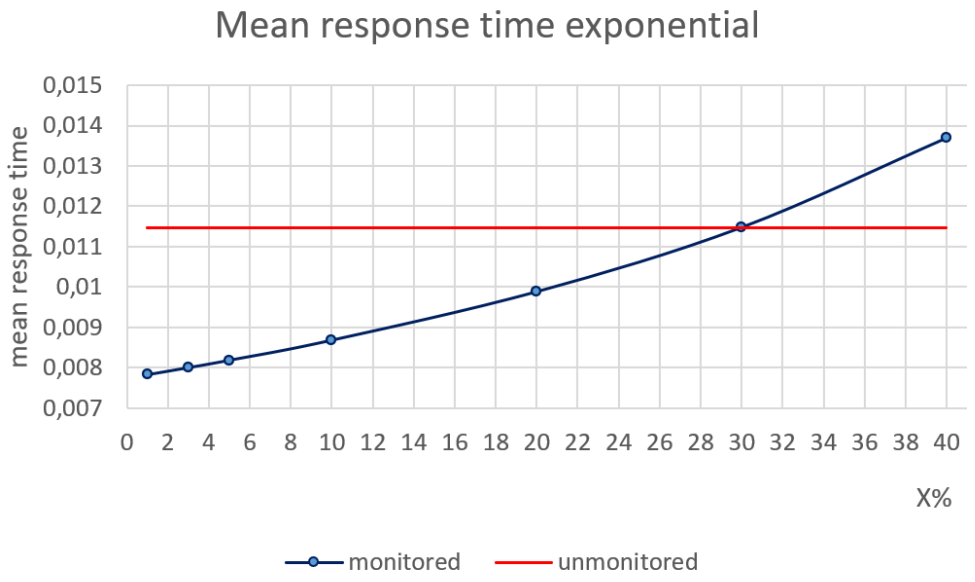


fig 6.3.1 exponential

the 99% CI is not shown since it would be impossible to see it

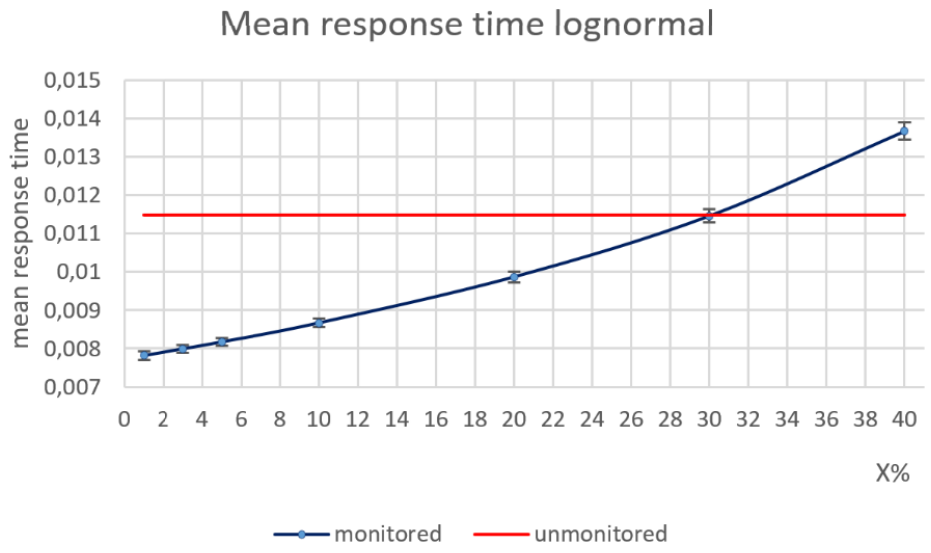


fig 6.3.2 lognormal the 99% CI is shown in the plot

6.4 Analysis of the queue length varying the malus

One of the main aspects to highlight in this analysis is the behavior of $E[N_q]$ related to the increase of *X%*. As expected, the value of $E[N_q]$ increases as the malus increases.

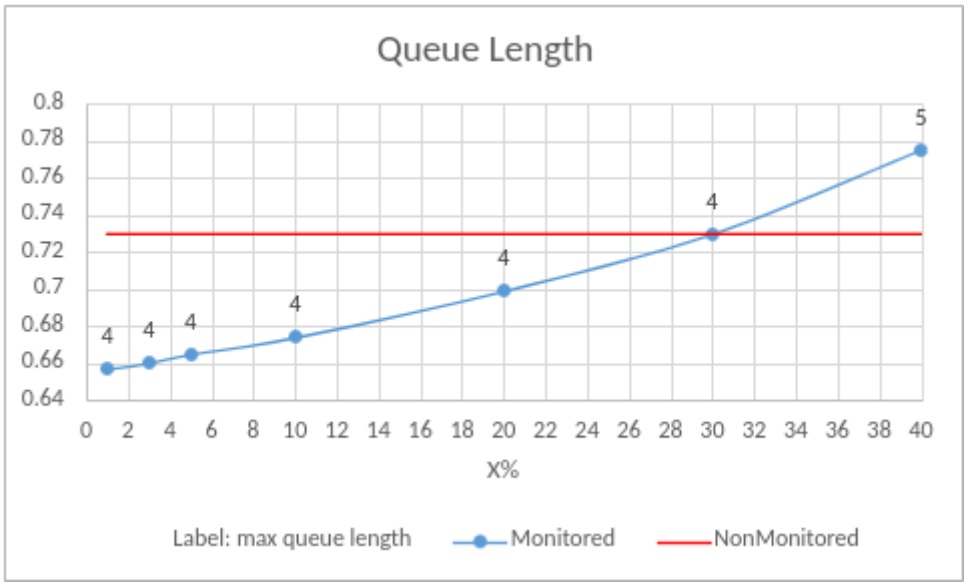


fig 6.4.1

the 99% CI is not shown since it would be impossible to see it

7 Factorial analysis

The $2^k r$ analysis of the system has been computed with three parameters: *X*, *nDL* and the mean value of the exponential distribution used to generate t.

The values used for the computation are shown in the table ahead.

	min	max
X	1%	40%
t mean	0.8	1.5
nDL	2	80

The result is shown in fig 7.1. As expected, and as can be seen in the graph, t mean is negligible, while X and nDL are the main parameters influencing the results. Moreover, X is clearly the most important one, as well as the easiest to work on.

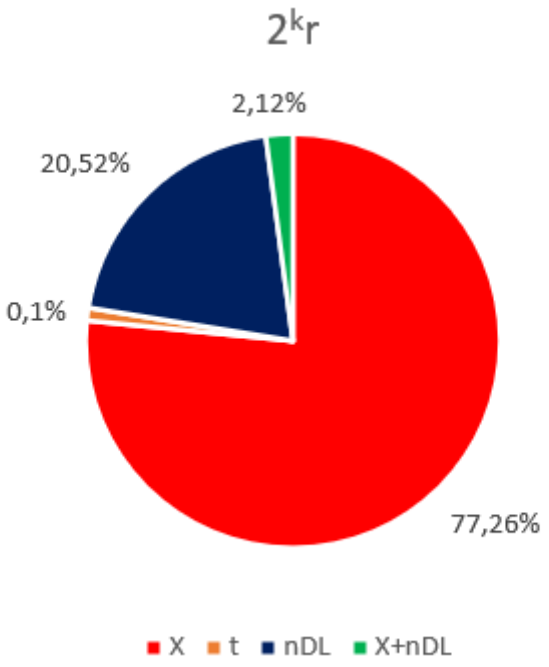


fig 7.1

8 Different interpretation

In this part of the documentation, another possible interpretation of the *malus* and of the overall monitored mode is discussed. In this version, there isn't a constant *X%* malus on the capacity, but instead an *X* seconds monitoring time (malus) every *m* seconds, where *m* is the monitoring period of the system.

In a nutshell: the system monitors the capacity of all the *DLs* every *m* seconds, and the monitoring takes *X* seconds to complete, where $X = k + nDL * c\,;\ k, c \in \mathbb{R}^{+}$.

The system obviously works only if $X < m$.

A visual representation of the system behavior is shown in fig 8.1

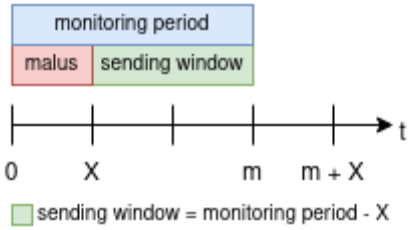


fig 8.1

8.1 Analysis varying the monitoring period

As can be seen in fig 8.1.1, if *m* is too little, the system has too little time to send packets in each period, so most of the monitoring period will be composed of malus instead of actual message forwarding.

On the contrary, if m is too large, the system gets no advantage in monitoring the *DLs* so the behavior becomes basically the same as the unmonitored mode. This can be seen best if m is set equal to the all simulation time. The behavior of such a system can be seen in fig 8.1.1 and fig 8.1.2. The following graphs are based on a simulated system with exponential generation distribution since there are no remarkable differences between a lognormal and an exponential distribution in this case.

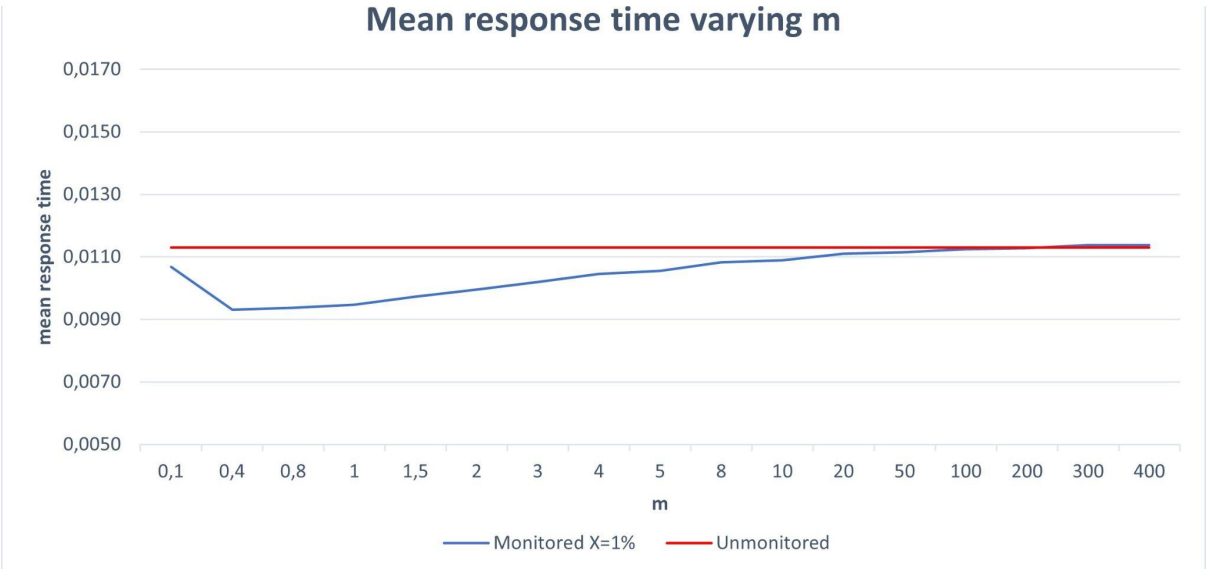


fig 8.1.1

the 99% CI is not shown since it would be impossible to see it

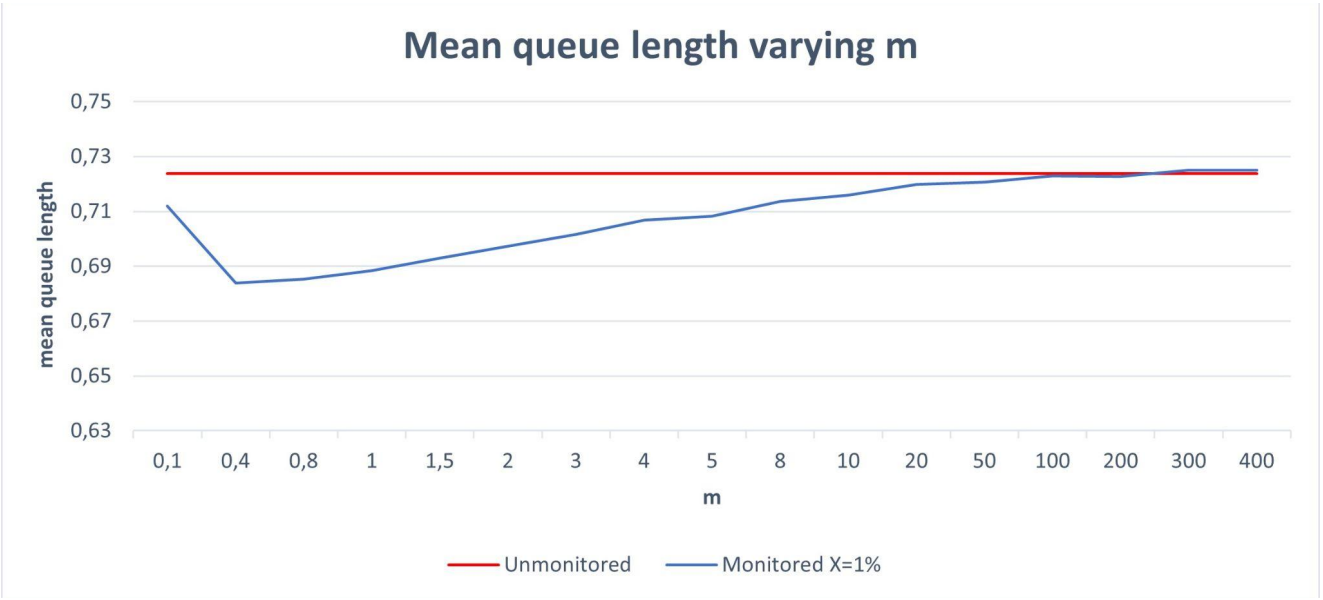


fig 8.1.2

the 99% CI is not shown since it would be impossible to see it

8.2 Analysis varying the malus duration

The behavior has been tested also for different values of *X*. As can be seen in fig 8.2.1, when $m = 1$, the break-even point sits around $X = 0.05$. After this point the system performance becomes worse than the unmonitored mode's performance, due to the noticeable impact of the malus. If a different m value is used, as shown in fig 8.2.2, the system tolerates an higher malus, but loses some performance due to the variation of the *DLs'* capacity during the sending window.

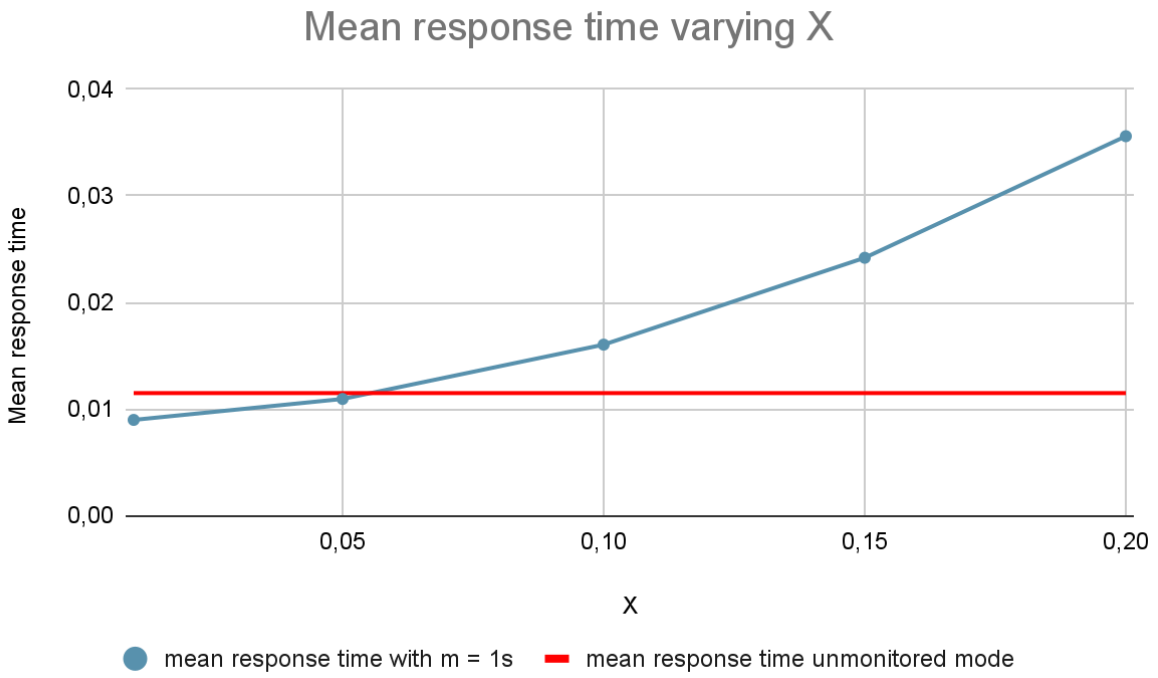


fig 8.2.1

the 99% CI is not shown since it would be impossible to see it

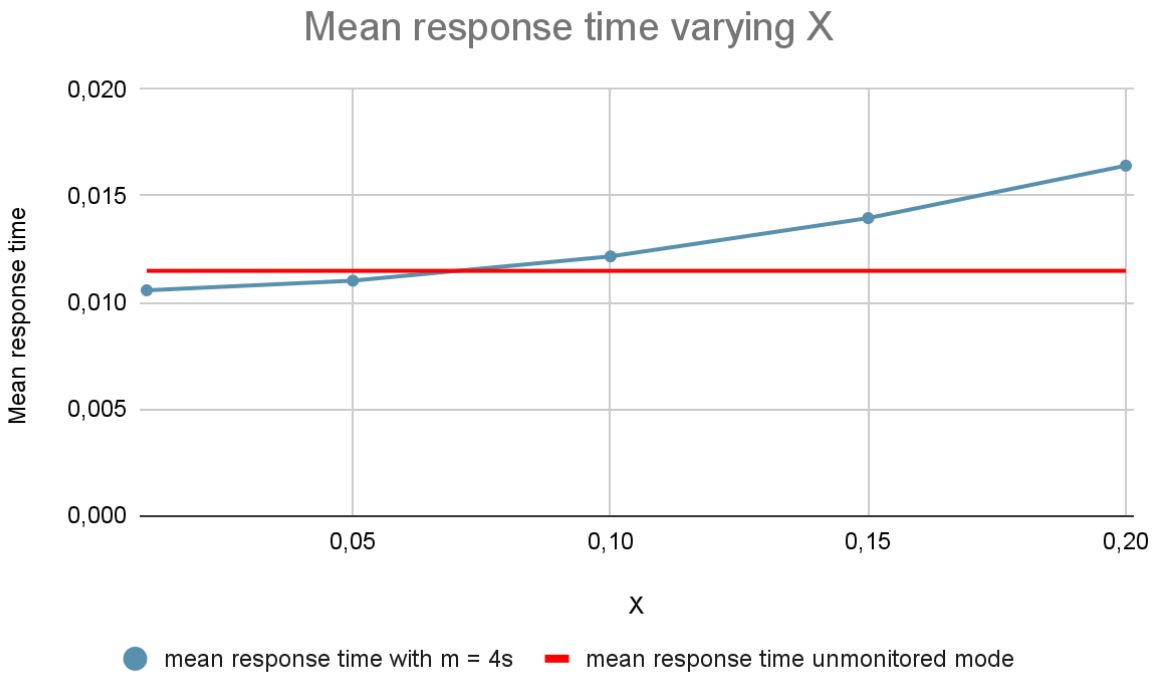


fig 8.2.2

the 99% CI is not shown since it would be impossible to see it

9 Conclusions

We can conclude that the choice between the two implementations is mostly influenced by the value of *nDL* and *X%*.

To achieve better performance in monitored mode, compared to the unmonitored one, given that $nDL \geq 15$, a malus lower than the 30% of the *DL* capacity is needed. Otherwise the system wastes more capacity monitoring than the one it gains by selecting the optimal *DL*. With lower values of *nDL*, for instance with values lower than 3, there's no gain at all in using a monitored system, even with a malus of only 10% of the transmission capacity.

The simulated system was built using the infinite queue abstraction, but, with a malus below 30%, the queue stabilizes around a size of 4 packets.

In a real implementation of such a system, *nDL* won't probably vary that much. As such, the best way to achieve better performance would be trying to reduce the capacity occupancy of the monitoring stream of data as much as possible, either by reducing the size of packets as much as possible or by finding a less capacity consuming monitoring algorithm.