Group 1

Vacation Server

Performance Evaluation Project

Group 1
Francesca Righetti, Giulio Micheloni, Sara Egidi

1. Introduction

The aim of this documentation is to report in detail the study of a system composed by a queue served by a server, which goes on vacation whenever the queue is empty.
To describe the behaviour of this system, we could think of a Wi-Fi network in which the queue is composed by packets, or a post office in which the queue is composed by people waiting. The server can be seen as “vacation” router for the former and a “vacation” employee for the latter. The functioning of a vacation server is to do different things from serving for a random time whenever it finds the queue empty. If returning from a vacation it finds the queue empty again it immediately starts a new vacation.
In this documentation the system will be referred as a general system composed of a queue in which there are jobs are served by a vacation server.
The workload of this system is represented by the interarrival time of the jobs and their service time. The metrics in which the analysis is focused are the mean number of jobs in the system, the response time and the *emptying time*, i.e. the time it takes for the queue to become empty.
Studying how the model reacts to different workloads, we expect to find a better introspection of the real system.
In order to collect statistics for the performance analysis, the system behaviour has to be simulated. For this purpose OMNeT++ will be used.

The rest of the documentation is organized as follows:

In section 2 the system will be described in detail, whereof implementation is presented in section 3. For what concerns performance analysis and experimental design they will be both illustrated in section 4. Finally in section 5 results are shown with their related comments. The final conclusions are presented at the end of the project.

2. System description

To represent the system, which is basically composed by a client that makes requests to a server, it has been decided to subdivide these two in two modules.

The client is a module which sends jobs to the server module. The last one, structured to have a queue inside, has to process jobs unless the queue is empty, in which case it will go idle for a random “vacation time” (with “idle” is intended that the server stops processing jobs, but the queue is accumulating incoming jobs).

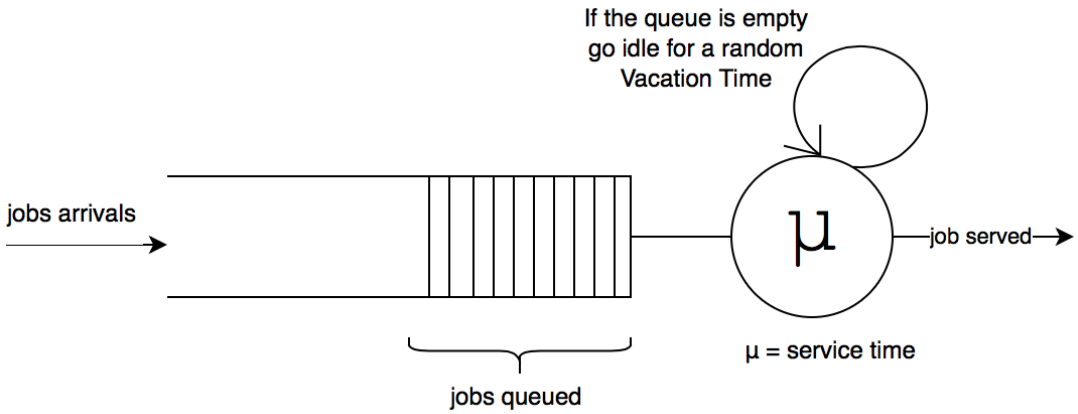

Figure 2.0-1 - Server module

The client sends requests at certain rate (interarrival time), so if the server is busy, jobs will queue up. Since this study is focused on the impact of a vacation in a simple M/M/1 system, an infinite queue was modelled. The probability of job loss does not concern this case of study.

In the Figure 2.2 it can be observed that the server does not reply to the client for its requests. This assumption seldom holds in real client-server systems; this behaviour was not modelled since the analysis of the system focuses on the server dynamics, so it's not meaningful for the goal.

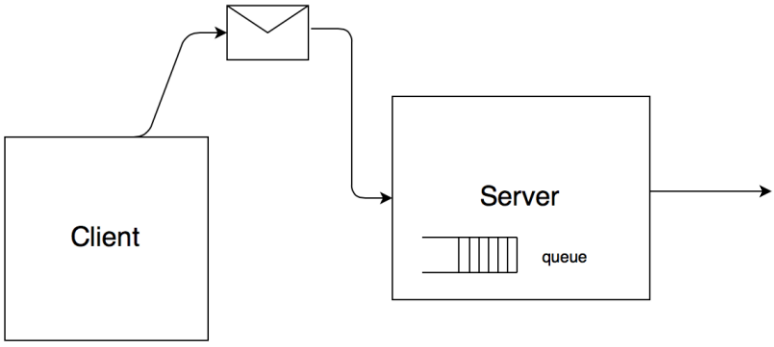

Figure 2.2 – Model of the system

Since this type of system behaviour, it may be interesting see how it works under some particular cases:

- Considerable difference between interarrival time and service time in the way that the first one is greater than the other, so the server will be able to process all the jobs in the queue and go on vacation when this is empty. In this case what is interesting to analyse is how the vacation time affects the overall system.
- Interarrival time and service time are almost the same, so here it is interesting see where saturation occurs, i.e. find which scenario brings the system in an unsustainable situation.
- With a fixed vacation time and increasing workload, but holding service time below interarrival time; in this situation it is interesting to see in which way the workload affects the system, in respect to other cases.

Even if in real-life infinite queues situations scarcely occur, those can be simulated with some proper simplifications.

To make this analysis closer to reality, the system was compared to a real world system so that future considerations could be applied to the latter.

The system chosen for this purpose was a bank with its teller and people waiting to be served. Intuitively the teller will represent the server and the clients the jobs. There will be a queue of people and if the teller is not able to serve every person in a reasonable time, like office time, the bank system will not be considered efficient.

Referring to the scenarios described above some considerations can be made regarding this sample system.

Intuitively the director of the bank should choose a proper time of coffee break for his tellers as a function of the usual workload of the desk to avoid unpleasant long wait for the clients.

3. Implementation

This section will cover the implementation process of the studied system.

3.1 - Outline of the system

Before modelling the system its theoretical behaviour was evaluated.

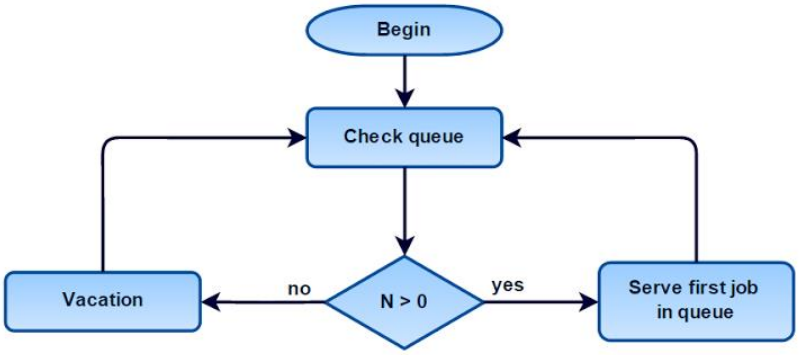

Figure 3.1 – Basic behaviour of the system

Once it was defined, a possible implementation on Omnet++ was conceived.

Since the system is composed of a client and a server, they will be implemented as two cSimpleModule. The first one is appointed to spawn jobs, and the second is in charge of managing the actual behaviour of the system.

The client module implementation is straightforward: since its only aim it is to produce jobs based on a delay supplied by the configuration file (either constant or exponential), it just has to loop on itself for the duration of the delay, and then send out a message.

The server module is a bit more complex. Before going on we should define properly how jobs are handled. The jobs are sent out by the client as simple cMessage(s), each message has not information by itself. Upon reception the messages will be added to the queue. The data structure chosen for the queue is a std queue, since pop() and push() operation

allow to easily add and remove the jobs from the queue. Each arrival will queue the job in the structure, and saved as timestamp for convenience.

To simulate delays on Omnet++ a function schedules a self-message and, upon any received message, the server will switch between the three cases: vacation has expired, service has expired or a new job has arrived. Depending on the type of the arrived message the server will behave accordingly.
The system is now defined; hence the previous flowchart can be extended as follows

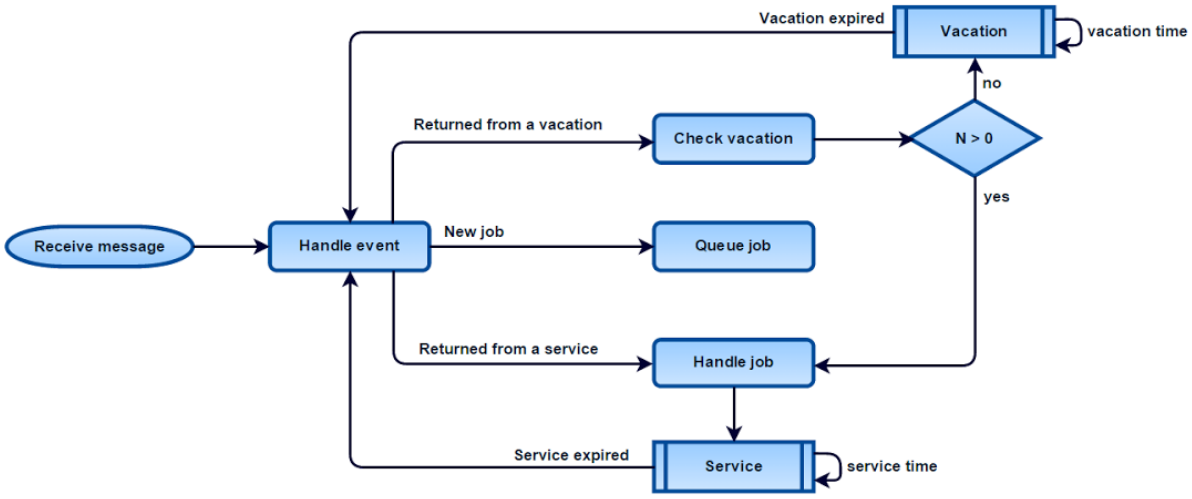

Figure 3.2 – Flowchart for the server

Further details to be found on the code comments.

3.2 - Statistics

Omnet++ gives the opportunity to collect plenty of statistics out of a simulation; this allows to analyse the system more in detail.

For this model the collected statistic are:

Statistics	Description	Collection method
Number of jobs	Number of jobs in the system	The number is collected anytime a job is pushed or popped from the queue
Weighted number of jobs	Number of jobs in the system in respect of time	Every event (push or pop) will result in a product of the interevent time and the number of jobs in the queue. These values will be summed up during the simulation and divided by the total simulation time at the end of such $\sum \frac{\Delta t * \#jobs}{simulation\ time}$
Wait in queue	Time that each job has to wait in the queue before being served	Every time a job is popped out from the queue its arrival time (which is stored in the queue) is subtracted from the current time

Response time	Time that each job takes to pass through the system	As the server comes back from a service the response time is computed as the current time minus the waited time for the job
Idle time	Total time the server spent on vacation	Value collected as the sum of all the vacation times
Job during vacation	Number of jobs arrived during a vacation	Collected as the size of the queue as the server comes back from a vacation
Emptying time	Time that took to the server to dispose all the queued jobs collected while on vacation plus the ones that arrived during the process	When the queue is empty, the time in which the server comes back from a vacation is subtracted to the current time

For what concerns the correctness of the application, it has been thoroughly debugged in small steps through some kind of modular debugging: each time a new functionality was added its consistency was tested scrupulously.

The result of this careful process is that the model is fully represented in any detail by the implementation, the system responds as expected by the analytical model.

3.3 - Configuration file

The configuration file allows turning the “knobs” of the system without changing the model.

For the purpose of this system, the factors that will be changing are:

- Interarrival time mean
- Service time mean
- Vacation time mean

which together form the scenario for the simulation.

The first two means can be either set as constant or exponentially distributed, the vacation mean is always set as exponential.

To change the workload between one simulation and the other, just the IAT and ST mean values are to be changed in the configuration file

4. Performance analysis

This section is dedicated to the analysis of the behaviour of the system

4.1 - Performance indices

In this section we will justify the choice of the metrics on which we focused.

Examining how the system works, it can be observed that it has a periodic behaviour in almost all the cases: if the queue is empty, there will be a vacation that will induce to an *accumulation* phase, where the arrival jobs are queued up. When it returns from the vacation, it will find a number of jobs in queue depending on the ratio between *vacation time* (VT) and *inter-arrival time* (IAT).

At this point the server starts to serve the jobs until all of them are served and the queue is empty, then another vacation occurs. When also this vacation will dispose all of the queued jobs, there will be another vacation, and so on and so forth.

This behaviour can be represented in figure 4.1 below

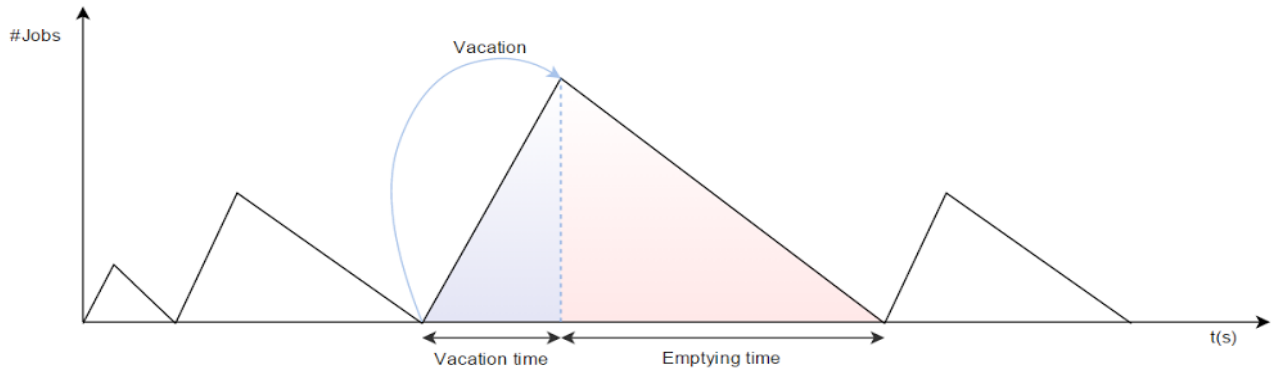

Figure 4.1 – Number of jobs in the system

From this analysis it results that one of the things that characterizes the systems is the *emptying time*, i.e. the time needed for the server to empty the queue.

Obviously this index may depend by some factors, which should be defined, together with their relation.

Intuitively the *emptying time* (ET) is the sum of *service times* (ST) for each job in the queue:

$$ET = \sum_{i=1}^{\#TotJobs} ST_i \tag{4.1}$$

Thanks to the additivity of mean values:

$$E[ET] = \sum_{i=1}^{\#TotJobs} E[ST_i] = E[ST] * \#TotJobs \tag{4.2}$$

So we need to know the total number of arrival jobs during vacation plus the ones arrived during the emptying phase.

From figure 4.1 it can be noticed that the average number of jobs that arrive during a vacation is:

$$E[jobs\ during\ vacation] = \frac{E[VT]}{E[IAT]} \tag{4.3}$$

But during the emptying time other jobs may arrive, on average their number is:

$$E[jobs\ during\ disposal] = \frac{E[ET]}{E[IAT]} \tag{4.4}$$

To simplify the notation E[ ] can be omitted and implied in all the values.

Putting all together we obtain:

$$ET = ST * \frac{ET + VT}{IAT} \tag{4.5}$$

After some algebra:

$$ET = \frac{VT * ST}{IAT - ST} \tag{4.6}$$

This theoretical formula will be compared with experimental results found in section [5]. The other two performance indices on which the study focused are: *average number of job* in the system, and *average response time*. Both of them can tell us directly how the system responds to different workloads, for instance when the system reaches an high level of congestion.

4.2 - Experimental Design

Moving forward on the experimental design, the different tests that have been driven will be explained.
Each test has the aim to show how the system responds to different workloads, changing certain factors and keeping the others constant.
As described above, the three factors that can be exploited are: Service Time (ST from now on), Inter-Arrival Time (IAT) and Vacation Time (VT), all expressed in seconds.

4.2.1 - Definition of scenarios

Two main scenarios were taken into consideration for the system.

- The **first** scenario is the constant one, where the IAT and ST are both constant during the simulation; vacation time on the other hand is a random variable exponentially distributed with a defined mean value.
  This scenario allows to observe the system in a deterministic way, where the only source of randomness is the vacation time. Thanks to the fact that just the VT is a RV, the analytical results can be confirmed by the data.

- The **second** scenario is the exponential one, where all three factors are random variables exponentially distributed, with a defined mean value.
  In this case more randomness is expected to result in the system's outputs, because this last depends from how the random values of each factors are combined together.

4.2.2 Definition of workload sets

To see how the performance metrics change depending on some factor, different sets of workloads were defined.

All the tests that are described below were done for each of two scenarios above.

- In the **first** case the study is inquiring into a possible relationship between the service time and the performance indices. For this purpose the inter-arrival time and vacation time values are steady (or mean values for the second scenario), and the only value that is incrementing is the service time one.
  Changing just the service time it's possible to obtain, in a single experiment, different responses for different workloads, from low level of congestion to high level of congestion.
  In detail the values used are:
  - IAT = 1 s
  - VT = 10 s
  - ST = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.98] s

  Vacation time value was chosen way higher than ST and IAT because in this way the analysis is more meaningful. It's clear that for a small value of it the system does not affected remarkably.

- The **second** case of study will investigate on how the change of vacation time influences the system. For this scope inter-arrival time and service time are kept constant, and this time just the value of vacation time will change.
  In detail the values are:
  - IAT = 1 s
  - ST = 0.9 s
  - VT = [0.5, 0.8, 1, 1.3, 1.5, 4, 8, 9.5, 9.8, 10] s

The value of service time was chosen to be 0.9 because in this way the system will have to deal with a high workload, and since this is one of the worst case scenarios the simulation is expected to give more relevant data to infer on.

The value of the vacation time is not spread evenly. This was done to obtain more information in the interesting areas, i.e. when the value of vacation time is comparable to service time and inter-arrival time or when this VT is much greater than the other values (VT about 10 times IAT, ST).

In order to obtain different independent samples of performance indexes, we collect the values from 10 different seeds.

5. Results and conclusions

After all the data was collected and processed, some results were observed.

5.1 Experimental results – First case of study

5.1.1 Constant scenario

In respect to the first scenario, one interesting result can be observed by the reaction of the system to an increase of the service time [Figure 5.1.1 – 5.1.2]

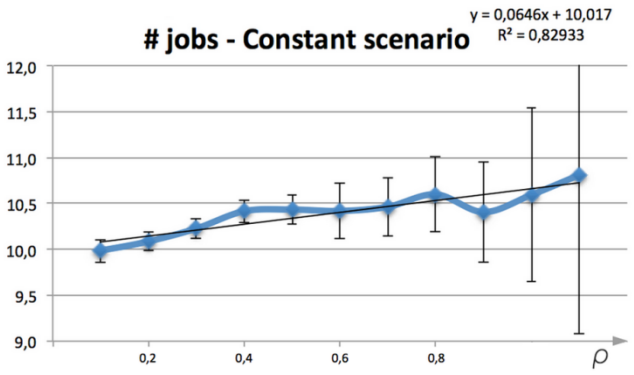

Figure 5.1.1 – #jobs for constant scenario

Inferring on the figure above it can be noticed that increasing the ratio between ST and IAT the number of jobs in the system (as well as the response time) shows a rather linear trend.

This behaviour confirms the intuition inferred during the analytical analysis [Figure 5.1.2], i.e. that in this case the system is to be considered deterministic, and thereby it has a periodic tendency

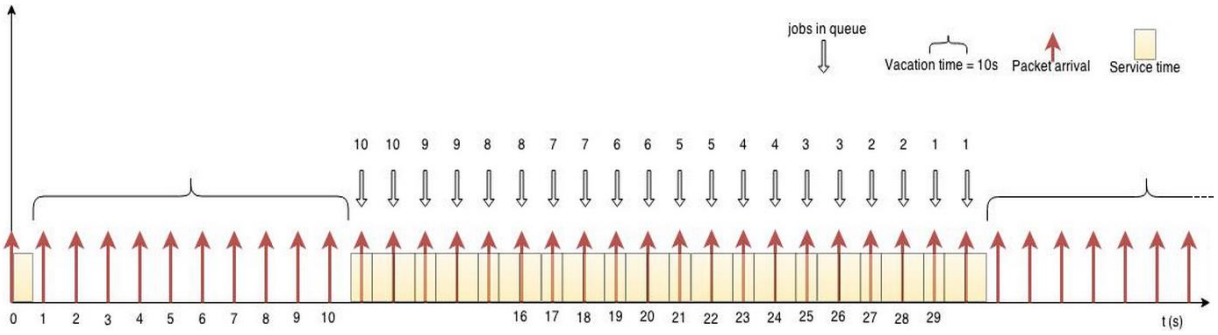

Figure 5.1.2 – Representation the system behaviour

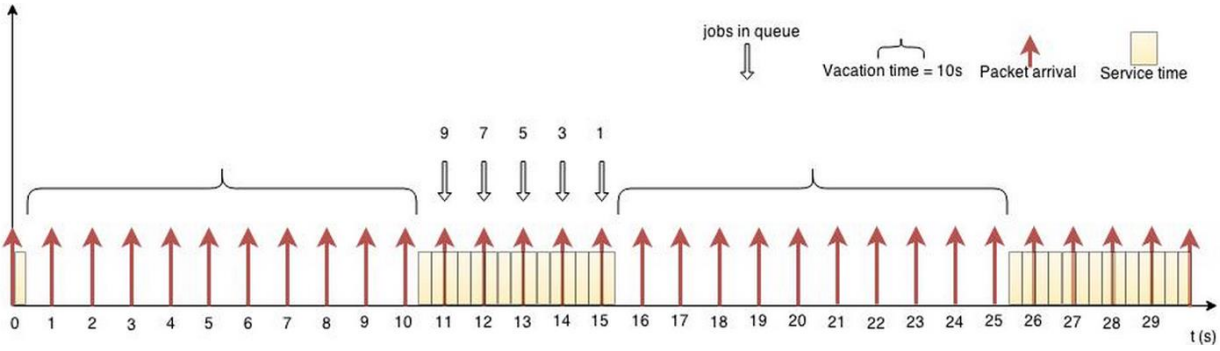

Figure 5.1.3 - Representation the system behaviour (2)

Thoroughly observing Figure 5.1.2 and Figure 5.1.3 it can be seen that the linear trend is justified by the fact that less is the ST, greater is the speed at which the queue empties. That means that the server will go in vacation frequently.

On the other hand as the service time grows, for the server it will take more time to empty the queue each time it comes back from a vacation. In this case the server won't start as many vacations as the previous case.

The behaviour explained above is confirmed by the simulation graphs as shown in Figure 5.1.4 and 5.1.5.

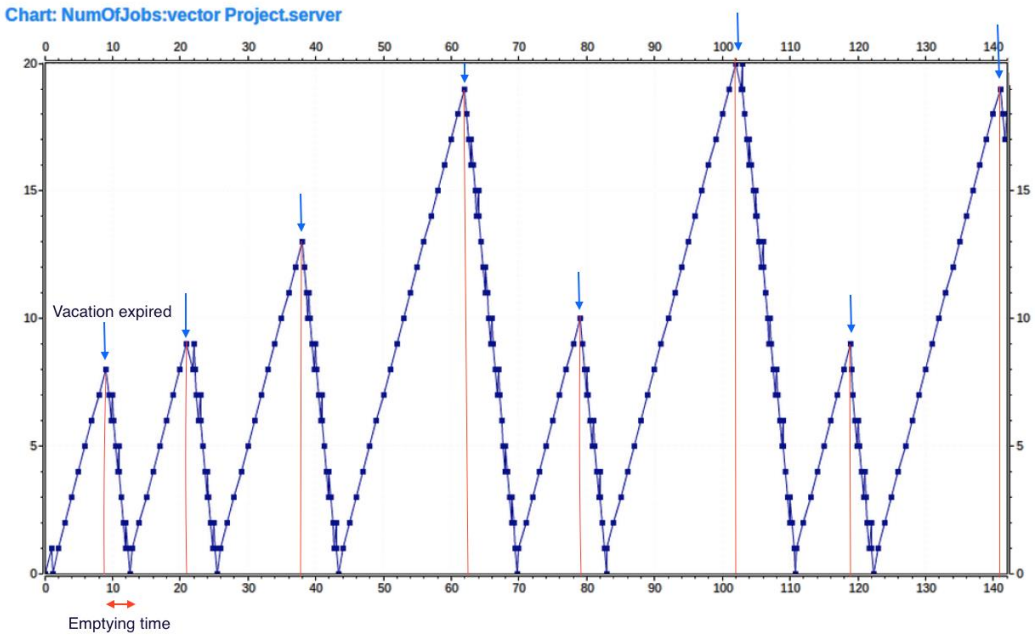

Figure 5.1.4 - IAT = 1; ST = 0.3; VT = 10 Constant scenario

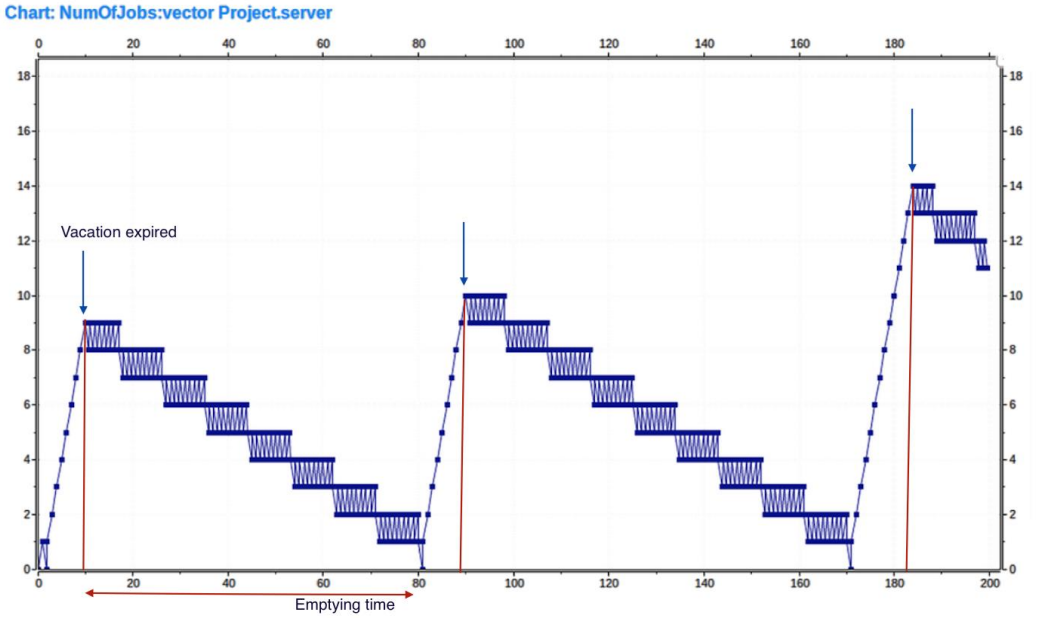

Figure 5.1.5 – IAT = 1; ST = 0.9; VT = 10 Constant scenario

From experimental results it can be perceived that the mean number of jobs in the system floats around the same value. This result is reasoned by the fact that those two phenomena perfectly balance each other.

As expected, the response time follows the tendency of the mean number jobs (as Little's theorem suggests) [Figure 5.1.6].

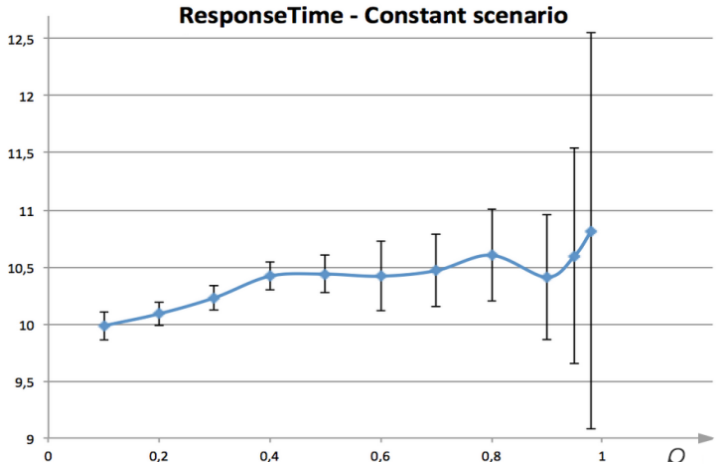

Figure 5.1.6 – Response time constant scenario

For what concerns the emptying time, the theoretical formula (4.6) gives a hint of the tendency it will follow. In fact the trend is expressed by the formula

$$\frac{ST}{1-ST} \tag{5.1}$$

for ST going to 1, for which it tends to ∞.

This behaviour is further confirmed by experimental data reported below in [Figure 5.1.7]

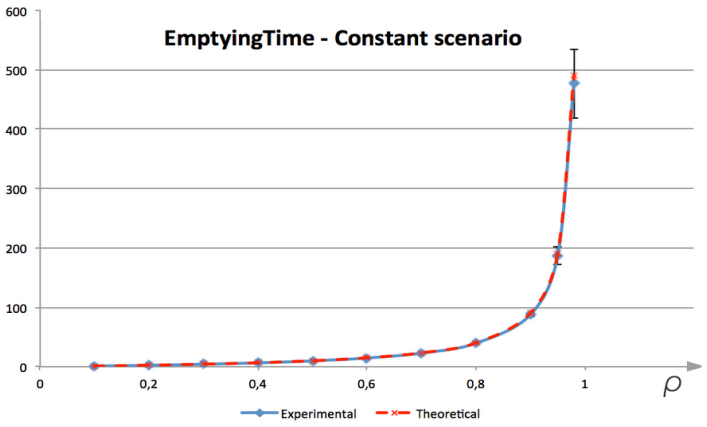

Figure 5.1.7 – Emptying time constant scenario

This tendency can be further clarified by splitting it by its two contributors: the former is the one just described, i.e. to a greater ST corresponds a greater emptying time; the latter is a factor derived from the formula (4.6) i.e. the difference between IAT and ST.

This last factor results in what can be seen as a *time advantage* that will sum up for each processed job [Figure 5.1.8].

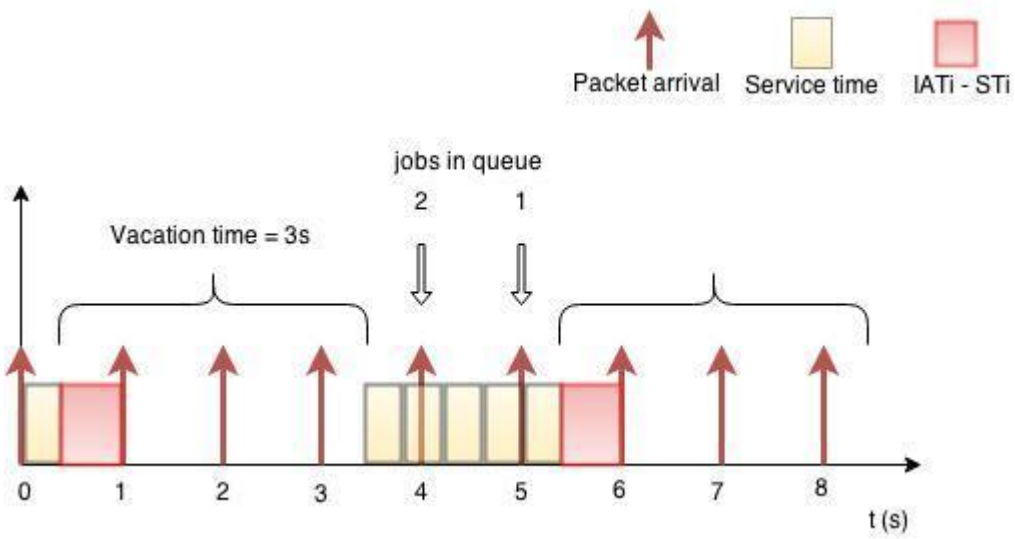

Figure 5.1.8 – Jobs "advantage"

As the difference between IAT and ST gets smaller, every new arrival results in a longer disposal of the queue.

These two components together contribute to the divergence of the emptying time.

5.1.2 Exponential case

The second scenario of interest consists of IAT and ST exponentially distributed.
Looking at the graphs of the increasing ρ [Figure 5.1.6 – 5.1.9] the difference between the number of jobs of the two scenarios is evident.

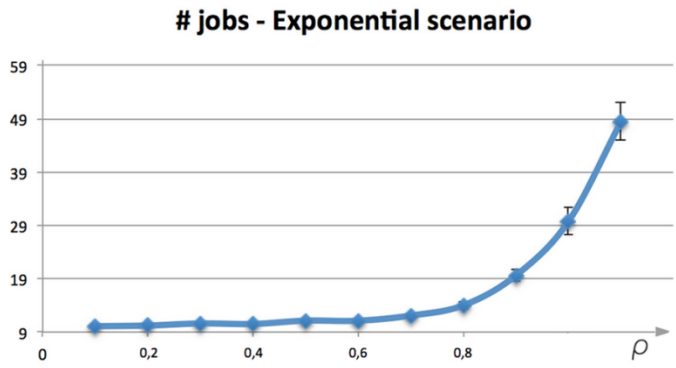

Figure 5.1.9 - #jobs for exponential scenario

The tendency of the graph [Figure 5.1.9] can be justified analysing how probabilities are spread in an exponential distribution.
From theory is known that the PDF that characterizes the exponential distribution is f(t) = $\lambda e^{-\lambda t}$, and that $E[x] = \frac{1}{\lambda}$. Increasing the latter will result in a descending value of λ, which as a consequence will enlarge the exponential tail making it heavier. This means that the probabilities to have large values will increase.
This is because in the exponential scenario the peculiar *periodic* behaviour of the system is annihilated, due to the two RVs introduced.
This last phenomena is more evident as much as the variance of the two RVs increases (see CIs in Section 5.2).

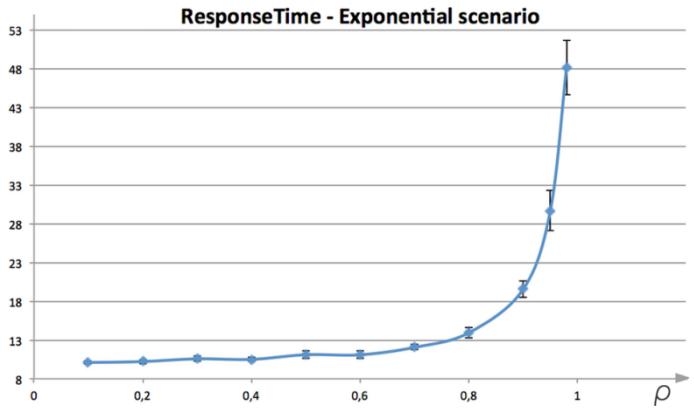

Figure 5.1.10 – Response time exponential scenario

As expected, the response time follows the number of jobs trend [Figure 5.1.10].

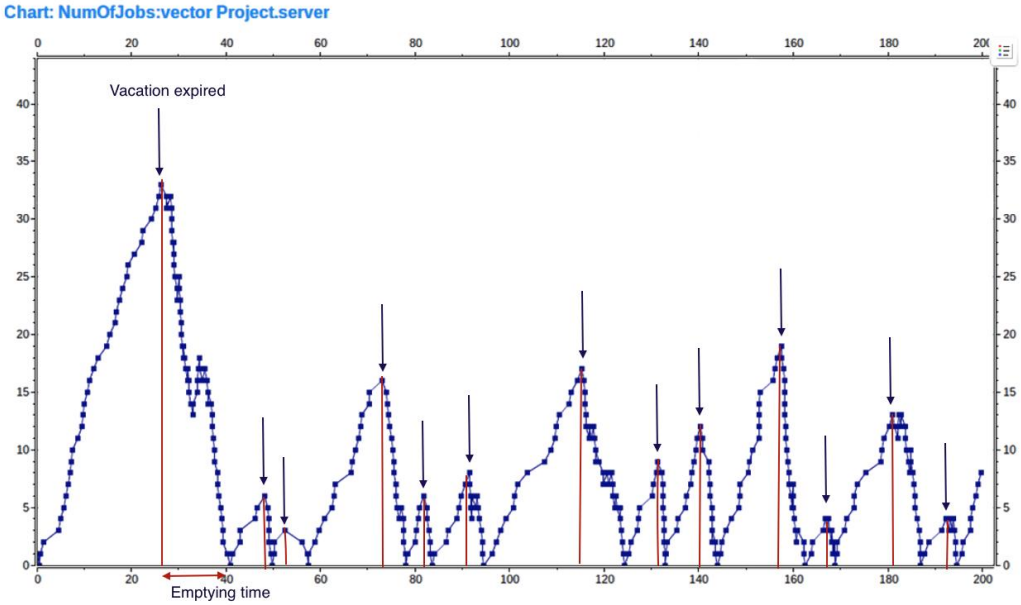

Figure 5.1.11 – IAT = 1; ST = 0.3; VT = 10 Exponential scenario

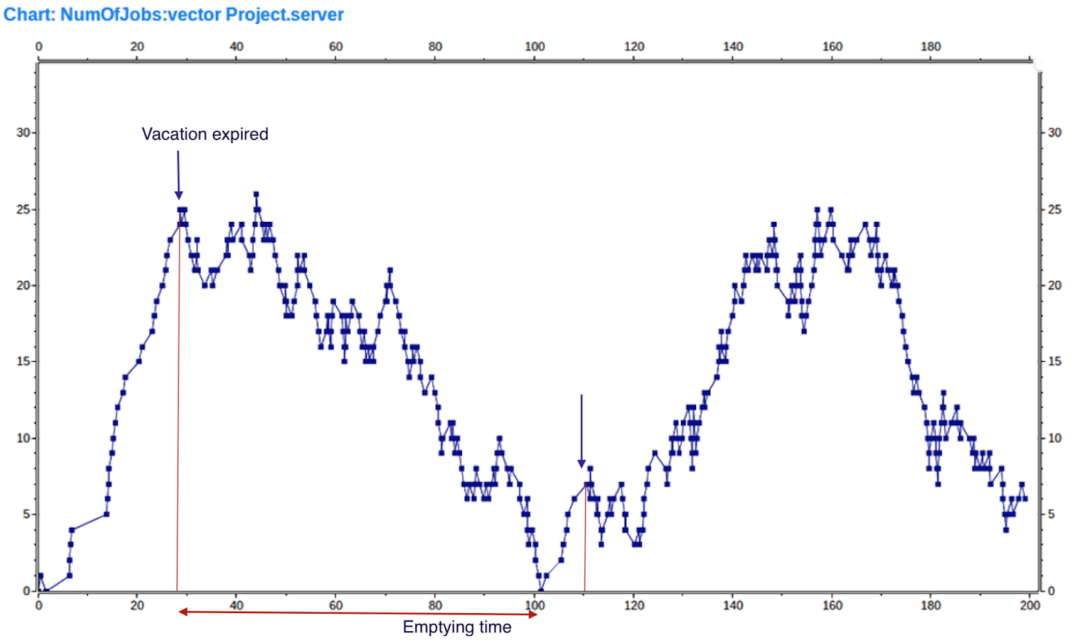

Figure 5.1.12 – IAT = 1; ST = 0.9; VT = 10 Exponential scenario

As it is possible to notice from pictures [Figures 5.1.11, Figure 5.1.12], they reflect the phenomena described in the constant scenario, i.e. as ST increases, the server goes less time in vacation but it takes more time to dispose the queue.

In addition thoroughly observing both graphs the fact that the periodicity is lost is confirmed.

As for the emptying time its tendency is akin to the first scenario, the only thing that is introduced by the two RVs is an increase of the variance of the sample and thus confidence interval

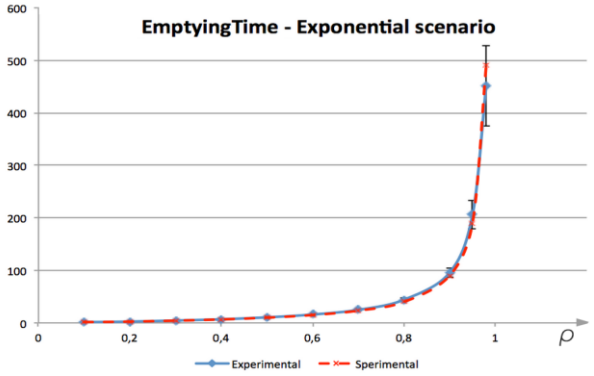

Figure 5.1.13 – Emptying time exponential scenario

5.2 Experimental results – Second case of study

Moving on the second case of study, this time the vacation is increasing and the other parameters are kept constant.

Starting from the constant scenario, in [Figure 5.2.1 – 5.2.2 – 5.2.3] the charts of the usual three performance indices are reported.

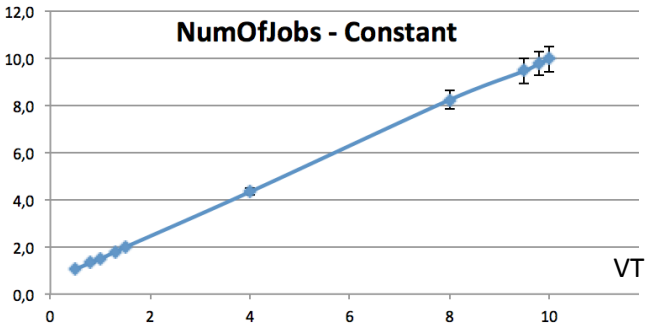

Figure 5.2.1 – Number of jobs constant scenario

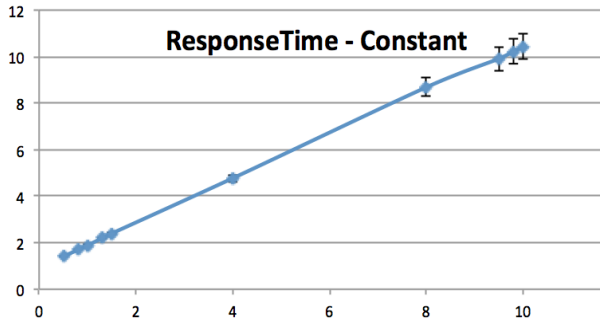

Figure 5.2.2 – Response time constant scenario

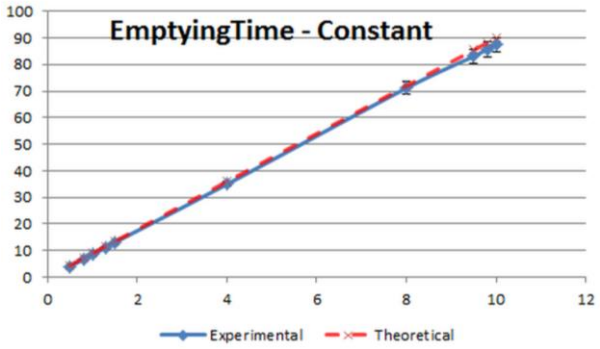

Figure 5.2.3 – Emptying time constant scenario

It can be noticed that all of three indices are linear with the respect of the vacation time. This fact can be justified from the intuition that in a deterministic system like the one analysed the only factor that makes jobs queue up is the vacation.

As the mean time of vacation increases there will be more jobs accumulating in the queue, consequently the emptying stage will be longer.

This will also result in the fact that in relation to the time of examination the queue will hold jobs for a longer period of time.
Looking at the graph in [Figure 5.2.3] of the emptying time, the linearity is confirmed both from the above reason and the theoretical formula (4.6).

For the exponential scenario the trend stays the same except for the CIs that, clearly, get wider. This is again due to the variance of the sample and the randomness introduced in the system.

The performance indices are presented in [Figure 5.2.4 – 5.2.5 – 5.2.6].

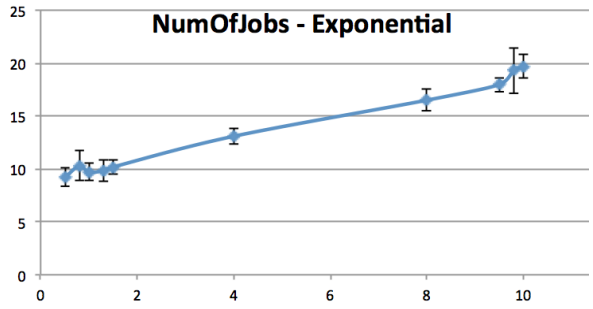

Figure 5.2.4 – Number of jobs exponential scenario

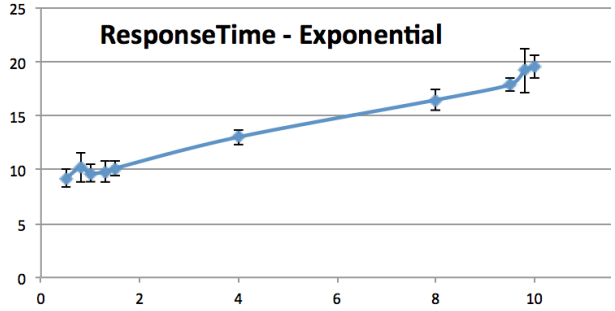

Figure 5.2.5 – Response time exponential scenario

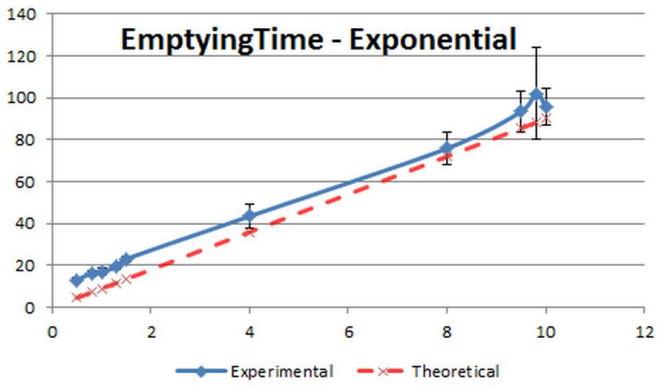

Figure 5.2.6– Emptying time exponential scenario

The fact that all performance indices remain linear in both scenarios, it is not surprising.

This is due to how the experiments are designed, because even if the second scenario introduces two random variables (i.e. service time and inter-arrival time), their mean values are held fixed along tests. Whereas the mean vacation time is varied, which is a random variable in both scenarios, with the same distribution.

Thus as seen by the graphs, the introduction of the two random variables increases the offset of the straight line (due to randomness that worsen the performance), but does not change the overall linear trend.

5.3 Confidence intervals

This paragraph will cover the computation for the confidence intervals of experimental data above, and the relative comments.

The number of samples collected for each test is 10 (as stated before) and each sample will represent one different seed.

To compute CIs over a small group of samples (< 30) the t-student formula should be used with its related hypothesis: the sample has to be normal distributed.

$$\overline{X} \pm \frac{S}{\sqrt{n}} * t_{\alpha/2,\, n-1}$$

To test for this assumption to hold QQ-plots were generated for each set of seeds, but for compactness only the worst case met will be described.

This case has shown up in the first exponential experiment with increasing workload, for some certain parameter values: ST = 0.98s, IAT = 1s and VT = 10s.

Figure 5.3.1 shows the QQ-plot relative to the values of average number of jobs in the system taken from all the 10 seeds.

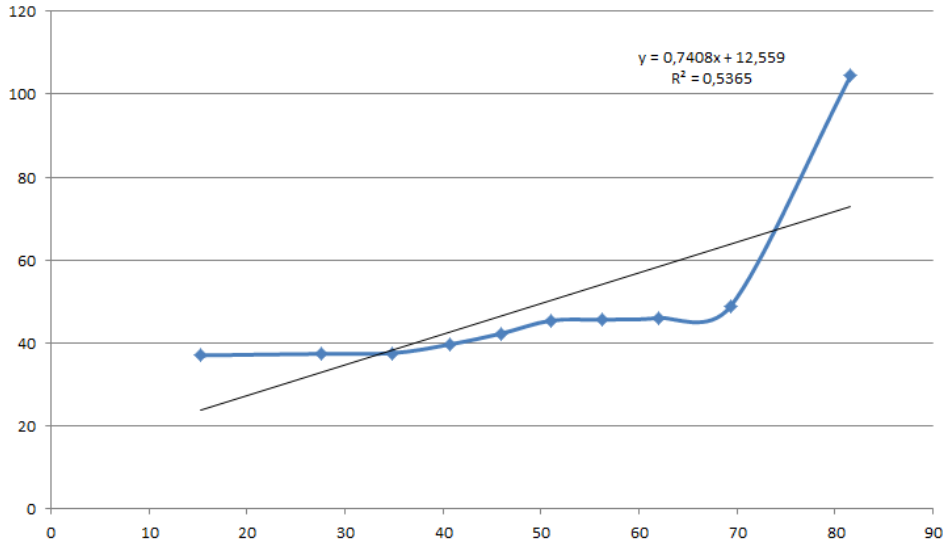

Figure 5.3.1 – QQ-plot with outliers

It is clear that the trend is not linear, but for all the other QQ-plots from the remaining experiments there is always some sort of linearity.
In fact in this particular case there is one value in the sample that is equal to 104.54, which is twice the mean value: 48.46. It is then reasonable to think of this particular value as an outlier and retry to plot the same graph without it. [Figure 5.3.2]

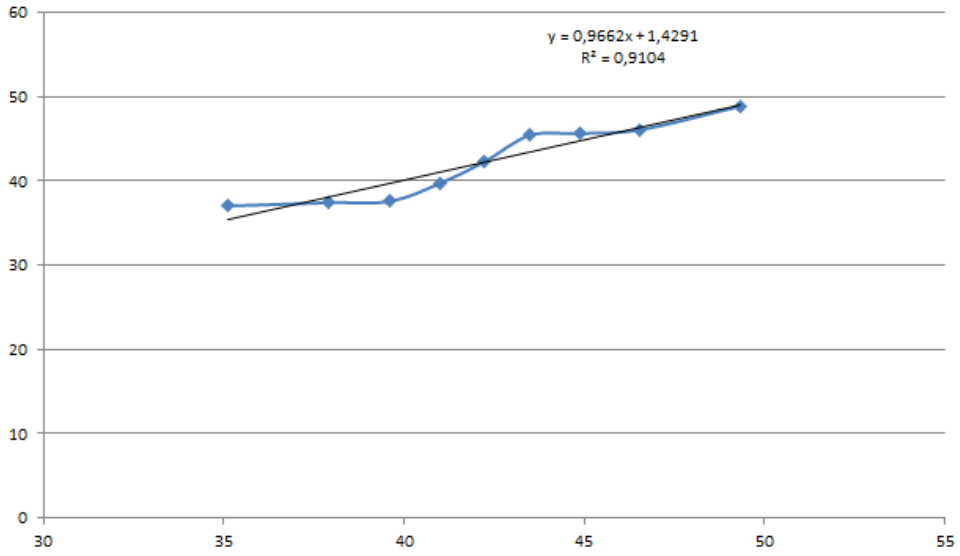

Figure 5.3.2 – QQ-plot without outlier

It can be observed that now it is quite linear, thus the normality of subsample is confirmed. Note that this situation occurs also for the response time, in the same case, since it depends on the number of jobs.
Therefore this outlier does not give certainty of that confidence interval. Assuming it as a particular behaviour of the system, it will not be discarded nor ignored. For this study it will just mean that this particular confidence interval should not be taken for granted.
In any case this situation regards the worst case, so all the other ones are confirmed to be normally distributed.

Conclusions

This documentation is aimed to show how a M/M/1 system is affected by a vacation. After the analysis and the study of the data what is resolved is that, in most cases, the period of vacation combined with the interplay between the service time and the interarrival time will deteriorate the system performance.

The way the vacation time affects the system depends on how the workload is chosen.

Indeed it can be noticed that for a service time that is smaller than the interarrival time (ST<= 0,6*IAT) the system will have an acceptable response time.