Project - The Carrefour

Mariano Basile, Angelo Buono, Corrado De Sio

January 17, 2016

Contents

List of Figures

1 Introduction

1.1 Problem Description

A supermarket has n tills. Two policies can be enforced to checkout:

a) Make a single queue in the supermarket, and send the head-of-line customer to the first idle till.

b) Allow each open till to menage its own (FIFO) queue. In this case, customers are expected to queue up at the till with the shortest queue (ties are broken arbitrarily).

Consider the following workload: customer inter-arrival times are IID RVs , their service demands are IID RVs. The number of tills can be varied (but stay constant in a single simulation). Under option a) consider a customer has to reach the first idle open till (say, till j), which takes $\Delta * i$ units of time - during which that tills remains idle.

1.2 Objectives

Assess the effectiveness of two different checkout policies enforced in a supermarket (e.g. The Carrefour).

More in detail:

a) Make a **single queue** and send the head-of-line to the first idle till (assuming a supermarket has n tills).

b) Allow **each** open **till** to manage **its** own (FIFO) **queue**. In this case, customers are expected to queue up at the till with the shortest queue (ties are broken arbitrarily).

Under option **a)** it's necessary to take into account that a customer in order to reach the first idle till - say till j - spends a time of $\Delta * j$ during which the till remains idle.

1.2.1 Performance Indexes

Based on these objectives we would like to compare the **queueing** and the **response** time of the two policies under a varying workload.What will be the best choice? It will be the policy with better queueing, response time. What does better mean in this case? It means achieving a **smaller queueing,response time**.

2 Modeling

2.1 Modeling the System

Under **policy a)** the model we have created is the following:

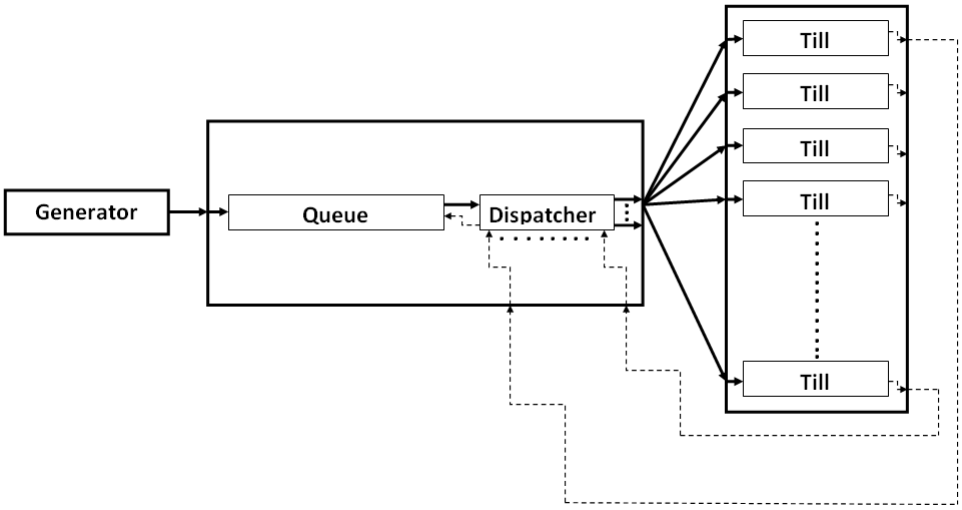

Figure 2.1: Model of the system under policy **a)**

Well, let's now explore deeply the model. There are three main components:

1. **Generator**: a single module. It models customers' arrivals at the supermarket;
2. **Waiters**: the first compound module. It consists of two simple modules:
   - a **Queue**, where all customers will queue up;
   - a **Dispatcher**, which is responsible of sending the head-of-line customer to the first idle Till.
3. **Tills**: the second compound module, an array of Tills, each of them modeled as a simple module.

We can now try to explain more into the detail what's the job of each module according to our code:

- **Generator**: it just schedules itself after a random time (exponentially or lognormal distributed) to create a new message that represent a new cutomer arrival into the system. The new message is than forwarded through an output gate called *"generatorServiceOut"*. Before sending, Generator module modifies message's scheduling priority to the value 1. Since Omnet++ in case of two or more messages received at the same time applies a scheduling based upon priority (where an higher value means a lower priority) the idea is to set an higher value so that Dispatcher module will be able to first update its state variables and then performs the routing. It's really important since there might happen that, at the same time, *Dispatcher* module receives a message both from Generator and any of Tills. And what if this was not handle? What does this mean in a real life scenario? Well, it's quite easy to explain: the next customer to be served will be routed to the wrong till, that means it won't be the first idle. In this way we're guaranteeing the correct behavior of the system.
- **Queue**: it has been implemented as a simple module and a class which has a data member of type *cQueue,* implemented and offered by Omet++ libraries. It has also an integer data type *counter,* that keeps track of the number of available Tills. During initialization counter is set to *"n"*, the (maximum) number of available Tills. When a new customer arrives if there's an idle till, so if counter is greater than 0, than he/she is forwarded to the Dispatcher in order to apply routing metrics. At the same time variable counter is decremented to keep track that one till is going to serve a customer (it's going to be busy).In case counter is equal to zero

so there's no till idle costumer just queues up. Our queue has been implemented as a module that is able to receive messages both from the Generator and Dispatcher. For the latter, messages can be seen as "*service messages*" since they do not represent customers, as those ones received from the Generator, but their aim is to inform the Queue that there's a new idle till, so counter can be incremented and the head of line customer can be sent to the Dispatcher (in case some customers have been queued up).

- **Dispatcher**: it's a simple module too. In order to compute routing metrics, that means in order to find the first idle till, a boolean array type of size equals to *"n"* has been initialized with all elements equal to true, since at the beginning all tills are supposed to be idle. Its name is "*exit*". When a new customer is received from the Queue, the module starts iterating over the array in order to find the first available till, so the first element which value is equals to true. Once found it, the corresponding till has marked to busy. This means setting the element's value in the array to false and then send the customer to that till through the gate having that index (the index at which we found the idle till). How does the Dispatcher know when one till has just finished to serve a customer, and why does it need to know? We can first answer the latter question: the reason is letting the queue forward one queued customer in order to let customers been served. How can this action be triggered? We can answer this question saying how the dispatcher know when one till has turned its own state into idle: when a notify message is received from any of the available tills, Dispatcher module can set again to true the element's value in the array which index is given by the arrival gate on which the notify message has been received. This repesents the trigger action and it is implemented by forwarding the received notify message to the Queue.

- **Till**: the lastone simple modules in our model. When a new customer is received, first it saves in one of its data member called *"enqueingCustomerTime"* the time at which the customer entered in the queue (this will be usefull later on). As soon as customer starts being served till module emits customer's queueing time, as the difference between that time and time at which he/she enters the queue (stored in *enqueingCustomerTime).* The module schedules itself with random time following different distribution (exponential or lognormal in our case) in order to simulate service times. Only at that end of a service demands till module emits customer's response time, as the difference between that time and (again) the time at which he/she enters the queue (keep stored in *enqueingCustomerTime). A notify message* is than created and sent to the dispatcher in order to let it updates its array named exit since one till is again idle. At this time notify message scheduling priority is set to 0 since it has priority over a new customer to let the system behaves correctly.

Let's now go one step further and start to have a look to the model we've created under policy **b):**

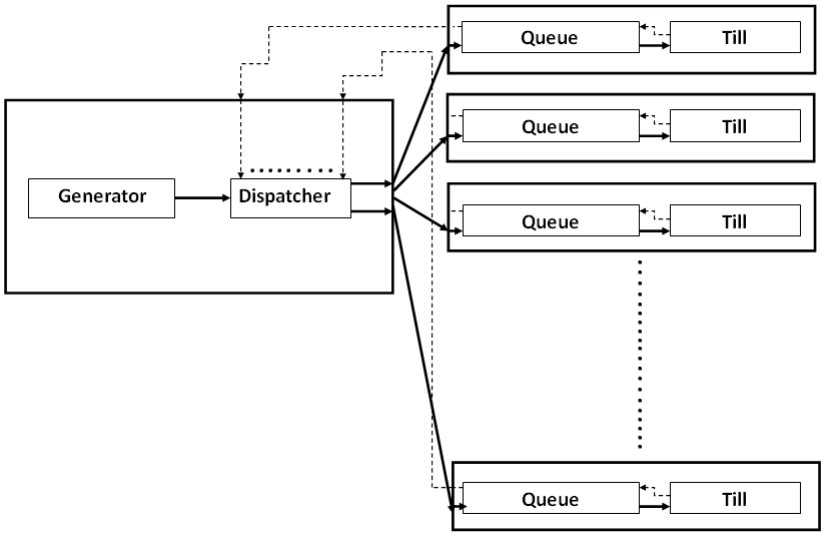

Figure 2.2: Model of the system under policy **b)**

Here the main difference is the fact that each open till has its own (FIFO) queue.

We can try to analyze this model by just showing components that behave different from before:

- **Dispatcher**: still a simple module. The previous array *"exit"* has been now turned into an integer array of elements called *"queuesLength"*, still of length "n" where n is the number of open tills.Why did we call it in that way? Because we thought each element can represent the number of queued customers at each till. At initialization all element's value have been set up to 0 since each queue at each till has 0 en-queued customers. What does it happen when a new customer arrives? Well, what the module does is just finding the queue with the minimum number of element queued and routes customers to that queue.Before routing it increases the number of en-queued customers at that till and sends through the gate having the index at which we found the minimum customer's number. What in case two or more queue have the same number of queued customers? In this case a customer will be routed to the till,above ones with the same minimum number of customers, with the smallest index in the array. The reason of this choice is based on the assumption that ties are broken arbitrarily. Anyway we can state this is not the best choice since, in general, this solution could result in worse performance indexes (e.g. higher queueing time)on some given tills, since we're enforcing this policy every time this situation shows up. Which tills are we talking about? We're talking about till#1 (or equivalently till at index 0 in the array) if first open till has the same minimum number of customers as ones of the following, or in general to the first with smallest index between ones having the same minimum number of customers. What the other main difference? Well now *"service messages"* are sent by Queue module to inform Dispatcher o update its state variable (and no more from any of available Till). Things are pretty much different since a *"service message"* is now created by some till to inform its associated queue to change its state variable *"free"* (which purpose will be described in the next paragraph) to true (that means till changes its state to idle). At this time queue module can evolves accordingly (still described in the next paragraph).One of the action performed is sending a notify message to the dispatcher in order to update (decrement) the overall residual number of customers relative to the that compound module (queue and till).
- **Queue**: the main differences are relative on the operations perform when a new customer arrives in the queue and when the associated till changes its state into idle. What about a new arrival? In this case by using a boolean variable called *"free"* each queue is able to determine its till's state.If till is in idle state the new customer is forward to the till and than free is set to false, otherwise the customers is queued. What instead when a new till finishes serving a customer? A notify message is received by the associated queue so that the state variable free can be changed to true again.At this time, of course, if there are queued customers, the head of line is forwarded to the till and the dispatcher performs exactly the same operations it does when a new customers arrives and finds till in idle state.

The compound module composed of Generator and Dispatcher has been named **Aisles**, instead the compound composed of till and its associated queue has been named **Checkout**.

2.1.1 Assumptions

In case of policy b) we made a constraints that consists in avoiding **customers arrivals from other queues**. A particular case, that in a real life scenario may happen is the following:

- suppose the dispatcher at some given point in time has to choose one till between the ones having the same minimum number of elements and suppose this number is equal to 1 (this means that the one and only one customers is being served). As we know, the dispatcher will route the new customer to the till with the smallest index in the array. What can happen in this case is that the choice may not be the best one since as soon as he/she arrives to the elected till, one till among the eligible ones finishes to serve. In this case what we have is that our customers is experimenting queueing time but one till is idle. What happens in a real life scenario is that once the customer is in the queue (choosen by him/herself) may decide to abandon that queue to immediately be served by the idle cashier. So what happens in a real life scenario is a little different of what happens in our model. Giving the fact this difference may arise only under particular circumstances and only in case the elected till is not the one at which cashier will terminate demand first and since the system's overall queueing and response time will not be affected by these episodes we can just ignore it.

2.1.2 Verification

To verify that our code correctly implement the model and in order to spot any possible *"logical bugs"*, we proceeded by incrementally adding complexity to our test simulations and verifying each time the correctness of results. At

very beginning we relied on some deterministic model in particular we assumed **constant inter-arrivals times and service times for either policy a) and b)**.

At the very beginning for both policy a) and b) we imposed:

λ	μ	n
60s	120s	1

and for policy a) we assumed:Δ= 2s. We ran some test simulations for 1 hour and we observed:

- **for policy b) we found out that only 30 customers out of 60** have been served at the end of simulation.This matched what we expected from the system since if each customer spends a time equals to 120s to be served, with just 1 till in 1 hour of simulation only **3600/120 = 30** will be served.
- **for policy a) we found out that only 29 customers out 60** (it was 61 but it depends on the time simulation stops) h**ave been served** at the end of simulation.This also matched what we expected from the system. Each customer, in order to be served, experiments a time (response time) that can be seen as the sum of the constant service time (120s) and the delay in order to reach the till. With only 1 till (open) in 1 hour of simulation only **3600/122=29** customers will be served.

Module	Name	Value
Carrefour_B.a.g	countCustomer:count	60.0
Carrefour_B.c[0].t	responseTime:mean	990.0
Carrefour_B.c[0].t	responseTime:count	30.0
Carrefour_B.c[0].t	queueingTime:mean	870.0

Module	Name	Value
Carrefour_A.g	countCustomer:count	61.0
Carrefour_A.ts.t[0]	responseTime:mean	990.0
Carrefour_A.ts.t[0]	responseTime:count	29.0
Carrefour_A.ts.t[0]	queueingTime:mean	901.0

Figure 2.3: **Verifying code** against the system under policy **b)** and **a)** (**verification phase**).

We added complexity to the system just increasing the arrival rate and setting up n such that all customers were able to be served at least for policy b) (since for policy a) we need to start taking into account the delay).We run some simulation tests with the following parameters:

λ	μ	n
30s	120s	4

and we still assumed for policy a):Δ= 2s.

We ran some test simulations for 1 hour and we observed:

- **for policy b) we found out that only 117 customers out of 120 have been served** at the end of simulation. Why? As we said before we expected that 120 out of 120 had been served. The reason is that last statistics hadn't been emitted.This case is pretty general and depends on time at which the simulation stops. In order to state that our model behaves as we expected we than verify that each customer didn't experiment any queueing time, that means that at each arrival each customer finds an idle till. This has been successfully verified as shown in pictures (Figure 2.6) just below.
- **for policy a)** things start to be a little tricky. Now time spent to reach each till is:
  - $\Delta = 2$ for till at index 0 (till #1);
  - $\Delta = 4$ for till at index 1 (till #2);
  - $\Delta = 6$ for till at index 2 (till #3);
  - $\Delta = 8$ for till at index 3 (till #4);

What we expected in this case is the following:

- Till #1 will be able to serve only 3600/122 = 29 customers;
- Till #2 will be able to serve only 3600/124 = 29 customers;

- Till #3 will be able to serve only 3600/126 = 28 customers;
- Till #4 will be able to serve only 3600/128 = 28 customers;

Their sum is equal to 114, but what we encountered is that only 112 out of 114 customers have been served at the end of simulation. The result can be consider correct because the problem we described in the paragraph just above; once again some statistics could not be emitted according to the time at which the simulation has been stopped.

Module	Name	Value
Carrefour_B.a.g	countCustomer:count	120.0
Carrefour_B.c[0].t	responseTime:mean	120.0
Carrefour_B.c[0].t	responseTime:count	30.0
Carrefour_B.c[0].t	queueingTime:mean	0.0
Carrefour_B.c[1].t	responseTime:mean	120.0
Carrefour_B.c[1].t	responseTime:count	29.0
Carrefour_B.c[1].t	queueingTime:mean	0.0
Carrefour_B.c[2].t	responseTime:mean	120.0
Carrefour_B.c[2].t	responseTime:count	29.0
Carrefour_B.c[2].t	queueingTime:mean	0.0
Carrefour_B.c[3].t	responseTime:mean	120.0
Carrefour_B.c[3].t	responseTime:count	29.0
Carrefour_B.c[3].t	queueingTime:mean	0.0

Module	Name	Value
Carrefour_A.g	countCustomer:count	121.0
Carrefour_A.ts.t[0]	responseTime:mean	186.20689655172
Carrefour_A.ts.t[0]	responseTime:count	29.0
Carrefour_A.ts.t[0]	queueingTime:mean	69.0
Carrefour_A.ts.t[1]	responseTime:mean	189.78571428571
Carrefour_A.ts.t[1]	responseTime:count	28.0
Carrefour_A.ts.t[1]	queueingTime:mean	72.413793103448
Carrefour_A.ts.t[2]	responseTime:mean	194.14285714286
Carrefour_A.ts.t[2]	responseTime:count	28.0
Carrefour_A.ts.t[2]	queueingTime:mean	76.551724137931
Carrefour_A.ts.t[3]	responseTime:mean	195.33333333333
Carrefour_A.ts.t[3]	responseTime:count	27.0
Carrefour_A.ts.t[3]	queueingTime:mean	77.428571428571

Figure 2.4: Verifying code against the system under policy **b)** and **a)** (**verification phase in case of higher complexity**).

We went one step further and by using Omnet++ **GUI**, we add more and more complexity and we tried to figure out what happens by **visual inspection**. We set up some simulation tests by changing parameters' values to the following ones:

λ	μ	n
20s	180s	8

(still with the assumption of having under policy a)Δ= 2s). At the end of this phase we stated that code performs as we expected. One (other) big change we made was running test simulations for for more than 1 hour (3h). We decided to run these simulations for 3 hours (of simulated time). We expected that 10800/20 = 540 customers enter in the system and since n is not sufficient to let all customers be served we expected an incresing in queueing time.

In particular when **policy b) is applied** in order to serve 540 customers under a costant service time of 180s it's necessary to set n equals to 9 instead of 8 since (10800/180)*9=540. **Under policy a)** things are even worse because delay. Delay's impact to the system increases linearly as the # of tills increases: in order to reach say till #8 a customer has to spend a time equals to 8*Δ which means 16s in caseΔ=2s (as in this case). **This under policy a) is pretty general (to reach any of available tills, if we suppose their number is equal to n, one customer have to waste a time equals to $\triangle$ * j with j=1,2,3.....n).**

2.2 Code

In this paragraph we're going to show an overall picture about system's implementation under the two policies. **The model of the system under policy a)** presented in Figure 2.1, has this general overview in the simulator:

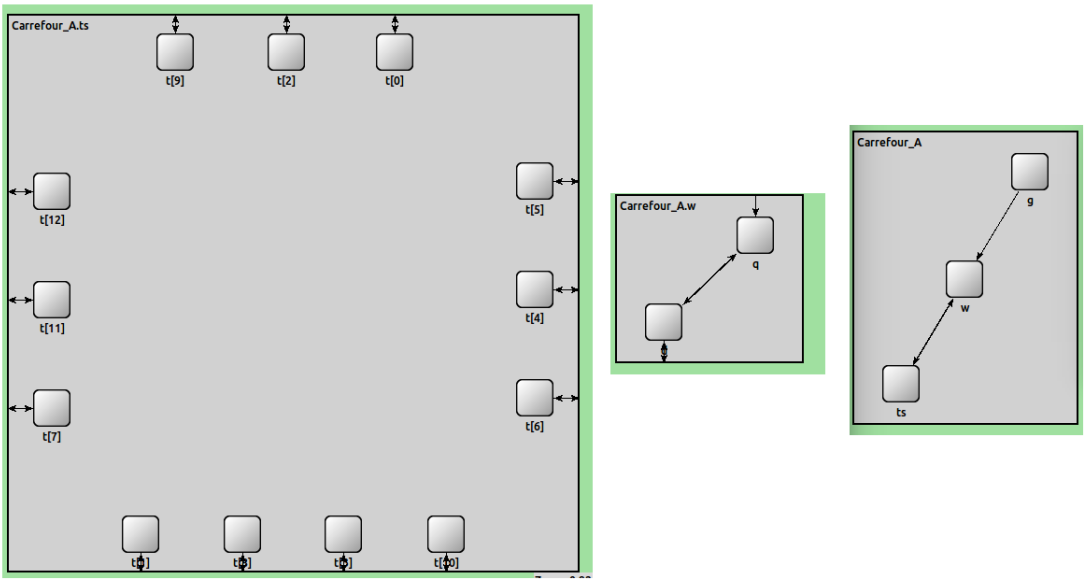

Figure 2.5: **Model of the system under policy a) in Omnet++.**

The model under policy a) described in terms of classes, data members and functions can be explained **by** the following **Class-Diagram**:

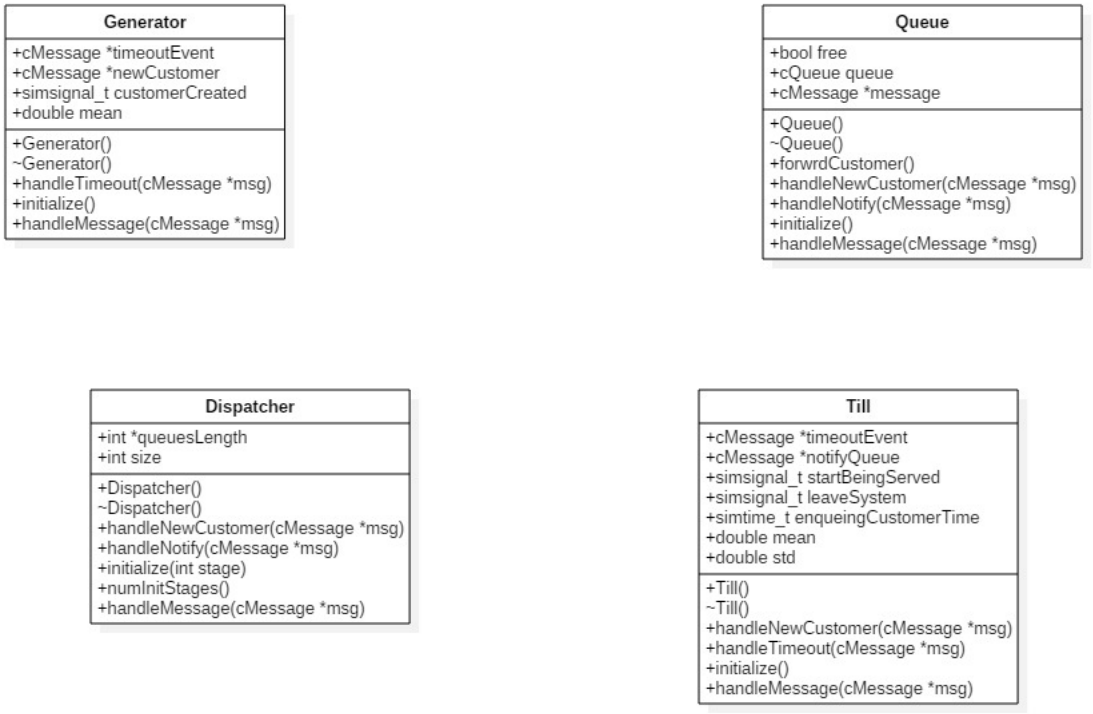

Figure 2.6: **Model's class-Diagram under policy a)**

The following scheme can be used instead to summarize **how the system behaves if policy a) has been applied**:

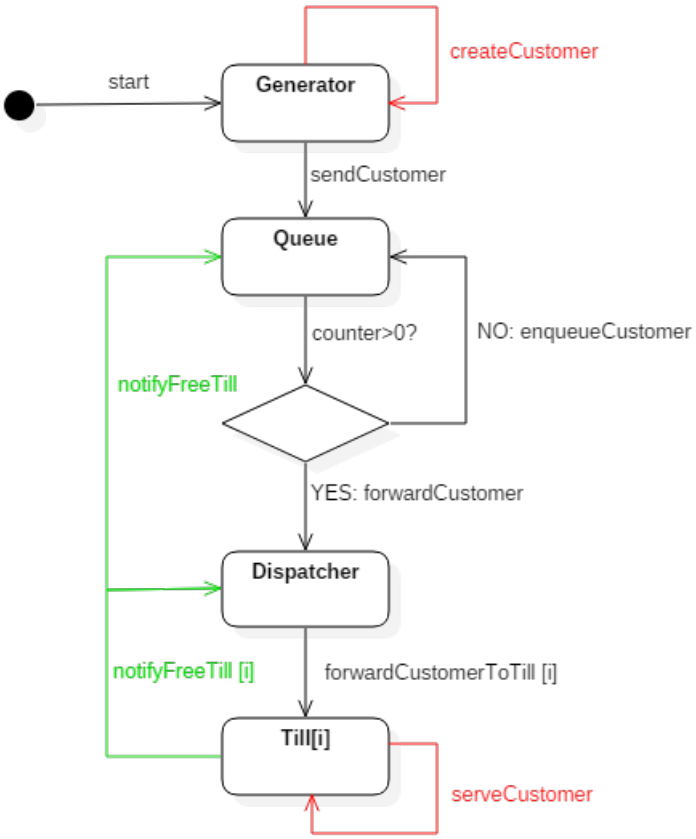

Figure 2.7: **System behavior under policy a)**

The model of the system under policy b), presented in Figure 2.2, has the following general overview in the simulator:

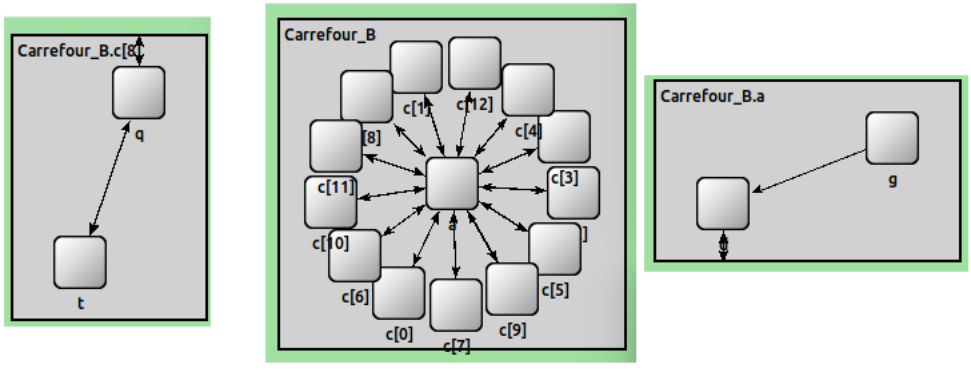

Figure 2.8: **Model of the system under policy b) in Omnet++.**

The model under policy b) described in terms of classes, data members and functions can be explained **by** the following **Class-Diagram**:

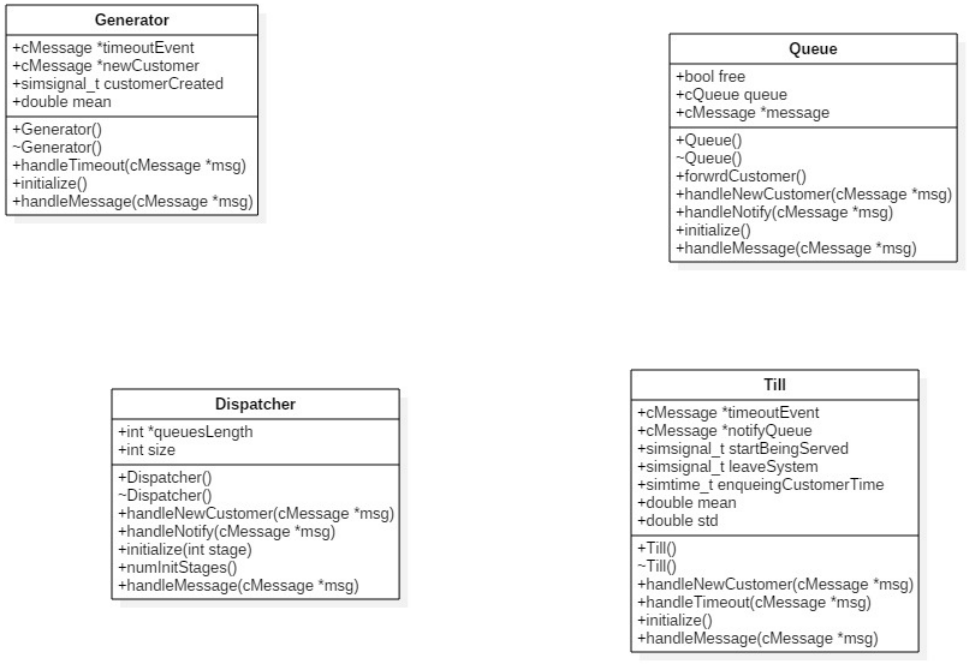

Figure 2.9: **Class-Diagram for the system under policy b)**

The general system's behavior under policy b) can be summarized in this schema:

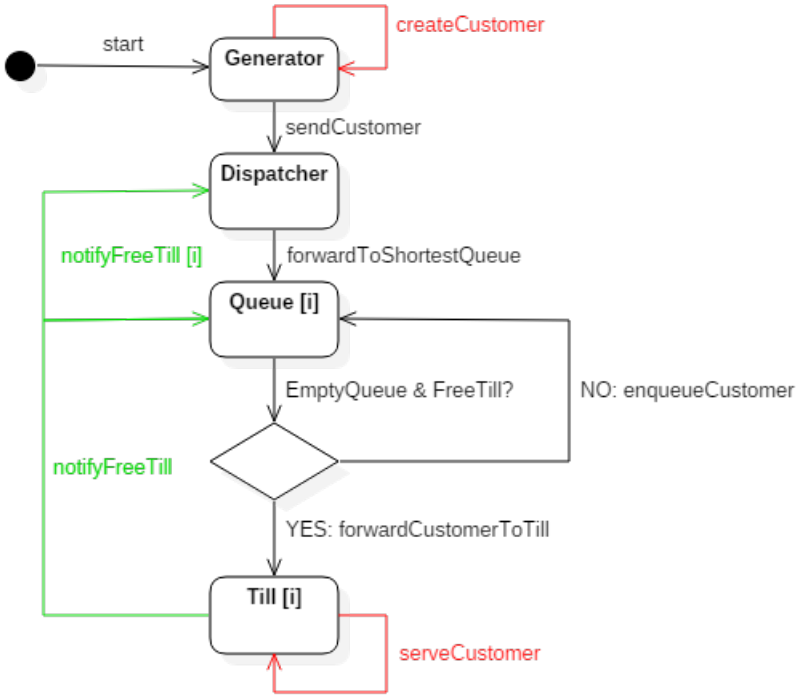

Figure 2.10: **System behavior under policy b)**

2.3 Factors and Calibration

2.3.1 Factors and Parameters

2.3.1.1 Factors We have a common factor in the two policies, the number of tills n. In the policy a) we have a second factor, Δ the unit of time to reach the next till. In details:

n Number of tills in the system. This parameter stay constant during the single simulation. It is a resource of the system and has a cost (so low is better).

Delay(Δ) In a) is the time (in seconds) that a customer need to reach the next till (so starting from the queue he will need Δ for reach the first till, 2Δ for the second one and so on). During this time the till remains idle. This factor is typical of the supermarket topology. When this factor is > 0 add a delay to the performance indexes. This means this factors degrade the goodness of the system, so for Δ low is better.

2.3.1.2 Parameters We have some parameters that define the scenario that we are going to simulate (this parameters are the same for a and b).

λ The inter-arrival times are IID RVs modeled with a exponential distribution. λ is the E[X] of the exponential distribution. This mean that in case we have $\lambda = \bar{\lambda}$ the exponential distribution of the arrives will be $\frac{1}{\lambda}e^{-\frac{1}{\lambda}x}$ when x>0, and 0 otherwise.

μ 1. When the service demands are IID RVs modeled with a exponential distribution μ is the E[X] of the exponential distribution. This mean that in case we have $\mu = \bar{\mu}$ the exponential distribution of the arrives will be $\frac{1}{\mu}e^{-\frac{1}{\mu}x}$ when x>0, and 0 otherwise.

2. When the service demands are IID RVs modeled with a lognormal distribution $\mu$is E[X] of the lognormal distribution.

σ When the service demands are IID RVs modeled with a lognormal distribution, σ^2 is the Var(X) of the lognormal distribution.

2.3.2 Scenario Definition

With the purpose of setting up some realistic scenarios we've tried to convert into numbers some statistics **provided by Google**. **We relied on some graphs which show past visits to the Carrefour situated here in San Giuliano Terme,(Pisa)**.

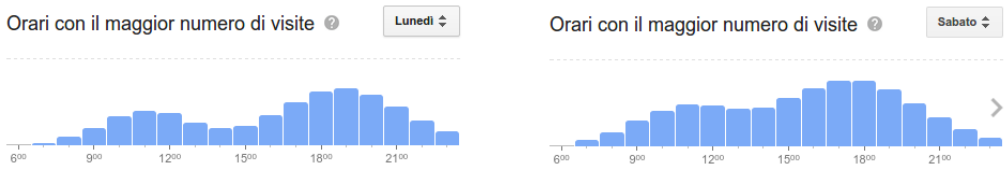

Figure 2.11: Past visits to Carrefour in San Giuliano Terme (Pisa).

We used these graphs quite a lot since they give us at least couple information:

- **arrival rates**, since it is possible to link these histograms to numbers. These numbers may represent (different) arrival rates at different hours in a day and/or in different days;
- **simulation time**, once warmup period has been discovered (once t0 has been set up) it's possible to set the right extreme of our study (which means t1) by just adding to t0 the value obtained by counting the numbers of rectangles having the same height. This is possible since we can assume that we have certain arrival rates for a finite time only.

2.3.3 Calibration

In order to choose realistic value for the factors and parameters we collected some information and ended to some considerations.

For this purpose Google provides us a set of graphs which shows a visits rate based on historical visits. From this we generated a λ associated to each visit rates and ended with various arrives scenarios mapped in some day and time of the week.

Hr	Sun	Mon	Tue	Wed	Thu	Fry	Sat
07:30-08:30	-	-	90s	–	-	-	-
08:30-09:30	-	-	90s	-	-	-	-
09:30-10:30	-	-	-	-	60s	-	-
10:30-11:30	-	-	-	-	60s	-	30s
11:30-12:30	-	-	-	-	60s	-	30s
12:30-13:30	-	-	-	-	60s	-	30s
13:30-14:30	-	-	-	-	60s	-	30s
14:30-15:30	-	-	-	-	60s	-	30s
15:30-16:30	-	-	-	-	60s	-	-
16:30-17:30	-	-	-	-	40s	-	20s
17:30-18:30	15s	25s	-	-	40s	-	20s
18:30-19:30	15s	25s	-	-	40s	-	-
19:30-20:30	-	25s	-	-	40s	-	-
20:30-21:30	-	-	-	-	40s	-	-
21:30-22:30	-	-	-	-	-	-	-

Table 1: Value of λ for different day and time

For service demands in both cases (exponential and lognormal distribution) after some interview and field observation we chose a E[X] =150s and a $Var(X) = 150^2 s$ that supposed fit realistic scenarios.

The factor n is choosed in function of λ to have a number of tills that can menage the incoming customers. Is realistic to suppose that the minimum n should be at least

$$\frac{1}{\lambda} \leq n\frac{1}{\mu} \Longrightarrow n \geq \frac{\mu}{\lambda}$$

and increasing n step by step if the simulation show that n is too little (remeber that n low is better, so i'm searching the littlest n that can menage the incoming customers). Should be easy to suppose that in case a) where we have an additional delay the minimum n that truth the disequation sometimes would be not enough.

Δ is set to 4 seconds cause usually the tills are not distant each other, and this seems a reasonably value. Anyway this factor is topology dependent and hard to decrease under a bottom value that we set at 4 seconds. Remember that Δ degrade the prestation of system and for Δ greater our performances for system a can only be worst.

The simulation time is chose in agree with Table 1, as Simulation time = Warmup time + #hour, cause is reasonably to suppose that the system will be yet in steady state after the first hours the supermarket has been open. Indeed warmup time is choosed to avoid the contribute of transit state in the simulation.

3 Data Collections and Analysis

3.1 Scenarios Evaluation

Our aim is to compare queueing and response times of the two policies under a varying workload.In order to do so we evaluated the following scenarios:

- exponential distribution of interarrival times and service demands;
- lognormal distribution of service demands;

3.1.1 Exponential Distribution Of InterArrival Times - Exponential Distribution Of Service Demands

As we said at paragraph 2.2.2, we relied on some data provided by Google: this data show information about past visits at the Carrefour situated here in San Giuliano Terme (Pisa).Our idea was to use this data in order to calibrate in a real manner our simulations.We used them not only to setup different scenarios in the same day but also to retrieve some significant results between different days.The followings we're presenting are the most significant results we've found.They covered a whole week visit to the mentioned supermarket.

Data shown in the above tables summarize many aspects:

- Scenario Calibration;
- IID-ness of outputs;
- Confidence Intervals;

3.1.1.1 Monday (λ=1/25)

Figure 3.1: **Simulating Monday evening 5.30 p.m - 8.30 p.m**

- ***POLICY A***

warmup - period	sim-time-limit	repetition	λ	μ	N	Δ
30000	40800	36	1/25	150	8	4s

We first verified the IID-ness of outputs data.

We based our study on the following assumptions:

Confidence Level	$z\alpha/2$	$\sqrt{n}$	$z\alpha/2/\sqrt{n}$
95%	1.96	6	0,326666667

The followings are the two correlograms used to verify the IID-ness of outputs data (queueing and response times):

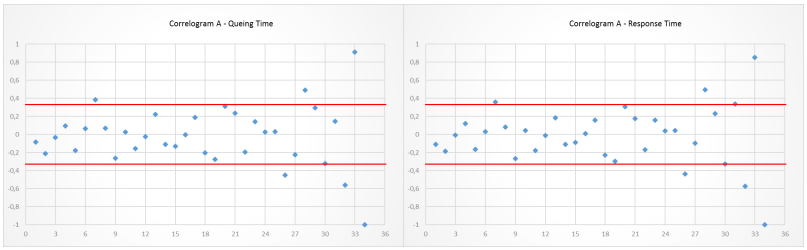

Figure 3.2: **Verifing IID-ness assumptions of output data in case of policy a) queueing times** .

From the above graphs the assumptions seems to be false, since not all auto-correlation coefficients are inside the bounds ±0,326666667. Anyway if we take a closer look we can see that these outliers are at large lags. Since autocorrelation function is less reliable due to the fact that the sum by which it is computed has too few values we cannot worry so much and state that the IID-ness of outputs data has met.

The average queueing and response time of the system are the following:

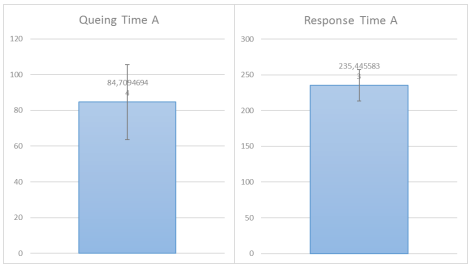

Figure 3.3: Average Queueing Time and Response times under policy a)

Values have also been summarize in the above tables:

E[W]	Confidence Interval	E[R]	Confidence Interval
84.70	[63.71; 105.70]	235.44	[213.46; 257.42]

- ***POLICY B***

warmup - period	sim-time-limit	repetition	λ	μ	N
26000	36800	36	1/25	150	8

As we did for policy a) we first verified the IID-ness of outputs data based on the same assumptions.

The two correlograms for policy b) are the followings:

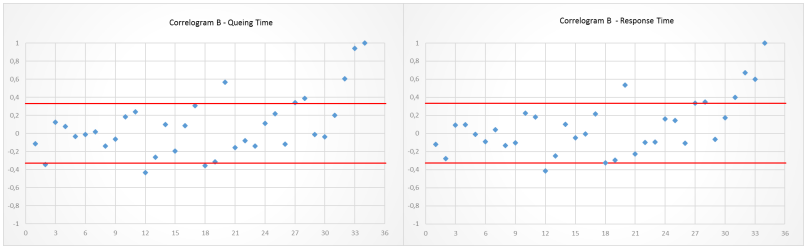

Figure 3.4: Verifying IID-ness assumptions of output data in case of policy b).

The average queueing and response time of the system are the following:

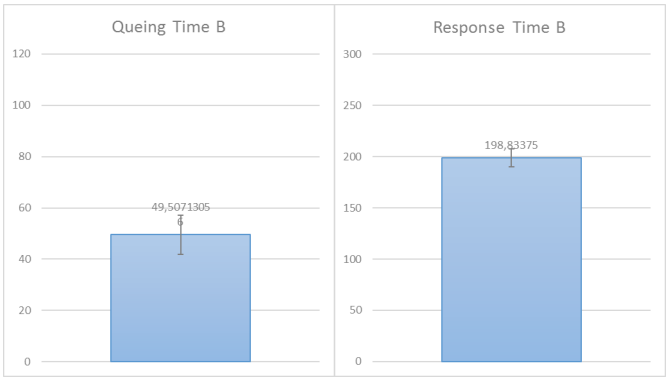

Figure 3.5: Average Queueing and Response times under policy b)

Summarized values are shown in these two tables:

E[W]	Confidence Interval
49.50	[41.83; 57.17]

E[R]	Confidence Interval
198.83	[190.00; 207.65]

- ***Comparing policies***

E[Wa]	E[Wb]	Ratio (E[Wa]/E[Wb])	E[Ra]	E[Rb]	Ratio (E[Ra]/E[Rb])
84.70	49.50	**1.71**	235.44	198.83	**1.18**

Since there's no overlapping in terms of confidence intervals, these tables ensure that the average queueing time of the system under policy a) is about 1.71 higher.The average response time is instead 1.18 higher. This means that giving to the system this same workloads, policy b) outperforms policy a).

3.1.1.2 Tuesday (λ=1/90)

Figure 3.6: **Simulating Tuesday morning 07.30 a.m - 9.30 a.m**

- ***POLICY A***

warmup - period	sim-time-limit	repetition	λ	μ	N	Δ
60000	67200	36	1/90	150	2	4s

We first verified the IID-ness of outputs data.

We based our study on the following assumptions:

Confidence Level	$z\alpha/2$	$\sqrt{n}$	$z\alpha/2/\sqrt{n}$
95%	1.96	6	0,326666667

As in the previous case of study we first assessed the IID-ness of outputs data (still relying on the same assumptions).

Correlograms from now on will not be presented since the IID-ness of outputs data has been always met.In any case correlograms are available for consulting within the files attached to this documentation.

The average queueing and response time of the system are the following:

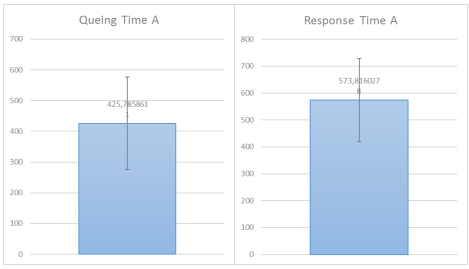

Figure 3.7: Average Queueing Time and Response times under policy a)

Values have also been summarize in the above tables:

E[W]	Confidence Interval	E[R]	Confidence Interval
425.78	[275.78; 575.78]	573.81	[420.03; 727.60]

- ***POLICY B***

warmup - period	sim-time-limit	repetition	λ	μ	N
60000	67200	36	1/90	150	8

As we did for policy a) we first verified the IID-ness of outputs data based on the same assumptions.

The average queueing and response time of the system are the following:

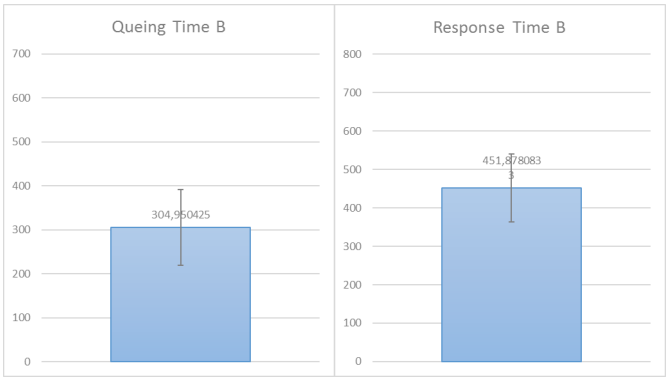

Figure 3.8: Average Queueing and Response times under policy b)

Summarized values are shown in these two tables:

E[W]	Confidence Interval
304.95	[218.88; 391.01]

E[R]	Confidence Interval
451.87	[363.17; 540.58]

- ***Comparing policies***

  Since there's overlapping in terms of confidence intervals, both for queueing and response times we cannot say nothing at all.This may be due to the higher confidence level (95%) so our idea was to decrease it and see if anything changes.

  Using a confidence level of 90% we get the following confidence intervals.

  Under **policy a)**:

Confidence Interval for E[Wa]	Confidence Interval for for E[Ra]
[299.89; 551.68]	[444.74; 702.88]

  Under **policy b)**:

Confidence Interval for E[Wb]	Confidence Interval for for E[Rb]
[232.71; 377.18]	[377.42; 526.32]

  We can conclude that since with a smaller confidence level (at 90%) we still observe overlapping, both for queueing and response times, anything meaningful can be stated.

3.1.1.3 First Thursday (λ=1/60)

Figure 3.9: **Simulating Thursday afternoon 10.30 a.m - 4.30 p.m**

- ***POLICY A***

warmup - period	sim-time-limit	repetition	λ	μ	N	Δ
70000	80800	36	1/60	150	3	4s

The average queueing and response time of the system are the following:

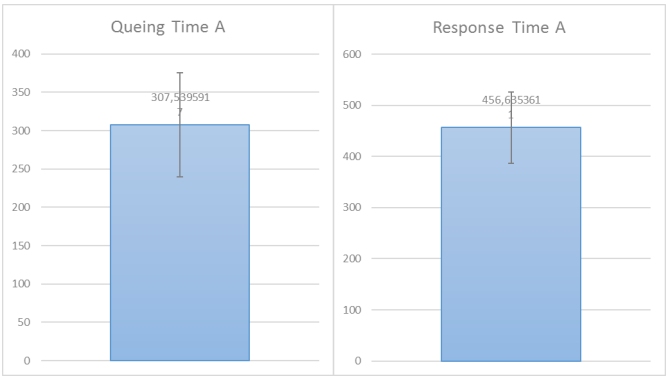

Figure 3.10: Average Queueing and Response times under policy a)

Values have also been summarize in the above tables:

E[W]	Confidence Interval	E[R]	Confidence Interval
307.53	[239.50; 375.57]	456.63	[386.86;526.40]

- ***POLICY B***

warmup - period	sim-time-limit	repetition	λ	μ	N
45000	55800	36	1/60	150	3

As we did for policy a) we first verified the IID-ness of outputs data based on the same assumptions.

The average queueing and response time of the system are the following:

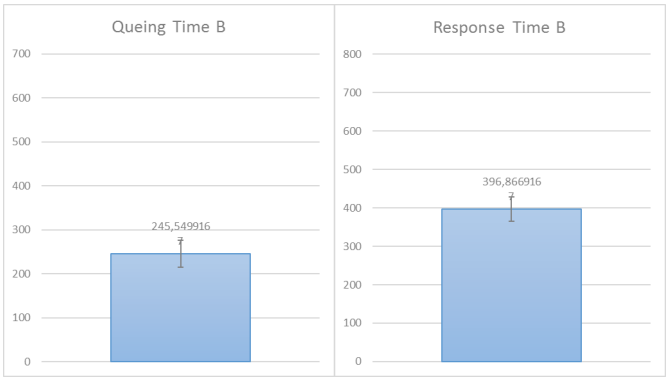

Figure 3.11: Average Queueing and Response times under policy b)

Summarized values are shown in these two tables:

E[W]	Confidence Interval
245.54	[214.92; 276.17]

E[R]	Confidence Interval
396.86	[364.67; 429.06]

In this case (as the previous one) because overlapping in terms of confidence intervals, both for queueing and response times, we cannot say which one of the policies performs better. We then decreased the confidence level to 90% as before and here what we observed:

Under **policy a)**:

Confidence Interval for E[Wa]	Confidence Interval for for E[Ra]
[250.44.50; 364.63]	[398.07;515.19]

Under **policy b)**:

Confidence Interval for E[Wb]	Confidence Interval for for E[Rb]
[219.84; 271.25]	[369.84; 423.88]

We can conclude that since with a smaller confidence level (at 90%) we still observe overlapping, both for queueing and response times, anything meaningful can be stated.

3.1.1.4 Second Thursday (λ=1/40)

Figure 3.12: **Simulating Thursday evening 4.30 p.m - 9.30 p.m**

- ***POLICY A***

warmup - period	sim-time-limit	repetition	λ	μ	N	Δ
20000	38000	36	1/40	150	5	4s

The average queueing and response time of the system are the following:

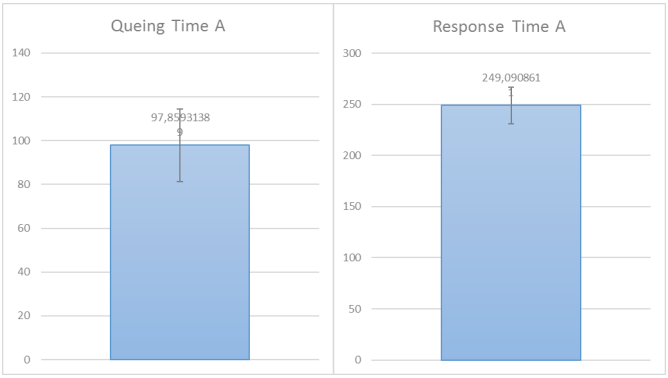

Figure 3.13: Average Queueing and Response times under policy a)

Values have also been summarize in the above tables:

E[W]	Confidence Interval	E[R]	Confidence Interval
97.85	[81.41; 114.30]	249.09	[231.04; 267.13]

- ***POLICY B***

warmup - period	sim-time-limit	repetition	λ	μ	N
45000	63000	36	1/40	150	5

The average queueing and response time of the system are the following:

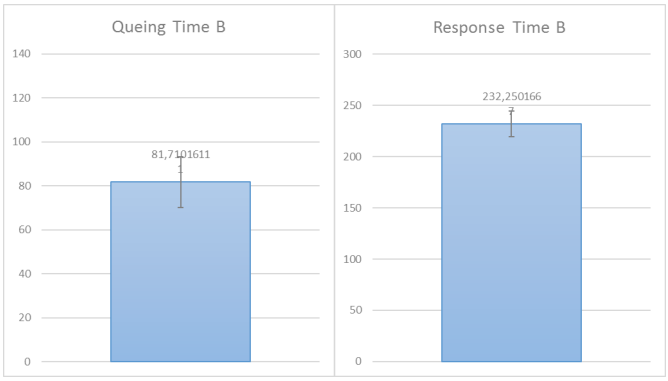

Figure 3.14: Average Queueing and Response times under policy b)

Summarized values are shown in these two tables:

E[W]	Confidence Interval
81.71	[70.05; 93.36]

E[R]	Confidence Interval
232.25	[219.61; 244.88]

In this case (as the previous one) because there's overlapping in terms of confidence intervals, both for queueing and response times, we can try to decrease confidence level to 90%.

Under **policy a)**:

Confidence Interval for E[Wa]	Confidence Interval for for E[Ra]
[84.05; 111.66]	[233.94; 264.23]

Under **policy b)**:

Confidence Interval for E[Wb]	Confidence Interval for for E[Rb]
[71.92; 91.49]	[221.64; 242.85]

We can conclude that since with a smaller confidence level (at 90%) we still observe overlapping, both for queueing and response times, anything meaningful can be stated.

3.1.1.5 First Saturday (λ=1/30)

Figure 3.15: **Simulating Saturday 10.30 a.m - 3.30 p.m**

- ***POLICY A***

warmup - period	sim-time-limit	repetition	λ	μ	N	Δ
45000	63000	36	1/30	150	6	4s

When data of this simulation has been tested for IID hypothesis, we obtained the next graph:

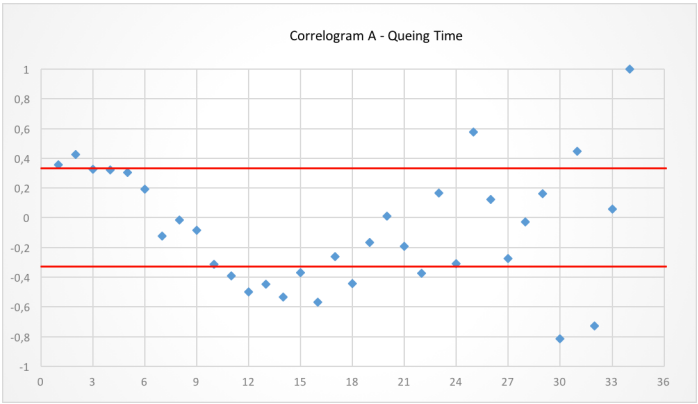

Figure 3.16: Correlogram before subsampling

We can see how some points are lower than low bound, and that they show a weird correlation. So we choosed to subsampling (32 sample) this results, and we arrived to a better correlogram

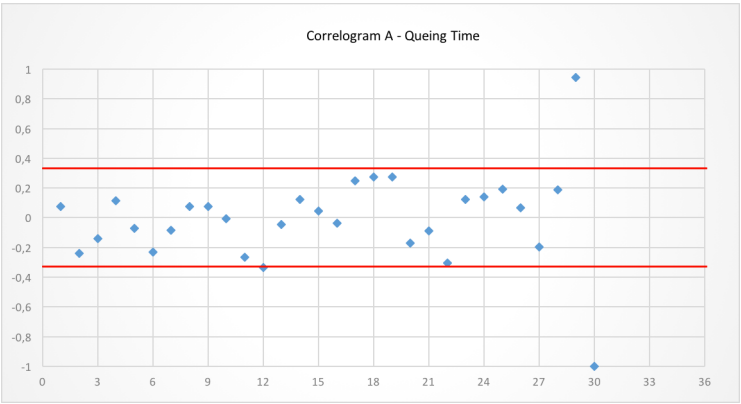

Figure 3.17: Correlogram with subsampling

The average queueing and response time of the system are the following:

Figure 3.18: Average Queueing and Response times under policy a)

Values have also been summarize in the above tables:

E[W]	Confidence Interval	E[R]	Confidence Interval
178.08	[154.46; 201.7]	356.76	[328.58; 384.93]

- ***POLICY B***

warmup - period	sim-time-limit	repetition	λ	μ	N
40000	58000	36	1/30	150	6

The average queueing and response time of the system are the following:

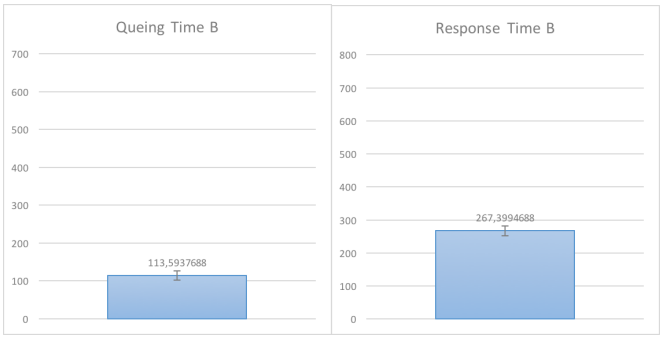

Figure 3.19: Average Queueing and Response times under policy b)

Summarized values are shown in these two tables:

E[W]	Confidence Interval
113.59	[101.08; 126.10]

E[R]	Confidence Interval
267.39	[252.61; 282.18]

- ***Comparing policies***

E[Wa]	E[Wb]	Ratio (E[Wa]/E[Wb])	E[Ra]	E[Rb]	Ratio (E[Ra]/E[Rb])
178.08	113.59	**1.57**	356.76	267.39	**1.33**

Tables just above ensure that the average queueing time of the system under policy a) is about 1.57 higher.The average response time is instead 1.25 higher. This means that policy b) outperforms policy a) also in this case.

3.1.1.6 Second Saturday (λ=1/20)

Figure 3.20: **Simulating Saturday 4.30 p.m - 6.30 p.m**

- ***POLICY A***

warmup - period	sim-time-limit	repetition	λ	μ	N	Δ
120000	127200	36	1/20	150	9	4s

The average queueing and response time of the system are the following:

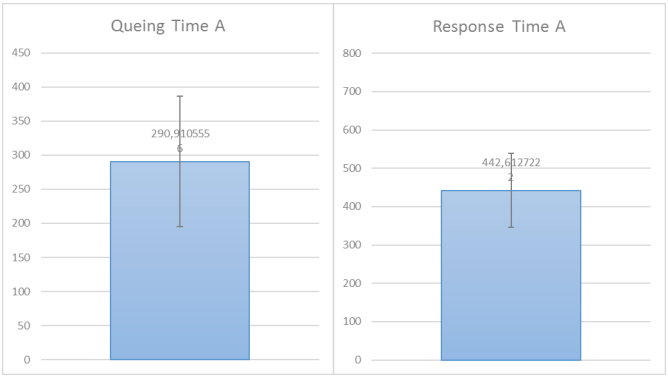

Figure 3.21: Average Queueing and Response times under policy a)

Values have also been summarize in the above tables:

E[W]	Confidence Interval	E[R]	Confidence Interval
290.91	[195.05; 386.76]	442.61	[346.17; 539.05]

- ***POLICY B***

warmup - period	sim-time-limit	repetition	λ	μ	N
27000	34200	36	1/20	150	9

The average queueing and response time of the system are the following:

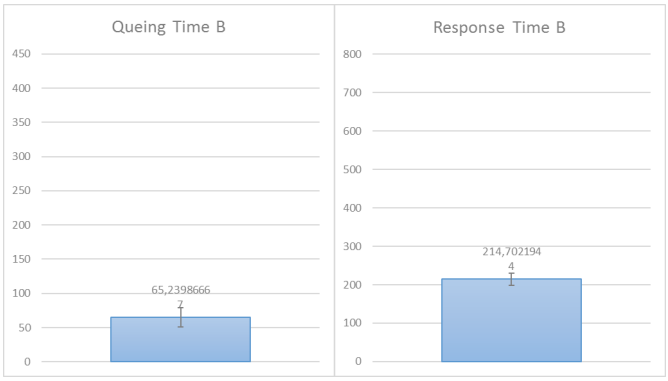

Figure 3.22: Average Queueing and Response times under policy b)

Summarized values are shown in these two tables:

E[W]	Confidence Interval
65.23	[51.08; 79.39]

E[R]	Confidence Interval
214.70	[19877; 230.62]

- ***Comparing policies***

E[Wa]	E[Wb]	Ratio (E[Wa]/E[Wb])	E[Ra]	E[Rb]	Ratio (E[Ra]/E[Rb])
290.91	65.23	4.45	442.61	214.70	2.06

These tables ensure that the average queueing time of the system under policy a) is about 4.45 higher.The average response time is instead 2.06 higher. This means that giving to the system this same workloads, policy b) outperforms policy a).

3.1.1.7 Sunday (λ=1/15)

Figure 3.23: **Simulating Sunday 5.30 p.m - 7.30 p.m**

Well let's now focus on the ***"critical" scenario***, the one in which we had really different system's behaviors according to the policy applied.We're going to simulate the scenario in which we have the highest peak in terms of inter-arrivals times.

We're going to split the system's analysis (under this scenario) by different values of n which represents the # of available tills. **Here we supposed inter-arrivals and service times are exponentially distributed with mean value equals to 15s and 150s respectively. For a given pair of** (λ, μ) **n has to verify the following disequation: n** $> \mu/\lambda$. **We started our analysis imposing n=11.**

- **CASE n=11**
- ***POLICY A***

  Let's try first to compute the system utilization: $\rho = \lambda/n\mu = 0.91$. In first approximation we may conclude that the system is positive recurrent (but we're above the knee). We modeled the system as an M/M/n system but we do not consider delays in order to reach anyone of open tills.This delay play a central role in the system's performance.

  Let's have a look on how the system's queueing time evolves in time:

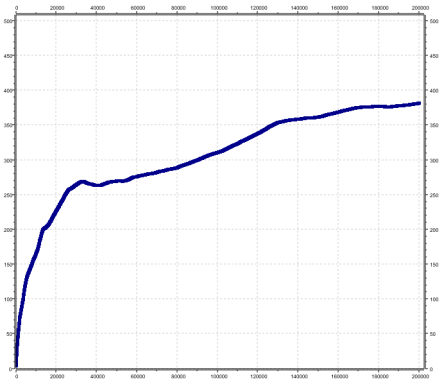

Figure 3.24: Queuing times under policy a) in case of "critical" scenario with n=11.

  The above pictures shows that queueing times increases linearly with time under this policy in this scenario and this means that **system won't reach a steady state, it will be always in the transient.**

- ***POLICY B***

  Does the same happen in case policy b) was applied? No it doesn't, at all!

The reason will be explained later on;so far we can concentrate our attention to the system's analysis:

warmup - period	sim-time-limit	repetition	λ	μ	N
250000	257200	36	1/15	150	11

The average queueing and response times of the system are the following:

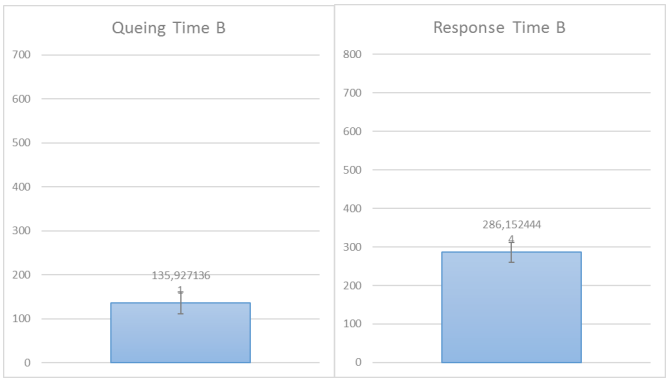

Figure 3.25: Average Queueing and Response times under policy b)

Summarized values are shown in these two tables:

E[W]	Confidence Interval
135.92	[111.13; 160.71]

E[R]	Confidence Interval
286.15	[260.48; 311.82]

- ***Comparing policies***

  The above results are really significant since they show that under policy a) system is in transient state while under policy b) system is not.The reason is always the same: delay in reaching open tills. We can think to this delay as one of the following (they are equivalent):

  - an higher queueing time, since in order to start being served each customer wastes some times in reaching the idle till;
  - an higher response time.We know that E[R] = E[W] +E[Ts] so for the same value of E[Ts] if E[W] increases E[R] will increase too.

  One possible solution to this situation is to increase the number of open tills. Why? Because as consequence we are decreasing system's utilization (remember that $\rho = \lambda/n\mu$). We proceeded in this way and we increment the number of open tills to 12.

- **CASE n=12**

- ***POLICY A***

  Now system utilization is decreased: $\rho = \lambda/n\mu$= 0.83. System now reaches a steady state. We set up our scenario as reported in table just below:

warmup - period	sim-time-limit	repetition	λ	μ	N
25000	32200	36	1/15	150	12

The average queueing and response times of the system are the following:

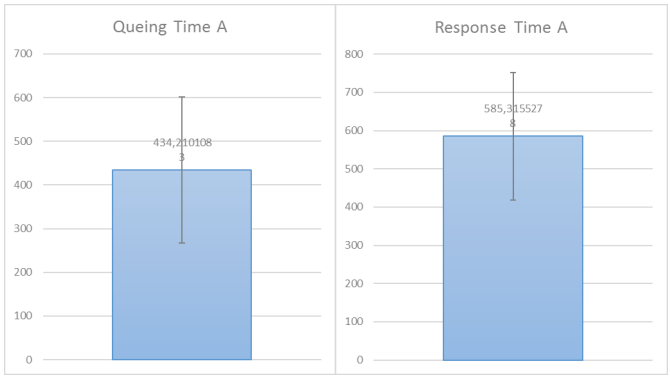

Figure 3.26: Average Queueing and Response times under policy a)

Summarized values are shown in these two tables:

E[W]	Confidence Interval
434.21	[267.04; 601.37]

E[R]	Confidence Interval
585.31	[418.06; 752.56]

- ***POLICY B***

warmup - period	sim-time-limit	repetition	λ	μ	N
300000	307200	36	1/15	150	12

The average queueing and response times of the system are the following:

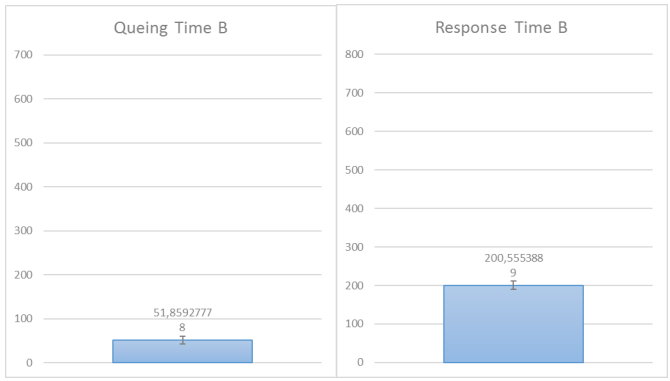

Figure 3.27: Average Queueing and Response times under policy b)

Summarized values are shown in these two tables:

E[W]	Confidence Interval
51.85	[42.91; 60.79]

E[R]	Confidence Interval
200.55	[189.87; 211.23]

- ***Comparing policies***

E[Wa]	E[Wb]	Ratio (E[Wa]/E[Wb])	E[Ra]	E[Rb]	Ratio (E[Ra]/E[Rb])
434.21	51.85	**8.37**	585.31	200.55	**2.91**

- **Under these considerably high workloads the average queueing time of the system under policy a) is almost 8.4 higher.The average response time is almost 2.9 higher.That's why we can state that policy b) outperforms policy a).**

3.1.2 Exponential Distribution Of InterArrival Times - Lognormal Distribution Of Service Demands

As we said at paragraph 2.2.2, we relied on some data provided by Google: this data show information about past visits at the Carrefour situated here in San Giuliano Terme (Pisa).Our idea was to use this data in order to calibrate in a real manner our simulations.We used them not only to setup different scenarios in the same day but also to retrieve some significant results between different days.The followings we're presenting are the most significant results we've found.They covered a whole week visit to the mentioned supermarket.

Data shown in the above tables summarize many aspects:

- Scenario Calibration;
- IID-ness of outputs;
- Confidence Intervals;

3.1.2.1 Monday (λ=1/25)

Figure 3.28: **Simulating Monday evening 5.30 p.m - 8.30 p.m**

- ***POLICY A***

warmup - period	sim-time-limit	repetition	λ	μ	σ	N	Δ
40000	50800	36	1/25	4.664061704	0.832554611	8	4s

We first verified the IID-ness of outputs data.

We based our study on the following assumptions:

Confidence Level	$z\alpha/2$	$\sqrt{n}$	$z\alpha/2/\sqrt{n}$
95%	1.96	6	0,326666667

The followings are the two correlograms used to verify the IID-ness of outputs data (queueing and response times):

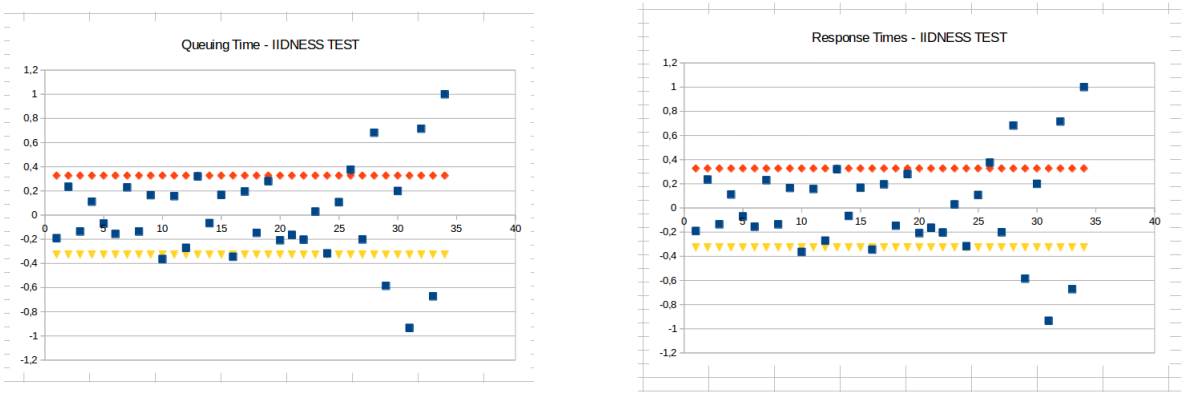

Figure 3.29: **Verifing IID-ness assumptions of output data in case of policy a) queueing times** .

From the above graphs the assumptions seems to be false, since not all auto-correlation coefficients are inside the bounds ±0,326666667. Anyway if we take a closer look we can see that these outliers are at large lags.Since autocorrelation function is less reliable due to the fact that the sum by which it is computed has too few values we cannot worry so much and state that the IID-ness of outputs data has met.

The average queueing and response time of the system are the following:

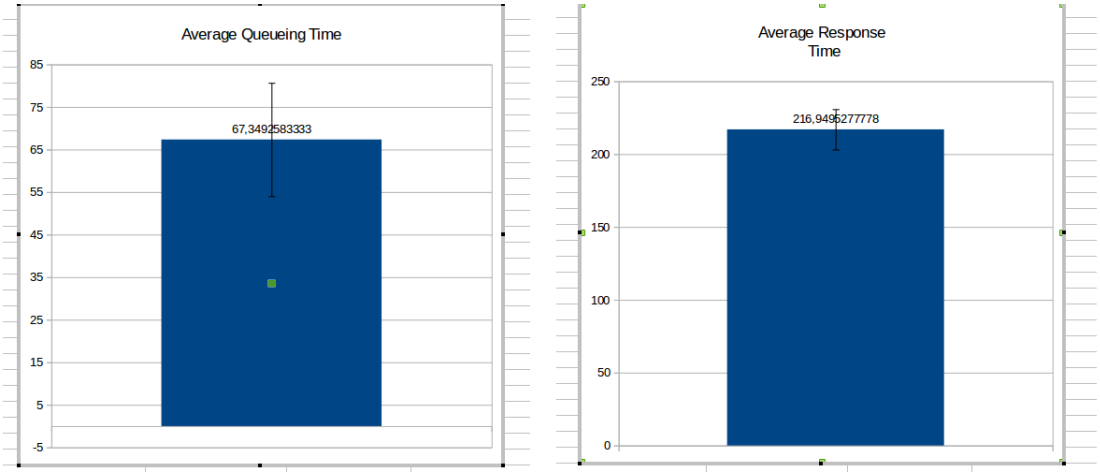

Figure 3.30: Average Queueing Time and Response times under policy a)

Values have also been summarize in the above tables:

E[W]	Confidence Interval	E[R]	Confidence Interval
67.35	[54,03; 80,67]	216.95	[203.08; 230.81]

- ***POLICY B***

warmup - period	sim-time-limit	repetition	λ	μ	σ	N
40000	50800	36	1/25	4.664061704	0.832554611	8

As we did for policy a) we first verified the IID-ness of outputs data based on the same assumptions.

The two correlograms for policy b) are the followings:

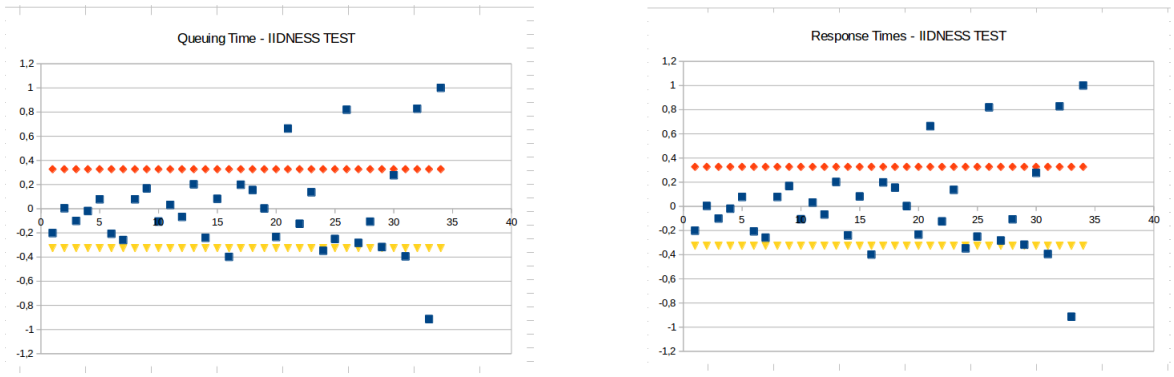

Figure 3.31: Verifying IID-ness assumptions of output data in case of policy b).

The average queueing and response time of the system are the following:

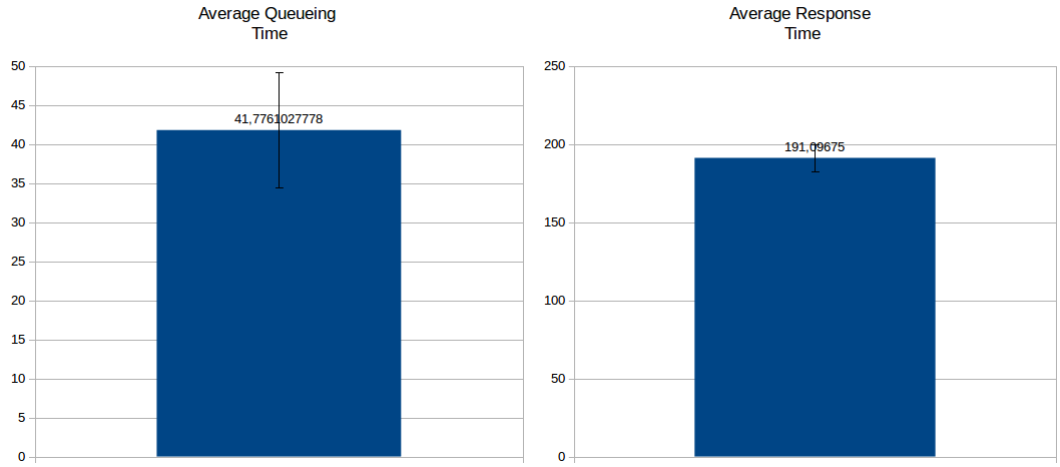

Figure 3.32: Average Queueing and Response times under policy b)

Summarized values are shown in these two tables:

E[W]	Confidence Interval
41.77	[34.40; 49.14]

E[R]	Confidence Interval
191.10	[182.22; 199.96]

- ***Comparing policies***

E[Wa]	E[Wb]	Ratio (E[Wa]/E[Wb])	E[Ra]	E[Rb]	Ratio (E[Ra]/E[Rb])
67.35	41.77	**1.61**	216.95	191.10	**1.13**

Since there's no overlapping in terms of confidence intervals, these tables ensure that the average queueing time of the system under policy a) is about 1.61 higher.The average response time is instead 1.13 higher. This means that giving to the system this same workloads, policy b) outperforms policy a).

3.1.2.2 Tuesday (λ=1/90)

Figure 3.33: **Simulating Tuesday morning 07.30 a.m - 9.30 a.m**

- ***POLICY A***

warmup - period	sim-time-limit	repetition	λ	μ	σ	N	Δ
50000	57200	36	1/90	4.664061704	0.832554611	2	4s

As in the previous case of study we first assessed the IID-ness of outputs data (still relying on the same assumptions).

Correlograms from now on will not be presented since the IID-ness of outputs data has been always met.In any case correlograms are available for consulting within the files attached to this documentation.

The average queueing and response time of the system are the following:

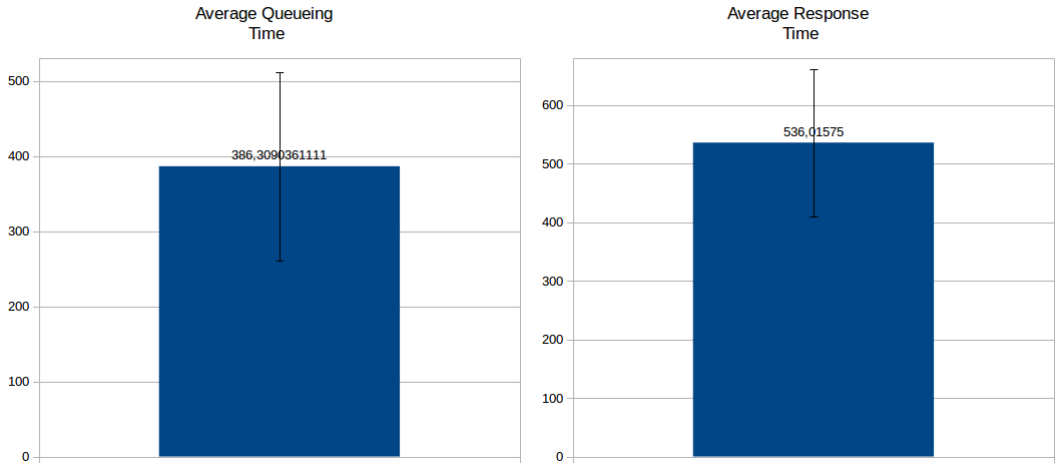

Figure 3.34: Average Queueing and Response times under policy a).

Values have also been summarize in the above tables:

E[W]	Confidence Interval	E[R]	Confidence Interval
386.31	[260.97; 511.65]	536.01	[409.41;662.62]

- ***POLICY B***

warmup - period	sim-time-limit	repetition	λ	μ	σ	N
40000	47200	36	1/90	4.664061704	0.832554611	2

The average queueing and response time of the system are the following:

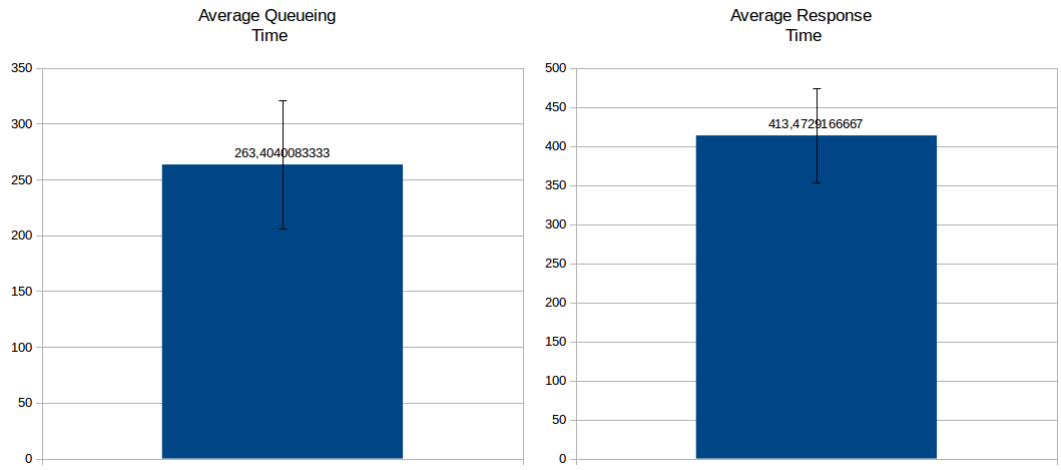

Figure 3.35: Average Queueing and Response times under policy b)

Summarized values are shown in these two tables:

E[W]	Confidence Interval
263.40	[205.98; 320.83]

E[R]	Confidence Interval
413.47	[353.34; 473.61]

- ***Comparing policies***

  Since there's overlapping in terms of confidence intervals, both for queueing and response times we cannot say nothing at all.This may be due to the higher confidence level (95%) so our idea was to decrease it and see if anything changes.

  Using a confidence level of 90% we get the following confidence intervals.

  Under **policy a)**:

Confidence Interval for E[Wa]	Confidence Interval for for E[Ra]
[281.11; 491,51]	[429.76; 642.28]

  Under **policy b)**:

Confidence Interval for E[Wb]	Confidence Interval for for E[Rb]
[215.21; 311,60]	[363; 463.94]

  We can conclude that since with a smaller confidence level (at 90%) we still observe overlapping, both for queueing and response times, anything meaningful can be stated.

3.1.2.3 First Thursday (λ=1/60)

Figure 3.36: **Simulating Thursday afternoon 10.30 a.m - 4.30 p.m**

- ***POLICY A***

warmup - period	sim-time-limit	repetition	λ	μ	σ	N	Δ
50000	75200	36	1/60	4.664061704	0.832554611	3	4s

The average queueing and response time of the system are the following:

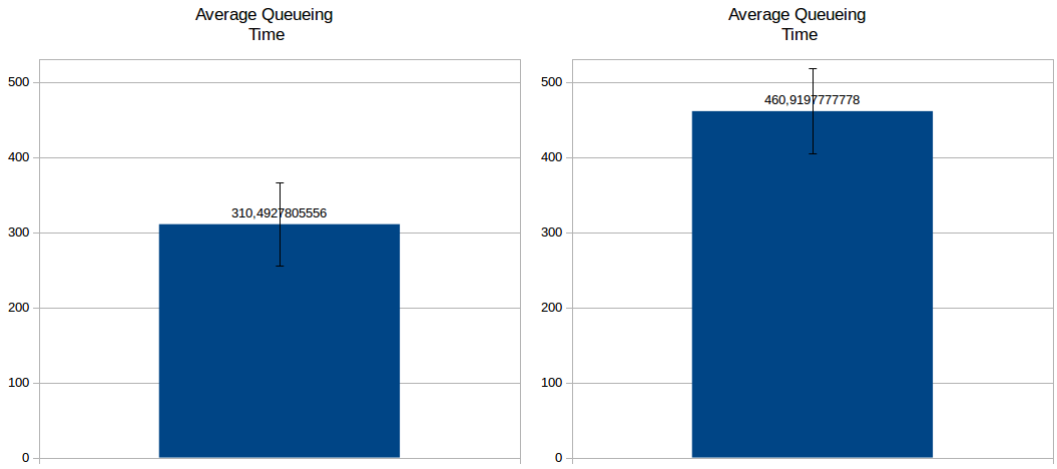

Figure 3.37: Average Queueing and Response times under policy a)

Values have also been summarize in the above tables:

E[W]	Confidence Interval	E[R]	Confidence Interval
310.43	[254.82; 366.16]	460.92	[404.37;517.47]

- ***POLICY B***

warmup - period	sim-time-limit	repetition	λ	μ	σ	N
50000	75200	36	1/60	4.664061704	0.832554611	3

The average queueing and response time of the system are the following:

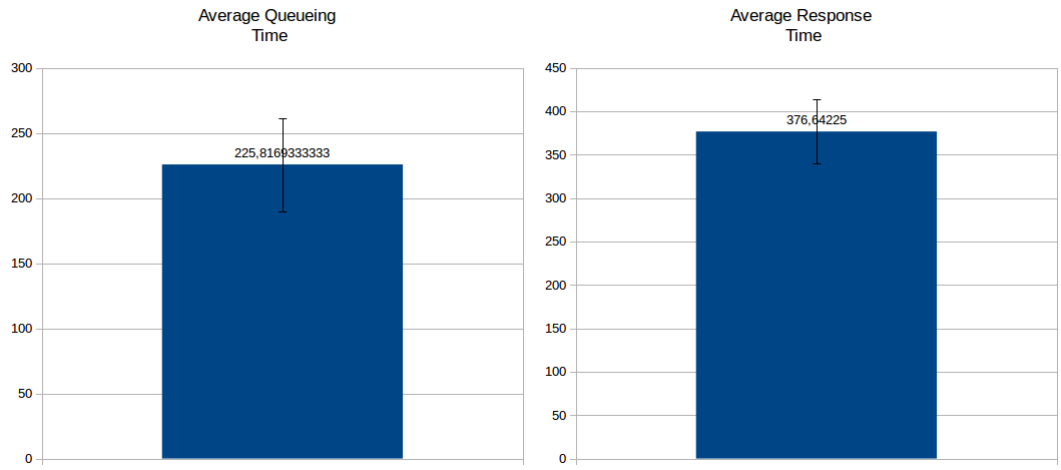

Figure 3.38: Average Queueing and Response times under policy b)

Summarized values are shown in these two tables:

E[W]	Confidence Interval	E[R]	Confidence Interval
225.82	[190.05; 261.58]	376.64	[339.92; 413.37]

In this case (as the previous one) because overlapping in terms of confidence intervals, both for queueing and response times, we cannot say which one of the policies performs better. We then decreased the confidence level to 90% as before and here what we observed:

Under **policy a)**:

Confidence Interval for E[Wa]	Confidence Interval for for E[Ra]
[263.77; 357.22]	[413.46; 508.38]

Under **policy b)**:

Confidence Interval for E[Wb]	Confidence Interval for for E[Rb]
[195.80;255.83]	[345.82; 407.46]

There's no overlapping in terms of confidence intervals, both for queueing and response times, so we can go one step ahead and compare the two policies under these (same) workloads:

- **Comparison between the two policies**:

E[Wa]	E[Wb]	Ratio (E[Wa]/E[Wb])	E[Ra]	E[Rb]	Ratio (E[Ra]/E[Rb])
386.31	263.40	**1.46**	536.01	413.47	**1.30**

Tables just above ensure that the average queueing time of the system under policy a) is about 1.46 higher.The average response time is instead 1.30 higher. This means that policy b) outperforms policy a) also in this case.

3.1.2.4 Second Thursday (λ=1/40)

Figure 3.39: **Simulating Thursday evening 4.30 p.m - 9.30 p.m**

- ***POLICY A***

warmup - period	sim-time-limit	repetition	λ	μ	σ	N	Δ
50000	68000	36	1/40	4.664061704	0.832554611	5	4s

The average queueing and response time of the system are the following:

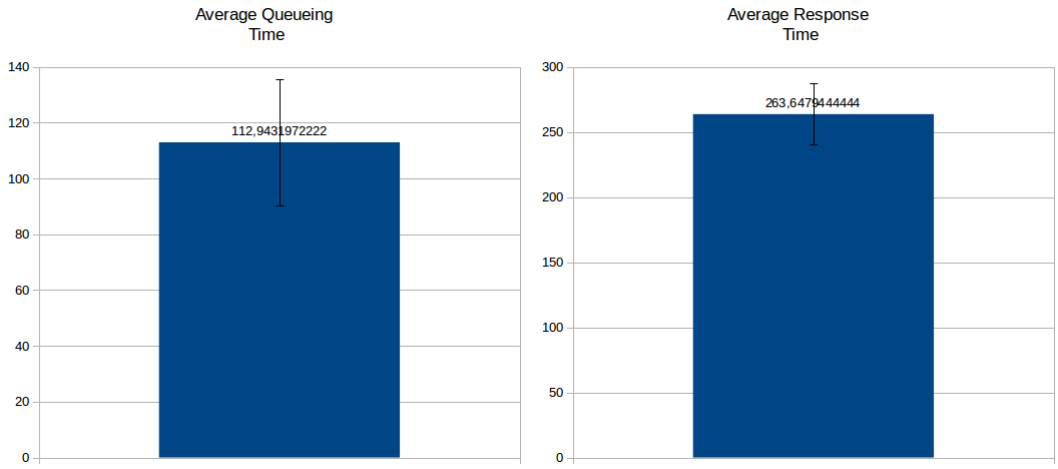

Figure 3.40: Average Queueing and Response times under policy a)

Values have also been summarize in the above tables:

E[W]	Confidence Interval	E[R]	Confidence Interval
112.94	[90.17; 135.72]	263.65	[240.10;287.20]

- ***POLICY B***

warmup - period	sim-time-limit	repetition	λ	μ	σ	N
30000	48000	36	1/40	4.664061704	0.832554611	5

The average queueing and response time of the system are the following:

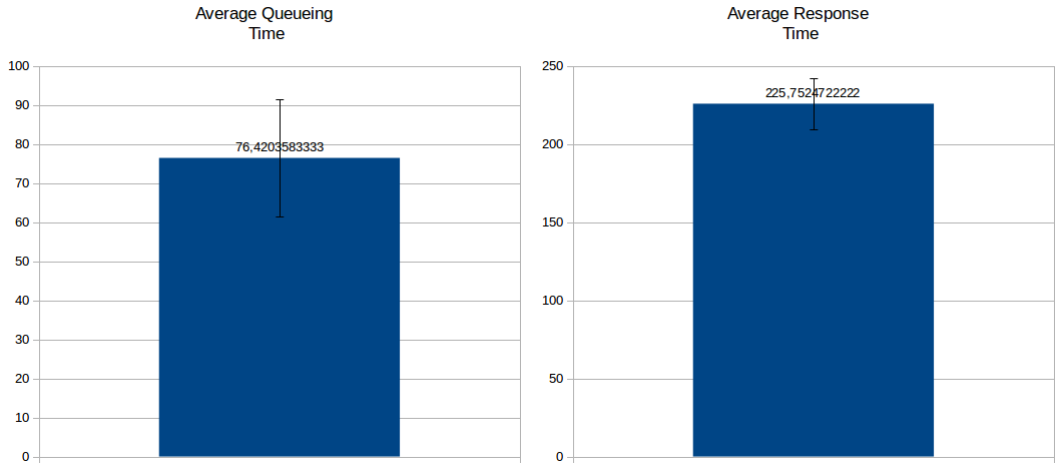

Figure 3.41: Average Queueing and Response times under policy b)

Summarized values are shown in these two tables:

E[W]	Confidence Interval	E[R]	Confidence Interval
76.42	[61.31; 91.53]	225.75	[209.45; 242.05]

In this case (as the previous one) because there's overlapping in terms of confidence intervals, both for queueing and response times, we can try to decrease confidence level to 90%.

Under **policy a)**:

Confidence Interval for E[Wa]	Confidence Interval for for E[Ra]
[93.83; 132.06]	[243.89; 283.41]

Under **policy b)**:

Confidence Interval for E[Wb]	Confidence Interval for for E[Rb]
[63.74;89.1]	[212.07; 239.43]

In this case (too) there's now no overlapping in terms of confidence intervals, both for queueing and response times, so we can compare the two policies:

- **Comparison between the two policies**:

E[Wa]	E[Wb]	Ratio (E[Wa]/E[Wb])	E[Ra]	E[Rb]	Ratio (E[Ra]/E[Rb])
112.94	76.42	**1.48**	263.65	225.75	**1.17**

These tables ensure that the average queueing time of the system under policy a) is about 1.48 higher.The average response time is instead 1.17 higher. This means that giving to the system this same workloads, policy b) outperforms policy a) (still in this case).

3.1.2.5 Saturday (λ=1/30)

Figure 3.42: **Simulating Saturday 10.30 a.m - 3.30 p.m**

- ***POLICY A***

warmup - period	sim-time-limit	repetition	λ	μ	σ	N	Δ
20000	38000	36	1/30	4.664061704	0.832554611	6	4s

The average queueing and response time of the system are the following:

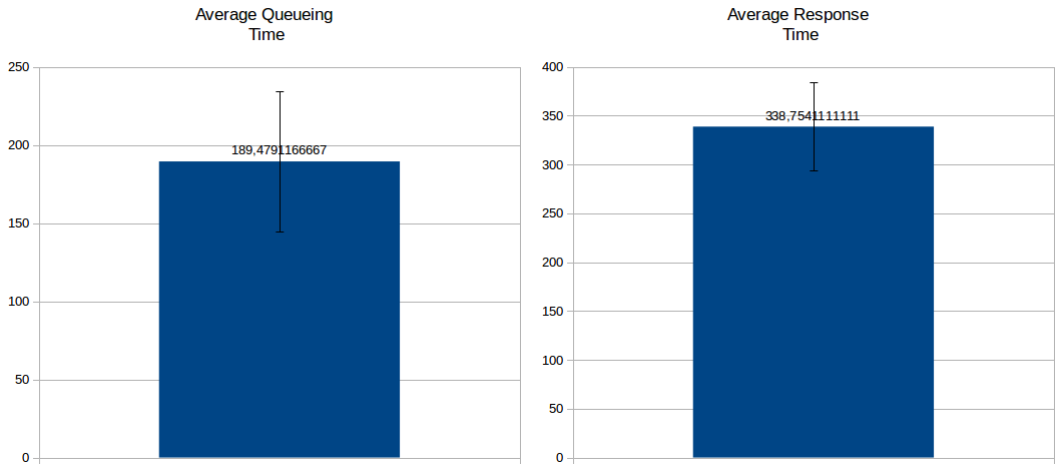

Figure 3.43: Average Queueing and Response times under policy a)

Values have also been summarize in the above tables:

E[W]	Confidence Interval	E[R]	Confidence Interval
189.48	[144.50; 234.50]	338.75	[292.42;385.058]

- ***POLICY B***

warmup - period	sim-time-limit	repetition	λ	μ	σ	N
20000	38000	36	1/30	4.664061704	0.832554611	6

The average queueing and response time of the system are the following:

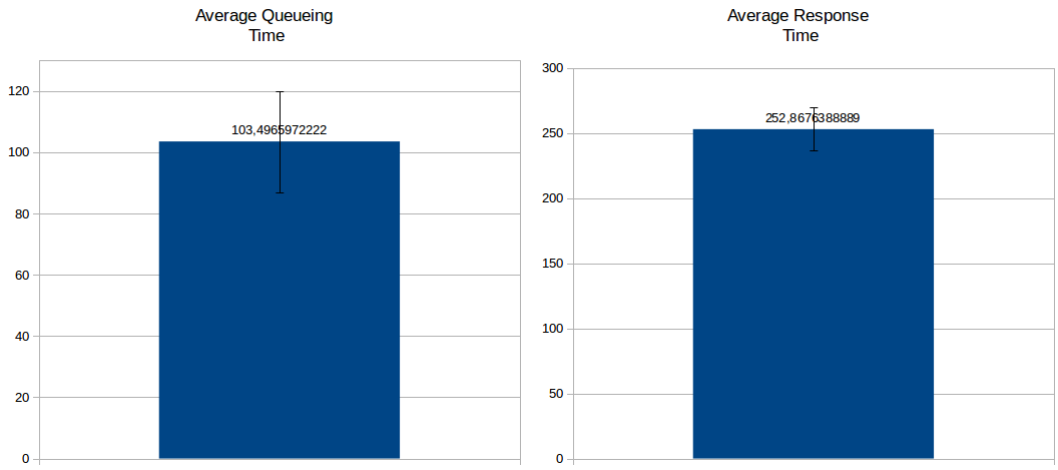

Figure 3.44: Average Queueing and Response times under policy b)

Summarized values are shown in these two tables:

E[W]	Confidence Interval
103.50	[87; 120]

E[R]	Confidence Interval
252.87	[234.80; 271]

- ***Comparing policies***

E[Wa]	E[Wb]	Ratio (E[Wa]/E[Wb])	E[Ra]	E[Rb]	Ratio (E[Ra]/E[Rb])
189.48	103.50	**1.83**	338.75	252.87	**1.34**

Tables just above ensure that the average queueing time of the system under policy a) is about 1.83 higher.The average response time is instead 1.34 higher. This means that policy b) outperforms policy a) also in this case.

3.1.2.6 Saturday (λ=1/20)

Figure 3.45: **Simulating Saturday 4.30 p.m - 6.30 p.m**

- ***POLICY A***

warmup - period	sim-time-limit	repetition	λ	μ	σ	N	Δ
40000	47200	36	1/20	4.664061704	0.832554611	9	4s

The average queueing and response time of the system are the following:

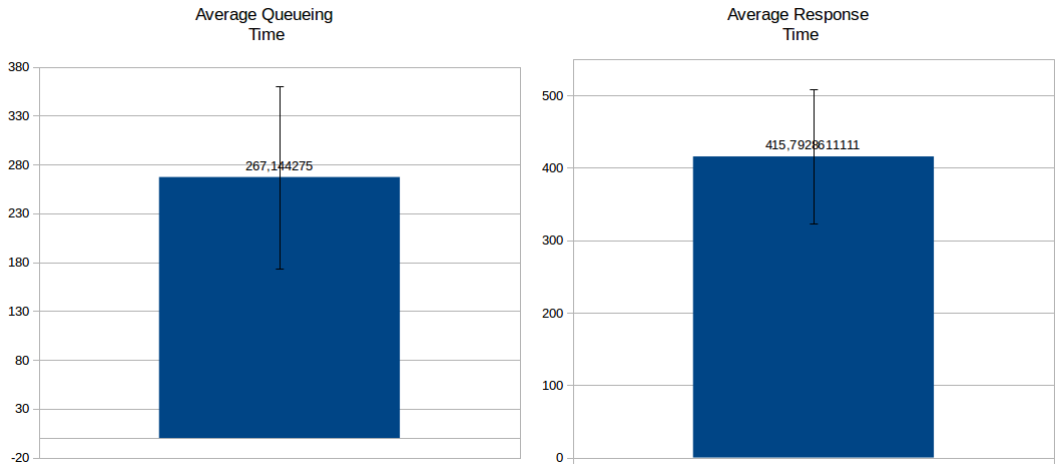

Figure 3.46: Average Queueing and Response times under policy a)

Values have also been summarize in the above tables:

E[W]	Confidence Interval	E[R]	Confidence Interval
267.14	[173.92; 360.38]	415.80	[320.92;510.67]

- ***POLICY B***

warmup - period	sim-time-limit	repetition	λ	μ	σ	N
20000	27200	36	1/20	4.664061704	0.832554611	9

The average queueing and response time of the system are the following:

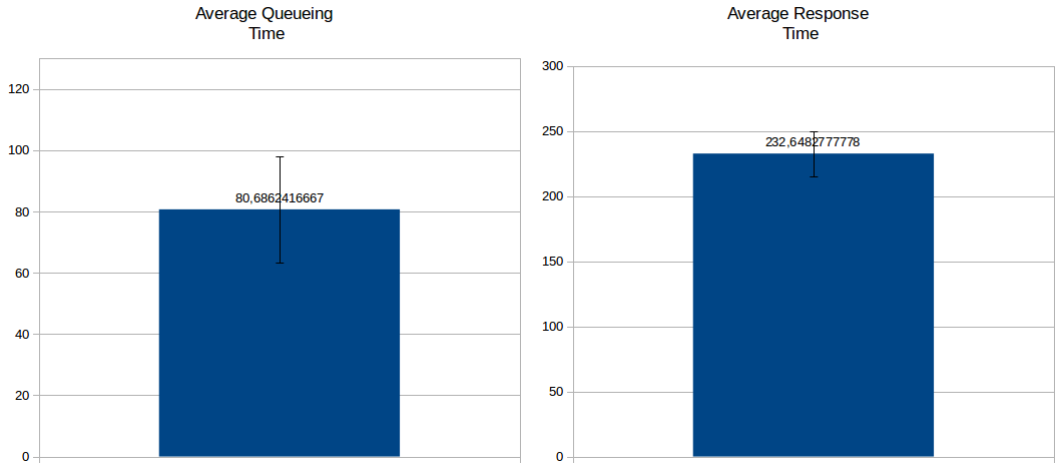

Figure 3.47: Average Queueing and Response times under policy b)

Summarized values are shown in these two tables:

E[W]	Confidence Interval
80.69	[63.28; 98.01]

E[R]	Confidence Interval
232.65	[213.20; 252.09]

- ***Comparing policies***

E[Wa]	E[Wb]	Ratio (E[Wa]/E[Wb])	E[Ra]	E[Rb]	Ratio (E[Ra]/E[Rb])
267.14	80.69	**3.31**	415.80	232.65	**1.78**

These tables ensure that the average queueing time of the system under policy a) is about 3.31 higher.The average response time is instead 1.78 higher. This means that giving to the system this same workloads, policy b) outperforms policy a).

3.1.2.7 Sunday (λ=1/15)

Figure 3.48: **Simulating Sunday 5.30 p.m - 7.30 p.m**

Well let's now focus on the ***"critical" scenario***, the one in which we had really different system's behaviors according to the policy applied.We're going to simulate the scenario in which we have the highest peak in terms of inter-arrivals times.

We're going to split the system's analysis (under this scenario) by different values of n which represents the # of available tills. **Here we supposed inter-arrivals and service times are exponentially distributed with mean value equals to 15s and 150s respectively. For a given pair of** (λ, μ) **n has to verify the following disequation: n** $> \mu/\lambda$. **We started our analysis imposing n=11.**

- **CASE n=11**
- ***POLICY A***

  Let's try first to compute the system utilization: $\rho = \lambda/n\mu$= 0.91. In first approximation we may conclude that the system is positive recurrent (but we're above the knee). We modeled the system as an M/M/n system but we do not consider delays in order to reach anyone of open tills.This delay play a central role in the system's performance.

  Let's have a look on how the system's queueing time evolves in time:

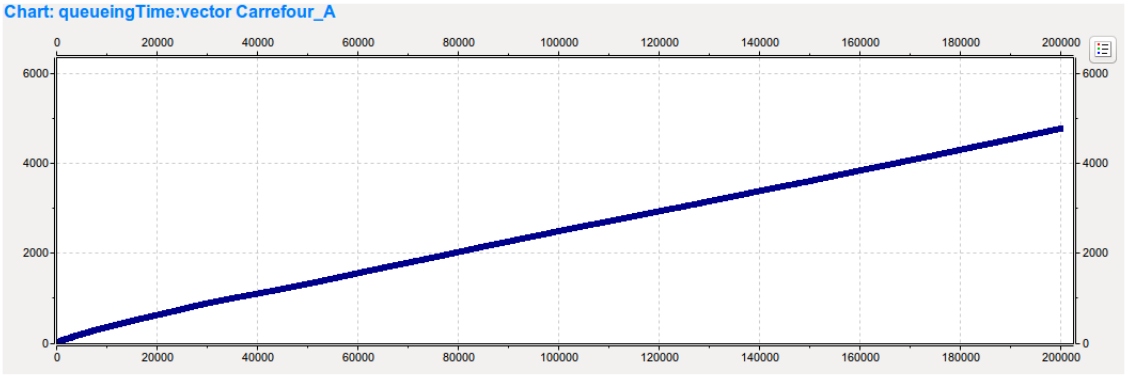

Figure 3.49: Queuing times under policy a) in case of "critical" scenario with n=11.

The above pictures shows that queueing times increases linearly with time under this policy in this scenario and this means that **system won't reach a steady state, it will be always in the transient.**

- ***POLICY B***

  Does the same happen in case policy b) was applied? No it doesn't, at all!

  The reason will be explained later on;so far we can concentrate our attention to the system's analysis:

warmup - period	sim-time-limit	repetition	λ	μ	σ	N
50000	57200	36	1/15	4.664061704	0.832554611	11

  The average queueing and response times of the system are the following:

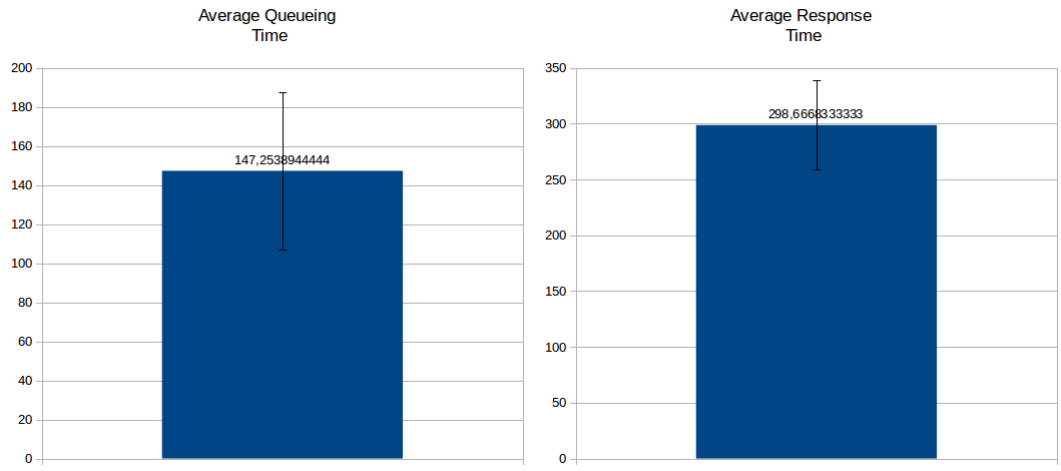

Figure 3.50: Average Queueing and Response times under policy b)

  Summarized values are shown in these two tables:

E[W]	Confidence Interval
147.26	[107.18; 187.33]

E[R]	Confidence Interval
298.66	[257; 340.34]

- ***Comparing policies***

  The above results are really significant since they show that under policy a) system is in transient state while under policy b) system is not.The reason is always the same: delay in reaching open tills. We can think to this delay as one of the following (they are equivalent):

  - an higher queueing time, since in order to start being served each customer wastes some times in reaching the idle till;
  - an higher response time.We know that E[R] = E[W] +E[Ts] so for the same value of E[Ts] if E[W] increases E[R] will increase too.

  One possible solution to this situation is to increase the number of open tills. Why? Because as consequence we are decreasing system's utilization (remember that $\rho = \lambda/n\mu$). We proceeded in this way and we increment the number of open tills to 12.

- **CASE n=12**

- ***POLICY A***

  Now system utilization is decreased: $\rho = \lambda/n\mu$= 0.83. System now reaches a steady state. We set up our scenario as reported in table just below:

warmup - period	sim-time-limit	repetition	λ	μ	σ	N
60000	567200	36	1/15	4.664061704	0.832554611	12

  The average queueing and response times of the system are the following:

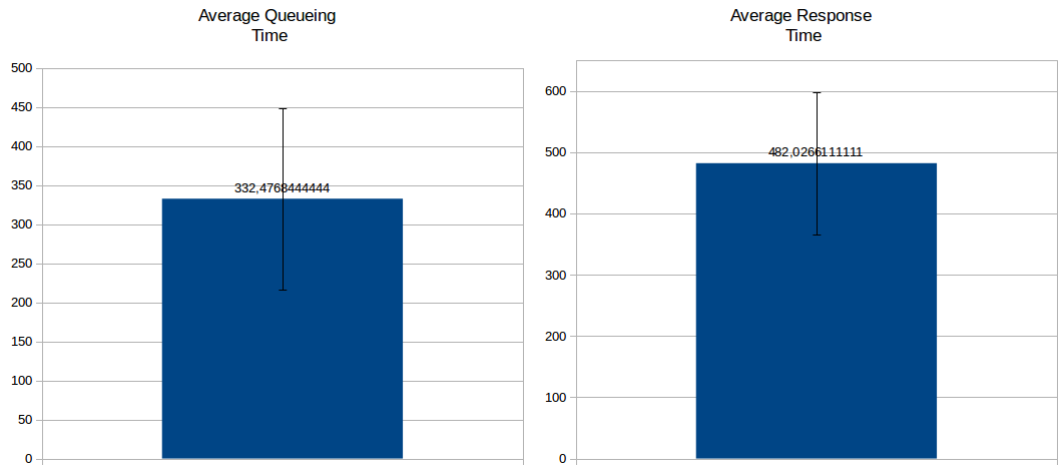

Figure 3.51: Average Queueing and Response times under policy a)

Summarized values are shown in these two tables:

E[W]	Confidence Interval	E[R]	Confidence Interval
332.48	[216.50; 448.45]	482.03	[365.83; 598.22]

- ***POLICY B***

warmup - period	sim-time-limit	repetition	λ	μ	σ	N
50000	57200	36	1/15	4.664061704	0.832554611	12

The average queueing and response times of the system are the following:

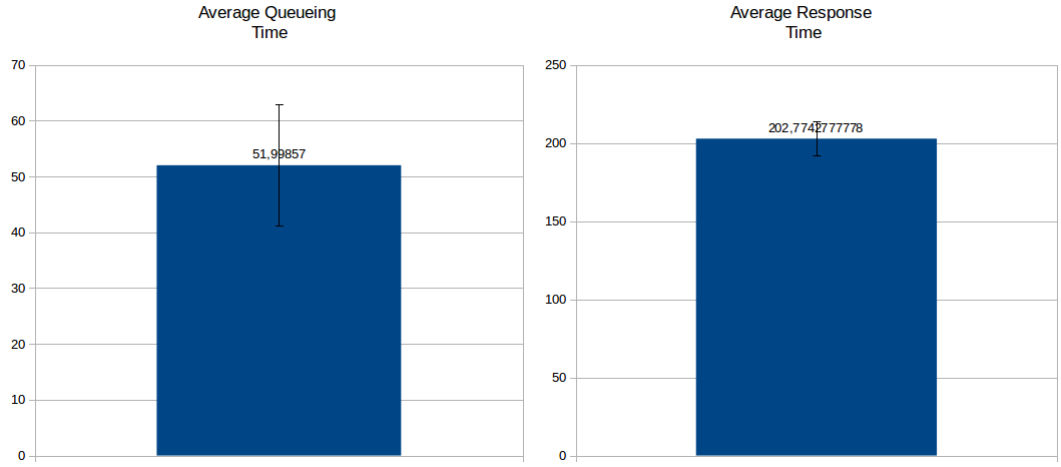

Figure 3.52: Average Queueing and Response times under policy b)

Summarized values are shown in these two tables:

E[W]	Confidence Interval	E[R]	Confidence Interval
52	[41.16; 62.84]	202.77	[190.1; 215.46]

- ***Comparing policies***

E[Wa]	E[Wb]	Ratio (E[Wa]/E[Wb])	E[Ra]	E[Rb]	Ratio (E[Ra]/E[Rb])
332.48	52	**6.39**	482.03	202.77	**2.38**

- **Under these considerably high workloads the average queueing time of the system under policy a) is almost 6.5 higher.The average response time is almost 2.5 higher.That's why we can state that policy b) outperforms policy a).**

4 Conclusion

We are going to summarize results and insight obtained in the analysis exposed.

The first consideration is that: **policy A is worse than B in each scenario we simulated**.This was easy to predict because the *"connection delay"* associated with each servers in A can be seen as an additional constant service time associated to servers. We end up with two system, one of which behaves according to policy A and the other according to policy B, having the same number of servers in what we called the same "*critical scenario*".

Policy B outperforms policy A: as soon as the number of available servers increases this gap increases too. **Why do we increase the number available of servers? This happens because as soon as the incoming flow of customers increase that means as soon as the expected value of the exponential distribution of interarrival times decrease, in order to have a positive recurrent system (ρ<1) we need to increase n. Under policy a) when n increases,the delay (Δ) has a big impact on system's performance. Once again we can think to this delay in this terms:**

- In order to start being served each customer has to reach the first idle till, suppose it's till #n. **This requires a time** $\Delta * n$, and this means that as **n increases this time will increase too**.

Anyway it could be a scenario (infrequently) where policy a) take advantages on b):

- suppose a customer at some given point in time has to choose one till between the ones having the same minimum number of elements and suppose this number is equal to 1 (this means that the one and only one customers is being served). The choice is perfectly arbitrary so he/she may encountered with not the best one. It could happen that one till among the eligible ones finishes to serve before the one selected and in this case what we have is that our customer is experimenting queueing time but one till is idle. In a real life scenario the customer may decide to abandon that queue to immediately be served by the idle cashier. If policy a) was applied (instead) we hadn't got the same problem and at the same a) takes advantages on b).Giving the fact this difference may arise only under these particular circumstances and only in case the elected till is not the one at which cashier will terminate demand first and since the system's overall queueing and response time will not be affected very much by these episodes we can just state that this contribute is negligible.