UNIVERSITÁ DI PISA
ENGEGNERIA DELL'INFORMAZIONE

Project 11 – The Carrefour

Bruno Augusto Casu Pereira de Sousa

Master's degree Computer Engineering – Computer Systems and Networks
592II Performance Evaluation of Computer Systems and Networks

Pisa, 2022

Contents

1. Introduction

The Carrefour Project is a study case of the supermarket checkout organization, that is, how the clients queue up at the tills, and can process the products they wish to buy. The Project aims to provide a simulator that defines two policies for the clients to perform the Checkout, and to evaluate the impact of different approaches on how to process the clients. The scenarios will be tested with different workloads and the results will be analyzed considering the most important Key Performance Indicators, such as the Average Checkout time and the Average Queue Time. These parameters will then define how well the different policies affect the quality of service of the supermarket simulated in the project.

The software used for the simulation is OMNet++, as it provides a proper framework to implement queues and processing centers, necessary for the representation of the supermarket checkout system. The data provided by the simulations will also be analyzed and processed using Microsoft Excel, to display the results in a comprehensive manner, as well as to produce relevant plots of the results.

2. System Description

To model the supermarket actors (clients, tills, and the queues), the concepts of Queueing theory will be used to provide a better view of the characteristics and behavior of those actors, providing a model of a queueing network. The simplest design of the supermarket checkout policy can be defined as a Source sending Clients (jobs) to a Queue, where they wait to be Processed at the Till. This simple network is illustrated in Figure 1:

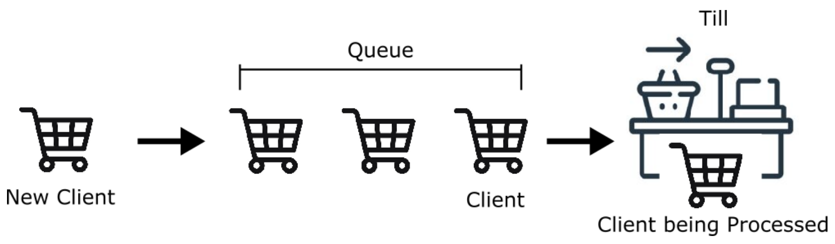

Figure 1 – Modeling of the simplified supermarket checkout process

The Clients will arrive from the source within a random interval, modeled as an Independent Random Value. The rate that those clients arrive at the queue is represented by λ. In this way the time value between each client arrival is represented by $1/\lambda$ (measured in seconds), as the Interarrival Time (IA). As the clients arrive for the Checkout, they will enter a First In First Out (FIFO) type Queue and will be placed at the end of it.

The till will be modeled as a processing unit (Service Center, SC), and will perform the client processing in a time interval (also modeled as an independent random value). The client processing rate is defined by μ. In this way, the time to process a client (PT) is then represented as $1/\mu$**.** The queueing network described for this system is then illustrated in Figure 2:

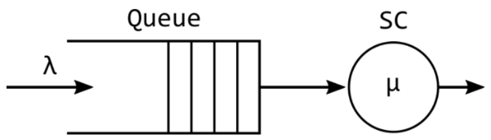

Figure 2 – Modeling of the simplified supermarket checkout process as a queueing network

From the queueing system described, it is possible to compute and evaluate the Key Performance Indicators of the supermarket and the characteristics that influence on the quality of service provided. The evaluated parameters will then be the Mean Queueing Time (**E[W]**), that is, how much time on average does a client spent on the queue, the Maximum Queue Size (max. **E[Nq]**), and the Mean System Response Time (**E[R]**), that is, the time between a client arriving at the system and leaving it, after the processing in the till. Also, an important parameter to be evaluated is the stability condition of the system. In the case of the described network, the condition is calculated by:

$$u = \lambda/\mu$$

2.1 Scenario A Description

The first policy to be described, is referred as **Policy A**, and uses a single queue to send the clients to N tills in the system. Clients will be placed at a FIFO type queue and forwarded to a free till in the system when any is available. Also, the clients must go through a certain distance to reach the target till, where it will be in idle while the client moves towards it.

For this reason, the till assignment to the client at the head of the queue will prioritize sending the client to the nearest available till. The distance is proportional to the till number **j** (till number 0 is the nearest, and till number N is the furthest). The time to reach the first till is based on a fixed time value **Δ**. For the next tills, the interval **Δ** is multiplied by the till number providing a linear increase in the time to reach the free till:

$$t = (j + 1) \times \Delta$$

The above-mentioned characteristics for the Policy A system are illustrated in Figure 3:

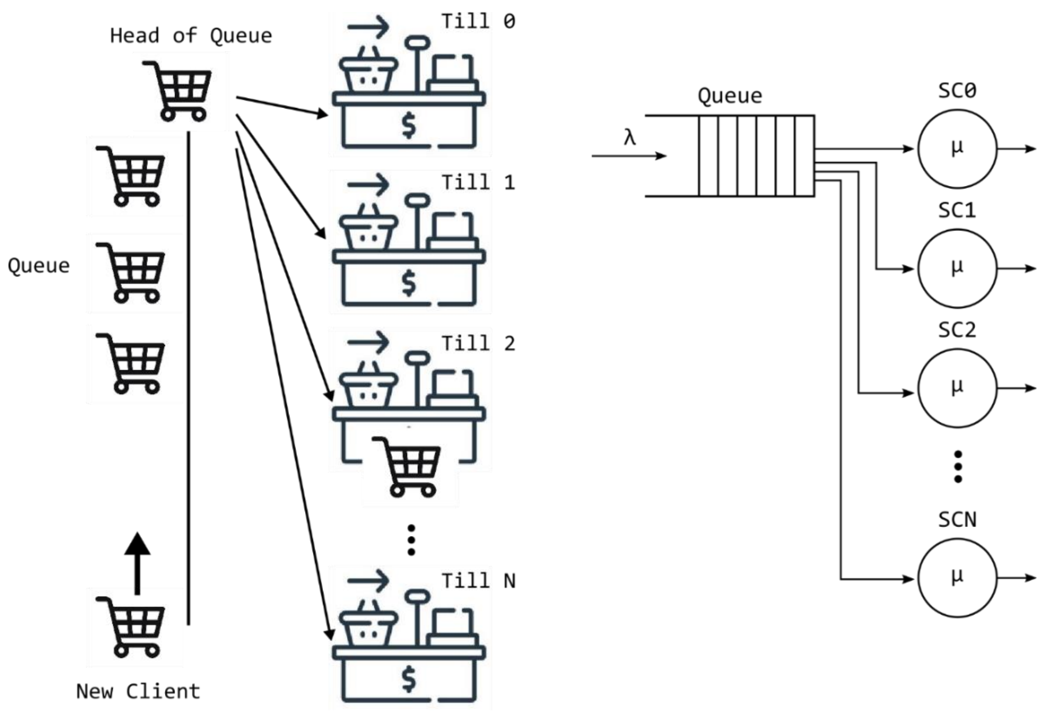

Figure 3 – Diagrams for modeling the Checkout Policy A

Given the modeled system and its characteristics, the continuous-time Markov chain (CTMC) of the implemented network can describe the expected behavior for the system. It is also important to consider that the value for the processing time ($1/\mu$) is composed of different time constraints that are present in the system. Therefore, the processing time is based on the average time to reach the till, a constant processing time (added to replicate a constant characteristic of processing a client at a till, such as printing the receipt) and the RV representing the client demand (proportional to the number of items a client is buying, and consequentially, influencing the time needed to process all the items). The CTMC illustrated represents an M/M/N queueing system, where N are the number of tills in the supermarket:

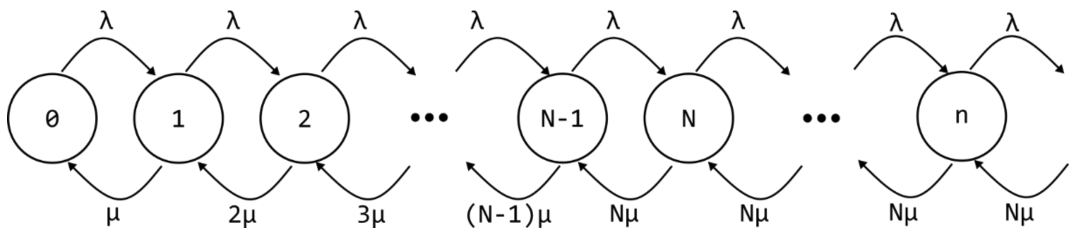

Figure 4 – CTMC for Checkout Policy A

Considering the steady state behavior of the system, it is expected a stable behavior when $u < 1$, where u in an M/M/N system is defined as the usage ρ , and that can be computed by:

$$\rho = \lambda/(N \times \mu)$$

As previously mentioned, two other time values will be added as way to simulate a system more realistic, they are the delay to reach the till, and the constant value for the checkout. Considering these factors, the following equation can be computed to evaluate the stability of a Policy A checkout system with N Tills:

$$IA > PT/N$$

where IA is the client mean interarrival time in the queue (modeled by an IID RV), and PT is the processing time, computed by:

$$PT = K + \left(\frac{\Delta(1+N)}{2}\right) + RV$$

In the Processing Time equation, K is the constant time value (added to improve the simulation consistency with a real system), RV is the mean value for the user demand and the middle term is the average time to reach an empty till in the system. As an example of a practical scenario, consider a system with 10 tills, a Δ value of 2 seconds, and that the user demand is an Exponential RV with a mean value of 240 seconds. The constant processing time is defined to be 30 seconds, so the computed total processing is:

$$PT = 30 + \left(\frac{2(1+10)}{2}\right) + 240 = 281s.$$

The computed average time to reach the till has a value of 11 seconds, as the client takes 2 seconds to reach the first till and 20 seconds to reach the last till. From the stability condition we can then infer that for this system configuration, the queue will not grow infinitely if the client interarrival time behaves as an exponential random variable with a mean value $IA > 28{,}1s$.

For the metrics to evaluate the performance, we can compute, based on the system characteristics (M/M/N queueing system) the E[R] (system average response time, equal to the checkout time), E[W] (mean wanting time, or time spent in the queue) and E[N_q] (mean number of clients in the queue). For the waiting time, we use the following equations, also considering the number of clients in the queue and the probability of no jobs in the system (p_0):

$$E[N_q] = \frac{u^N}{N!} \times p_0 \times \frac{\rho}{(1-\rho)^2} \qquad E[W] = \frac{E[N_q]}{\lambda}$$

Considering now the response time of the described system, the mean value can be computed by adding the average waiting time to the average processing time (mean service time described as $E[t_s]$):

$$E[R] = E[W] + E[t_s] \qquad E[t_s] = 1/\mu$$

Noticeably from the equation, the computed average response time is based on the actual time to process the client ($E[t_s]$), that is correspondent to an Independent RV, and the average queue time (waiting time).

2.2 Scenario B Description

Following the description of the first modeled scenario for the checkout policy, the next scenario changes the organization of the tills and the queue. In Scenario B, the policy is to make a queue for each till in the system, where clients join to be processed. To approximate the system to a real scenario, the clients will always choose the queue with less clients on it and will choose randomly if there is a tie. The described organization is illustrated in Figure 5:

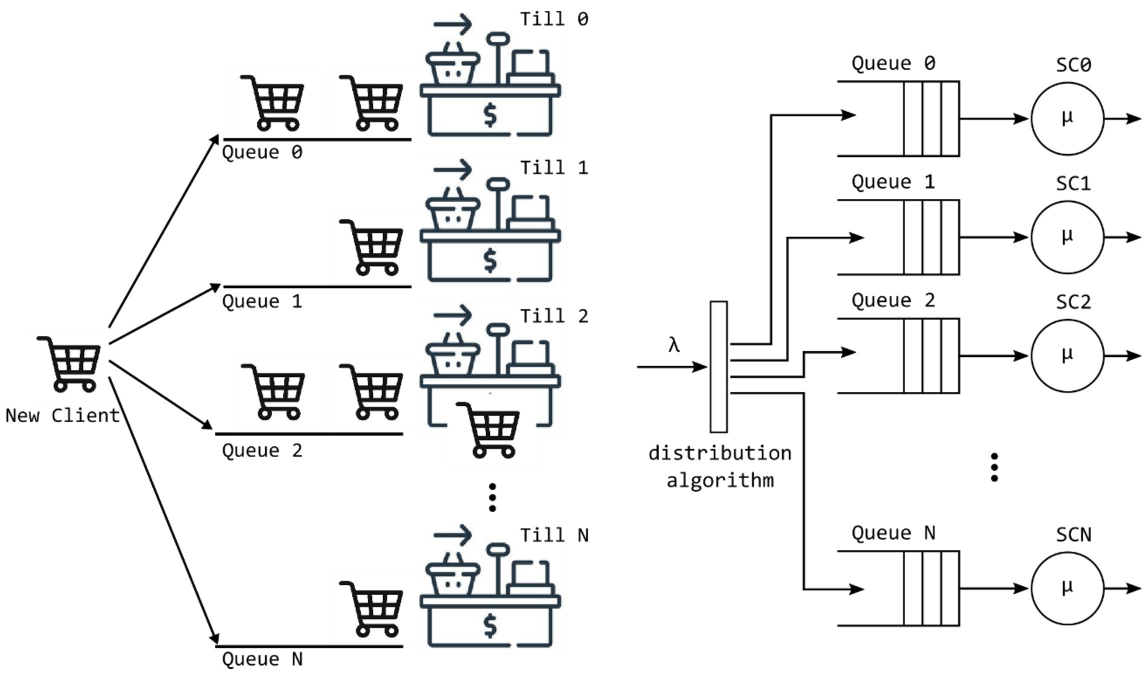

Figure 5 – Diagrams for modeling the Checkout Policy B

From the specified characteristics for this model, the main difference in the system from the previous is that each individual till will behave as a M/M/1 queueing system.

This then implies that the clients will be distributed to the available queues, depending on the status of the individual systems (queue-till pair). Since all individual systems behave the same (queue growth and processing time, considering independent RVs for this parameters), the clients will be uniformly distributed in all tills, and its respective queues. The effect that this modeling produces is that the client arrival rate for each queue is then evenly divided by the number of tills in the system. Therefore, the individual client arrival rate at the individual queues in a system with N tills is λ/N.

This maintains the previously stability condition inferred for the Scenario A as $IA/N > PT$, in spite the different influence that the system parameters may produce in the overall response time. For Scenario B the utilization for each individual till must be $\rho < 1$ to maintain the queue under control, considering now that $\rho = \lambda/(N \times \mu)$.

Considering only an individual queue and till pair in the system, the CTMC for this subsystem can be described as in Figure 6:

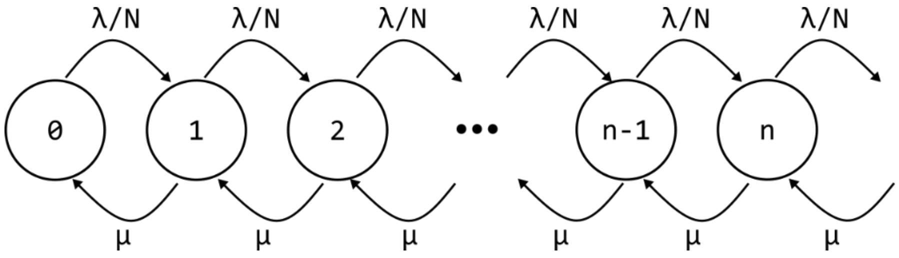

Figure 6 – CTMC for Checkout Policy B (on each individual queueing subsystem)

For the performance metrics in this scenario, the average response time of one individual checkout subsystem in the supermarket, considering that N tills are available, can be calculated as:

$$E[R] = \frac{1}{\mu - \lambda/N}$$

As for the waiting time in the queues, the mean value can be computed as a function of the response time:

$$E[W] = E[R] - \frac{1}{\mu}$$

3. Simulator Development

The modeling of the supermarket checkout policies will be made through a C++ code in the OMNet++ framework. The modeling aims to provide a simulator for the checkout policies, and to reproduce the behavior of the developed systems. The configuration of each policy is done by three different modules in the framework, as well as setting the desired parameters for the system workload and usage, and the connections between the modules.

From a code perspective, each scenario will then have a Source module, a Queue module and a Till module. The Source then provides the clients for the system using an Exponential Independent and Identically Distributed Random Variable, or Expo. IID RV. The Queue Module then receives the clients form the Source (managed using the message interface from OMNet++) and handles the forwarding process to send the clients to the tills.

Finally, a Till module will be used as a sink for incoming messages, providing a trigger message that simulates the client processing, again based on a IID RV. For both scenarios the maximum number of tills was set to 10.

3.1 Scenario A Development

Starting with the development of the simulator for the scenario A, the Source module must then serve on single queue in the system. A simple messaging module was the developed to send periodic messages to the Queue module. Messages are generated and sent using the framework functions.

On the network configuration of this module, a parameter will represent the client interarrival time, by generating a message in a random time value. In Figure 7, is possible to see the number of clients that reach the queue with their correspondent IA time. Noticeably the histogram resembles the exponential curve expected, and most of the arrivals tend to be at values between 0 to 30 seconds. In the plot the y axis represents the number of clients, and the x axis the IA time:

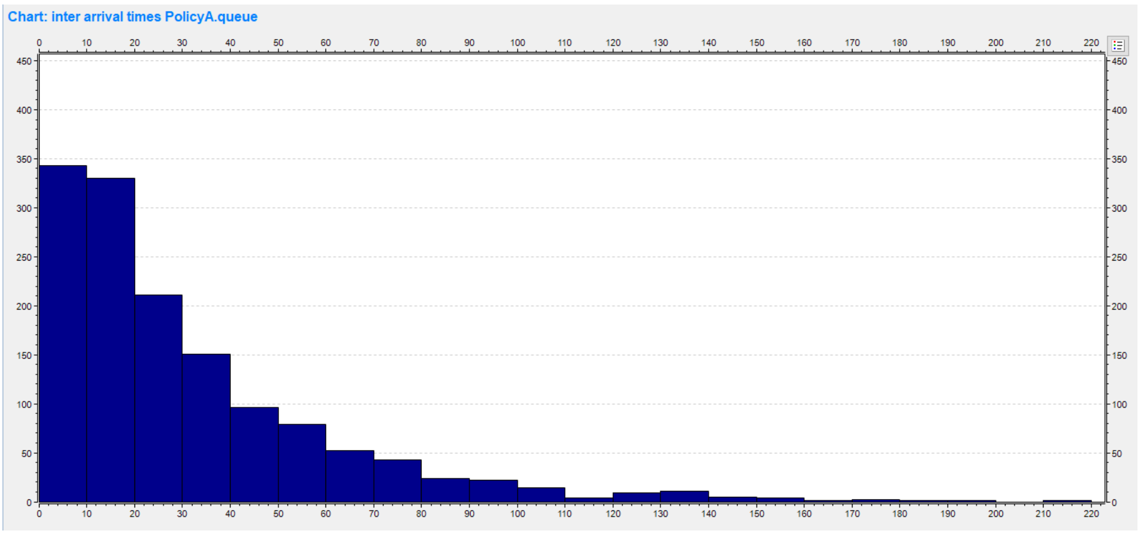

Figure 7 – IA time histogram of generated messages by the Source module (mean value = 30s)

Receiving the generated messages, the Queue module then performs, in this scenario, most computations necessary to simulate the checkout process. As the system only has one queue, it will receive the messages and execute a search algorithm to find any empty tills in the system. As the algorithm completes the search, if it finds one or more available tills, it will generate a message for the target till number. Also, if two or more tills are empty when the search algorithm is executed, the closest till will be given priority in the choice (the distance is proportional to the till number, the higher number, the furthest). After the decision, a message to the target till will be sent with a delay, based on the till number, as specified in the project requirements.

In case of all tills being used, the queue will increase the queue size parameter, and mark the entry time of this client to be used in the estimation of the Waiting time (Queue time).

A particular characteristic of the code developed, is that all tills are always deployed and connected to the queue. However, a parameter is set inside the queue module, limiting the number of possible queues to be used in a simulation. This is then used to change the system capacity according to the test requirements.

Another important parameter used to maintain the system consistency is the actual till number, sent in a custom type of message from the queue to all the tills. This feature was introduced as at the beginning of the simulation, the till modules do not know its own number.

Then, when the tills start receiving clients, the queue sends the till number information on the messages. With this, the Till module can set this parameter as it own, allowing the replies from the till to the queue to be identified.

Finally, the queue module is also responsible for collecting the simulation results and save them in vectors and histogram to be used for the performance evaluation. The general state diagram for all this processing inside the Queue module of scenario A is illustrated in Figure 8:

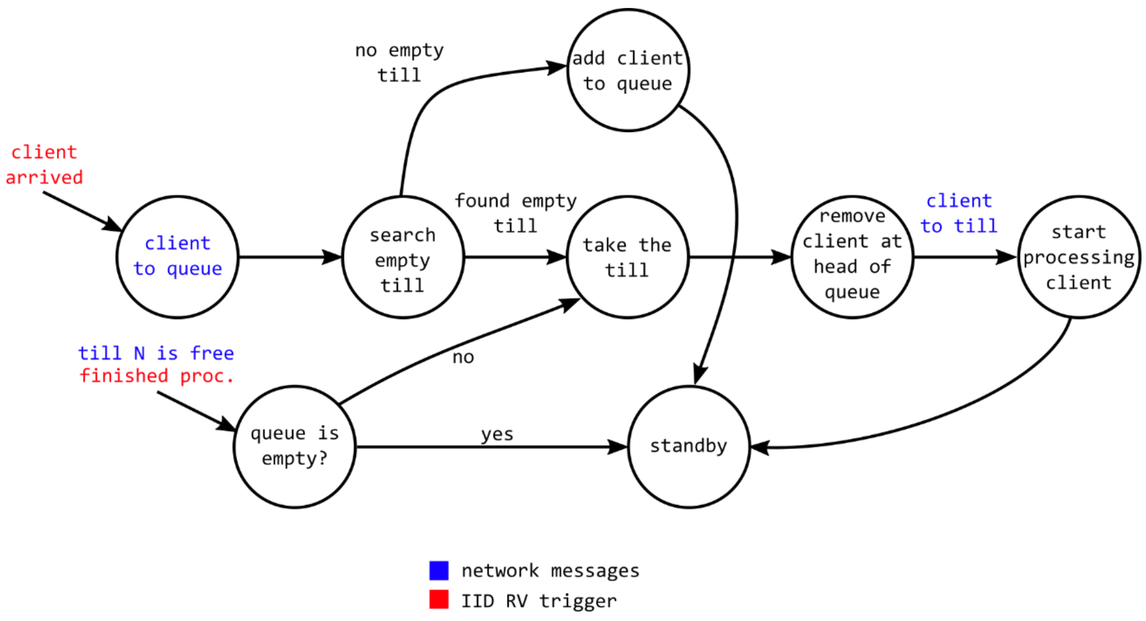

Figure 8 – Diagram of checkout policy A states handled by the Queue module

In the diagram it is possible to see the important state transactions and the messages exchanged in the process. The “till N is free” is the referred message from one of the tills to inform the queue that it is now available. Also, the “standby” state is an abstraction, as in this state, the system may or may not be processing clients. The importance of this state is only to show the system behavior when the client forwarding is completed, and there are no other actions to be performed. Considering this, the system is triggered to exit the standby state when the messages created by the modules are sent or received (which are tied to the RV generation).

To better detail the messages exchanged in the scenario A implementation, Figure 9 shows an example of a simulation run using 3 tills:

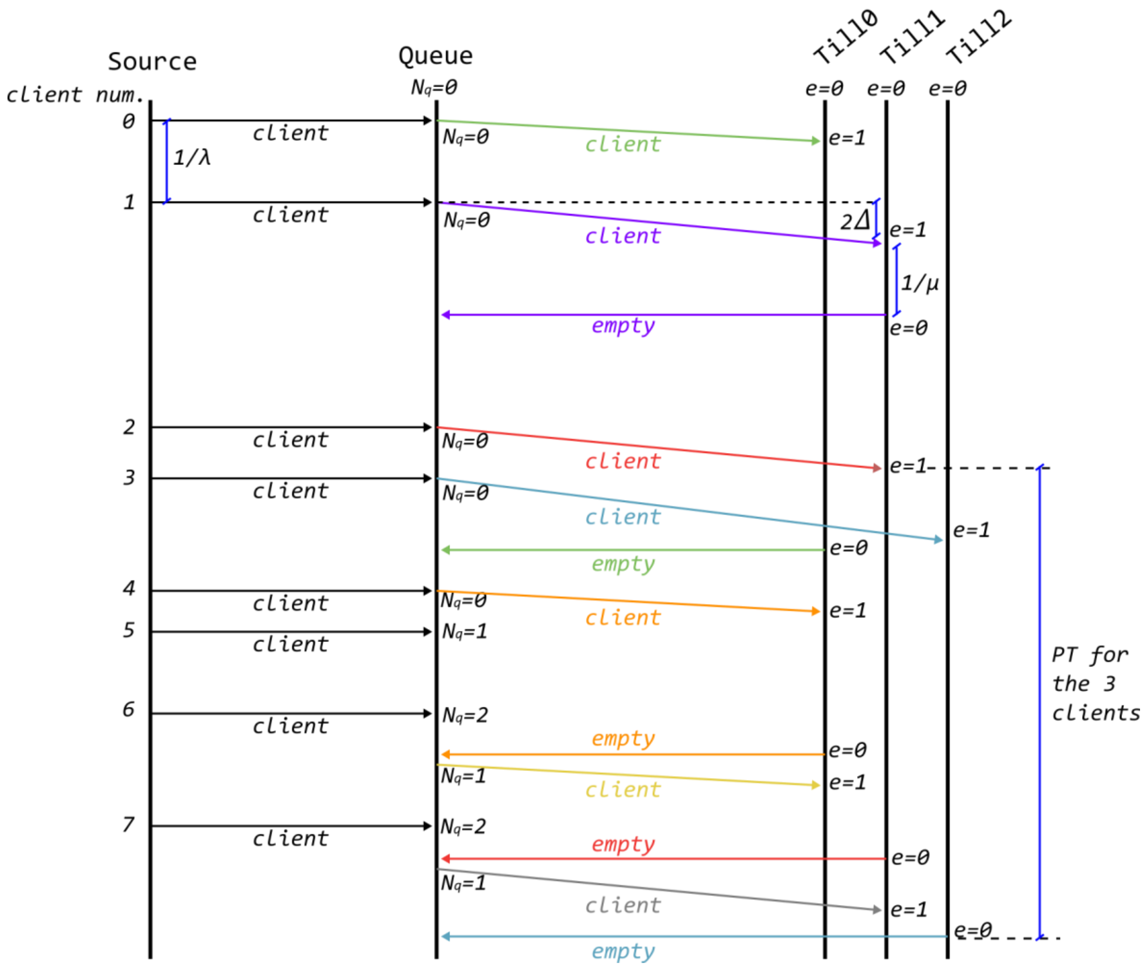

Figure 9 – Diagram of checkout policy A messages

On the messages exchanged it is possible to see the implemented Δ delay, used to simulate the time to reach the till. Also, it is noticeable the impact of the usage of multiple tills and the division algorithm, used to assign the clients. The result in Scenario A is that the overall Average Processing Time gets divided by the number of tills in the system.

The last module to be discussed is the Till. In this implementation the till only works as a sink, receiving the messages (clients/jobs) form the queue. The module then uses a RV to produce a response to the queue, informing that it has finished processing, and it is now empty. Figure 10 illustrates the histogram of processing times, considering and Exponential RV with mean value of 120s. Also, the Processing Time will also be tested as a Lognormal RV, Figure 11 contains the histogram for this distribution, for a mean value of 240s, with a spread of 0,5.

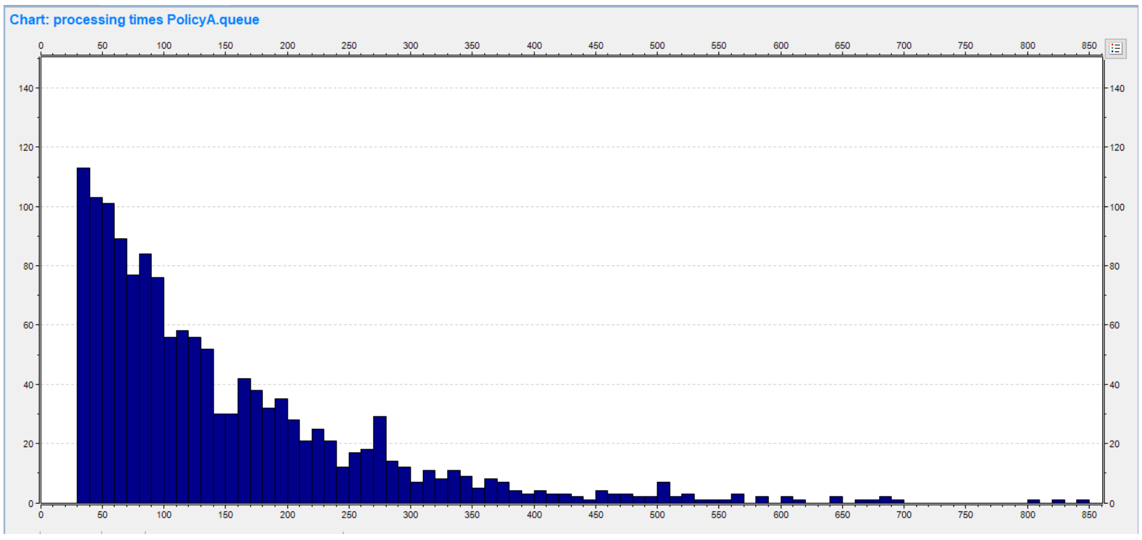

Figure 10 – Histogram of Processing Time as an Exponential RV

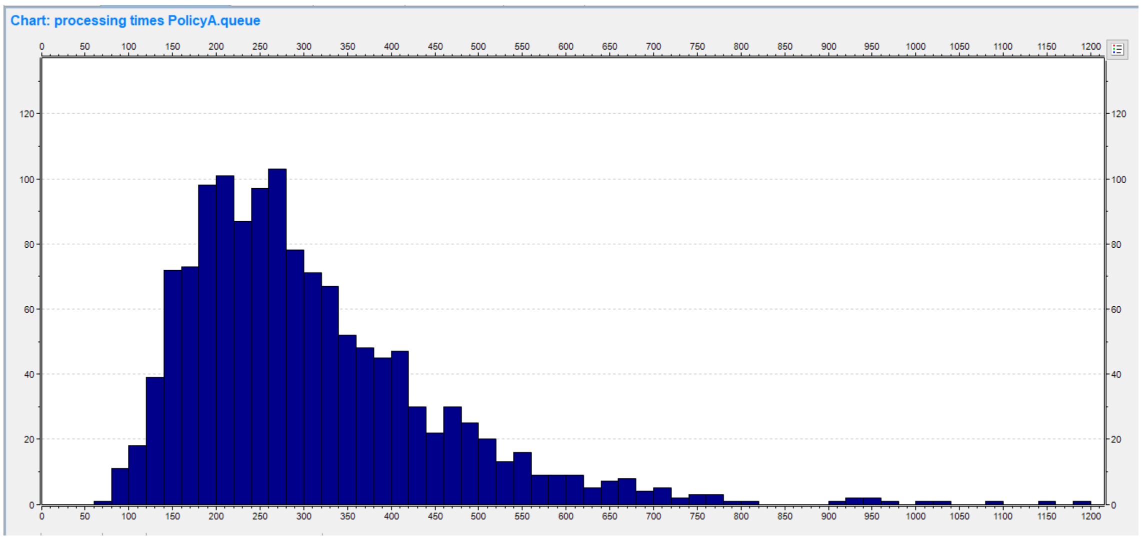

Figure 11 – Histogram of Processing Time as a Lognormal RV

The general view of the implementation of the checkout policy A is illustrated in Figure 12, generated by the simulation tool of OMNet++:

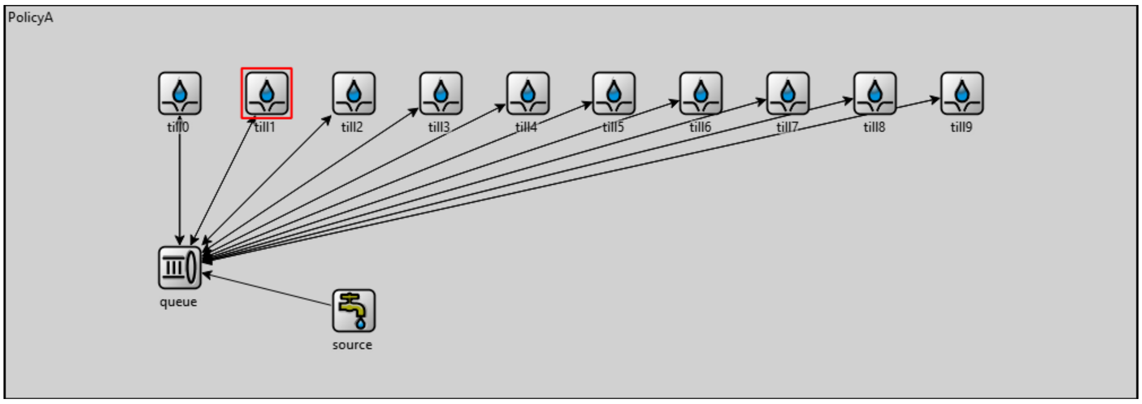

Figure 12 – OMNet++ Simulation block diagram for scenario A

3.2 Scenario B Development

For the implementation of the scenario B, a similar approach was used to develop the modules. However, some changes were necessary, considering now that the decision algorithm must be used in the Source module. The queues and tills then were simplified, as no selection need to be done in those modules for this scenario.

For the Source module the generation still follows the Exponential RV distribution. In this scenario, the module uses a search algorithm to find the queues with least clients on it. Also, the search must consider if the till is processing a client or not (as multiple tills may have a queue of size zero, but some of them may have clients still being processed), prioritizing sending the client to the queue with an empty till. In figure 13, the states that are handled by the Source in the checkout policy B are illustrated.

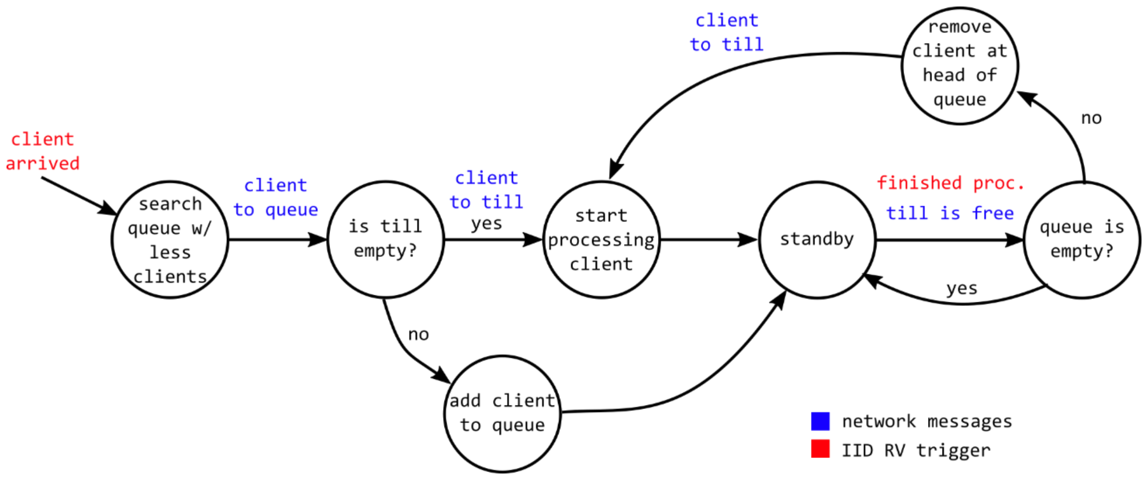

Figure 13 – Diagram of checkout policy B states handled by the Source module

The search algorithm used in the source must then keep track of the size of the queue and the state of the till in each subsystem. For this reason, there are two messages to be handled when a client is processed, one from the till module to the queue, informing that it is now empty, and a second message from the queue to the source, informing the update in the subsystem. The updates are then handled by the source module, and the status of all queues are controlled and tracked by it, as well as most of the data collection.

The processing done in the queues is now only to forward the clients as the till is empty, and to increase the client counter when the till is, informing the source of all updates in the subsystem (updates refer to changes in the queue, and when the till becomes empty).

In Figure 14, an example of the handling of clients and messages by the system in policy B are shown. A remarkable difference is noticed in the behavior of the steady state of this implementation, which differs the overall processing behavior from the previous scenario. Since the clients are evenly divided between the multiple queues, the Average Interarrival of each individual queue is divided by the number of queue-till subsystems used.

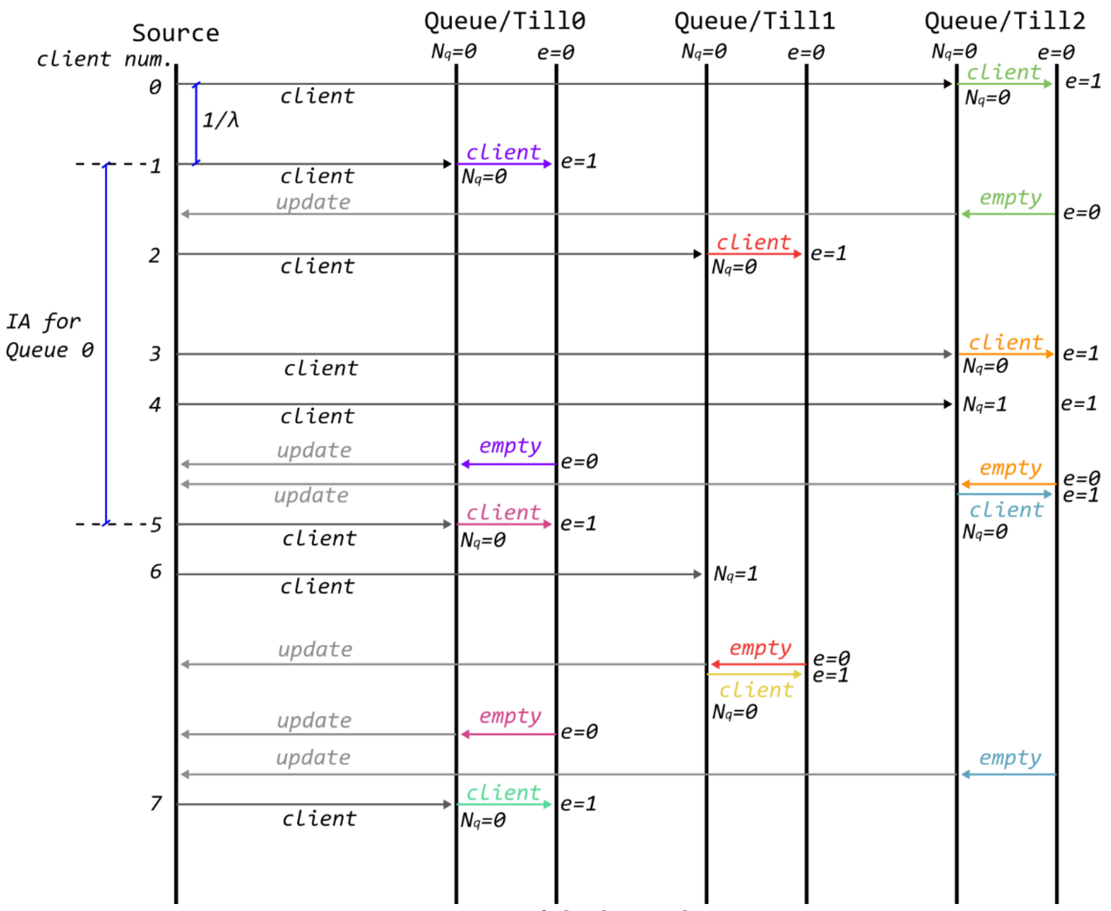

Figure 14 – Diagram of checkout policy B messages

The block diagram of the simulation environment for the checkout policy B, generated by OMNet++, is illustrated in Figure 15:

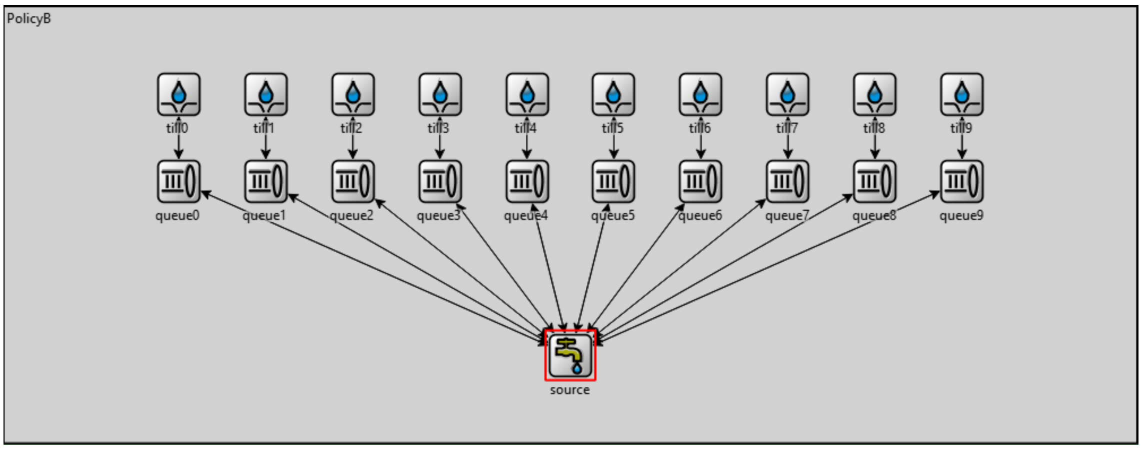

Figure 15 – OMNet++ Simulation block diagram for scenario B

4. Simulations and Results

To obtain meaningful results from the simulation environment, some specific metrics were defined to evaluate the systems. They are the average Queue Waiting Time, the average Checkout Time (system response time) and the maximum queue size. These parameters then define the quality of the supermarket service. To set the system configuration, three main parameters will be modified, the Client mean interarrival time (IA), the Client mean demand time (or processing time, PT) and the number of opened tills. With this implementation, it is possible to execute the checkout policies simulations with different utilization and workload values. The following table shows some examples of the used values to define the system checkout configuration, that is the Mean value of the RV distribution generated by the simulator:

Table 1 – Simulation parameters reference

SYSTEM WORKLOAD	SERVICE DEMAND
LOW CLIENT ARRIVAL (60s)	LOW DEMAND (120s)
NORMAL CLIENT ARRIVAL (45s)	NORMAL DEMAND (240s)
HIGH CLIENT ARRIVAL (30s)	HIGH DEMAND (360s)

For the system capacity, the maximum number of available tills is set to 10 and can be changed before the execution of the simulation. The time values used were obtained by observing a real scenario, in a local supermarket. It was noticed that clients were arriving for checkout with an average of one minute between them. Following the client arrival observations, the average processing time was noticed to be about two to four minutes in a normal buy, therefore, the medium demand value was set to 240 seconds.

As away to check if any of the system customizable parameters were affecting the key performance indicators (KPIs) before the full tests, a 2^kr factorial analysis was performed, using the average queue time and maximum queue time as reference.

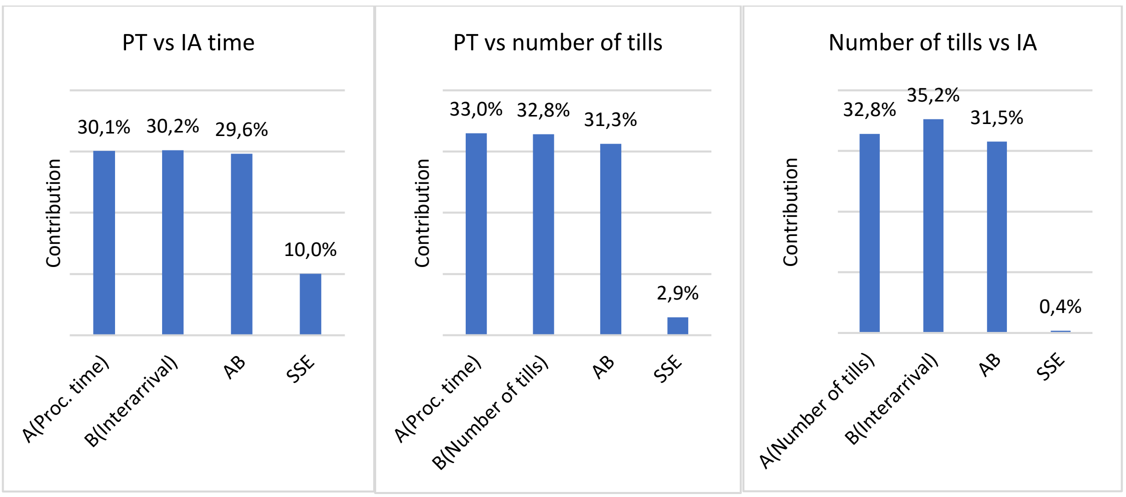

Figure 16 – 2kr factorial analysis for Average Queue time

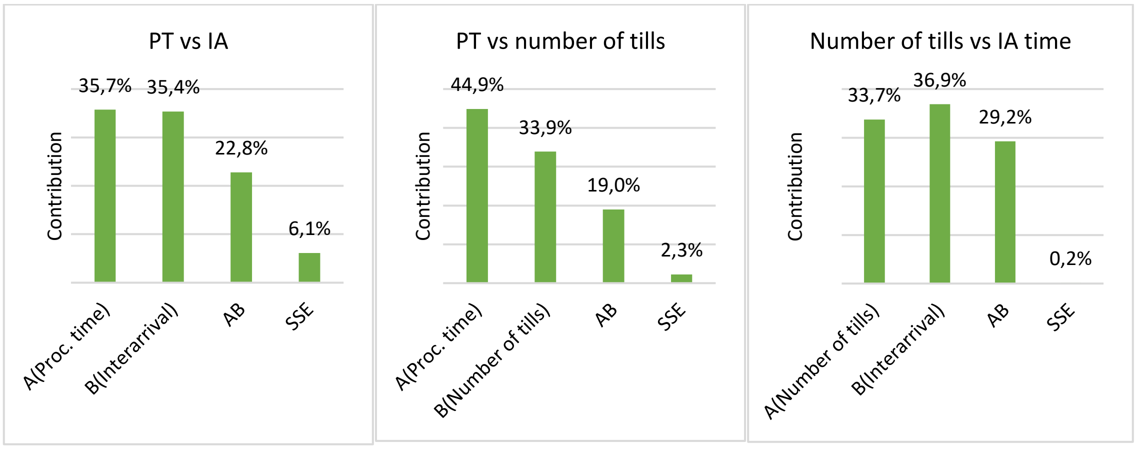

Figure 17 – 2kr factorial analysis for Maximum Queue time

From the results obtained in this analysis, no parameter was considered to have a significant contribution higher than the other. This was expected, as the relevant parameter to change the KPI values significantly is the system utilization. As both IA time and PT are generated randomly, and the utilization is calculated by the division of these variables, no prevalence for any of them was expected.

In addition to the parameter impact analysis, two different set of supermarkets were initially used to check if the system overall size had an impact on the KPIs. There were two configurations, one using a longer processing time (mean of 240s) and a total of 10 tills, considered as the big supermarket, and a second containing a total of 6 tills with a mean value of 120s for the client demand. Using similar client arrival times, it was possible to compare the systems under the same utilization rate. The results show that despite the larger system produce longer queues on average (due to the higher variance of the Exponential distribution), the confidence interval still makes both systems very close, considering their responses to the interarrival time changes.

The preferred system for the testing was then chosen as the 10 Till setup with a mean value for the client demand as 240s (without adding the constant PT value and the time to reach the till).

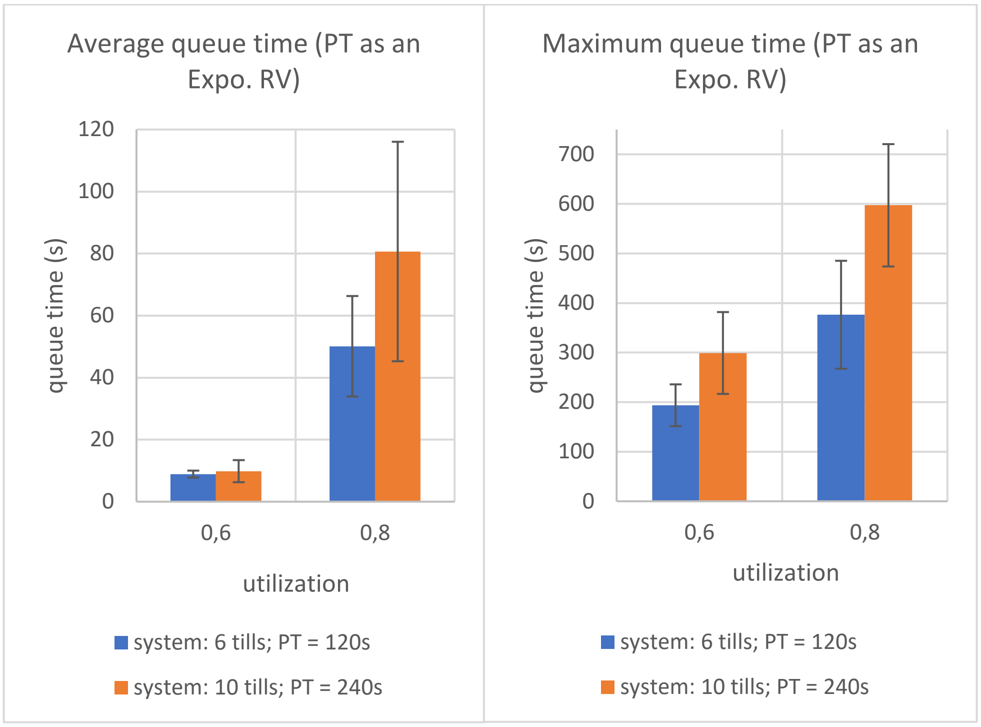

Figure 18 – System setup comparison

Furthermore, the main parameter to be explored to change the workload of the system will e the clients interarrival time, as it is expected that a supermarket checkout system to have a constant configuration (number of tills) and a similar service demands throughout a working day. The analysis will then focus on how the systems with different policies behave as the number of clients entering the system changes.

4.1 Checkout Policy A Testing

The first section of testing considered the evaluation of the system workload in the policy A setup. To produce a reasonable amount of data and, therefore, meaningful results, tests were made using a total simulation time of 12 hours for each run (reproducing an average service day).

4.1.1 Policy A tests with Clients Demand as Exponential RV

For the Checkout time test, and the model evaluation, the system will be explored with the variation of IA time, executing the supermarket simulation with different workloads. In this scenario the average Processing time was set with a mean value of 240s, using an Exponential RV. Also, as defined in the simulation creation, the constant processing time added is 30s, and the time to reach the first till is 2s (delta value). With all these parameters, the calculated Average Processing Time is then 281s. In this test, the system was at full capacity, using then all 10 available tills and three different random seed values. With these settings, a confidence interval of 90% were calculated for the available samples, producing the following curve:

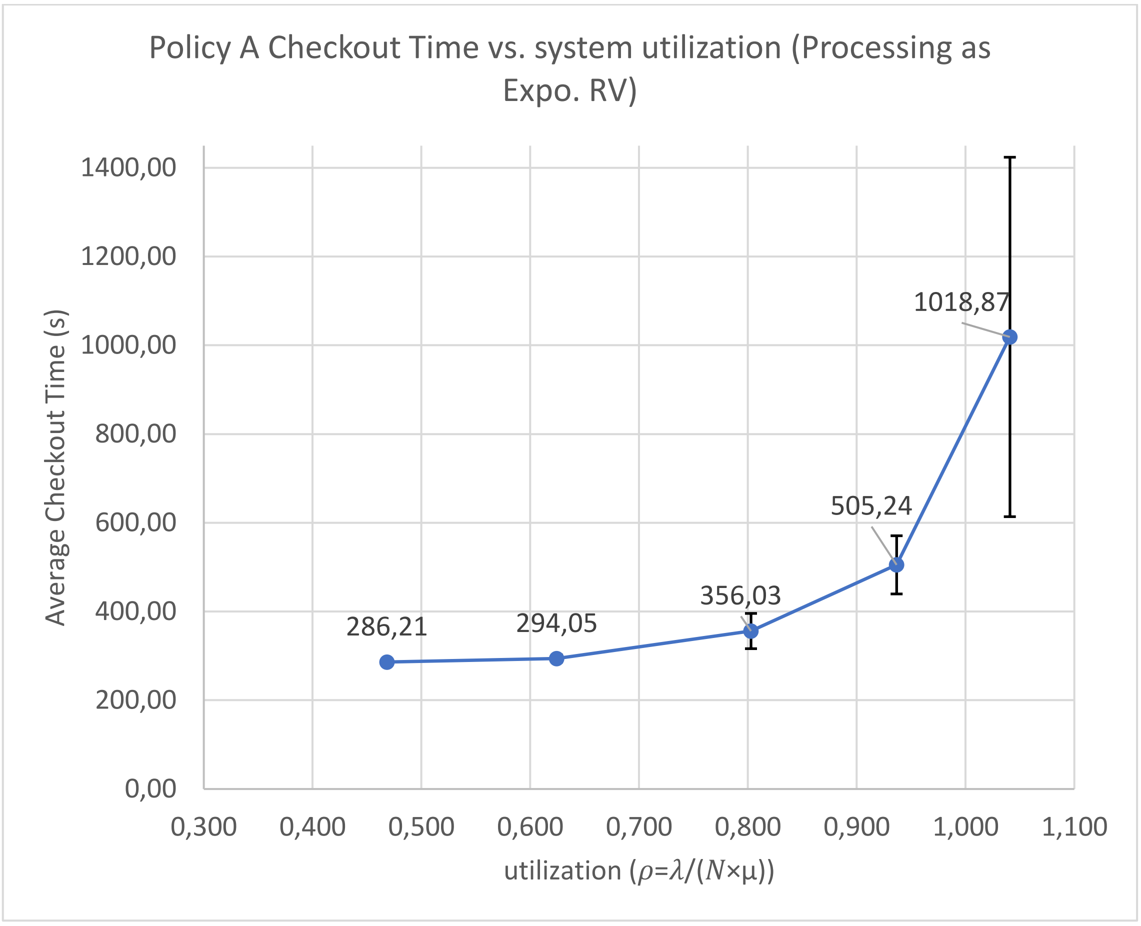

Figure 19 – Policy A checkout time plot – using an Expo. RV for PT

Noticeably, when the usage of the system is low, the checkout time approaches the calculated processing time (as only small queues or no queue are formed when the usage is low). As the utilization approaches 1, the system response becomes unstable, and the queue size grow indefinitely, producing larger response times.

To illustrate the queue behavior in the different workload utilization in the system, the following plots show the results of some of the tests. The data displayed shows how the queue size is every 10 minutes on the simulation, as well as the number of clients entering the system on that interval.

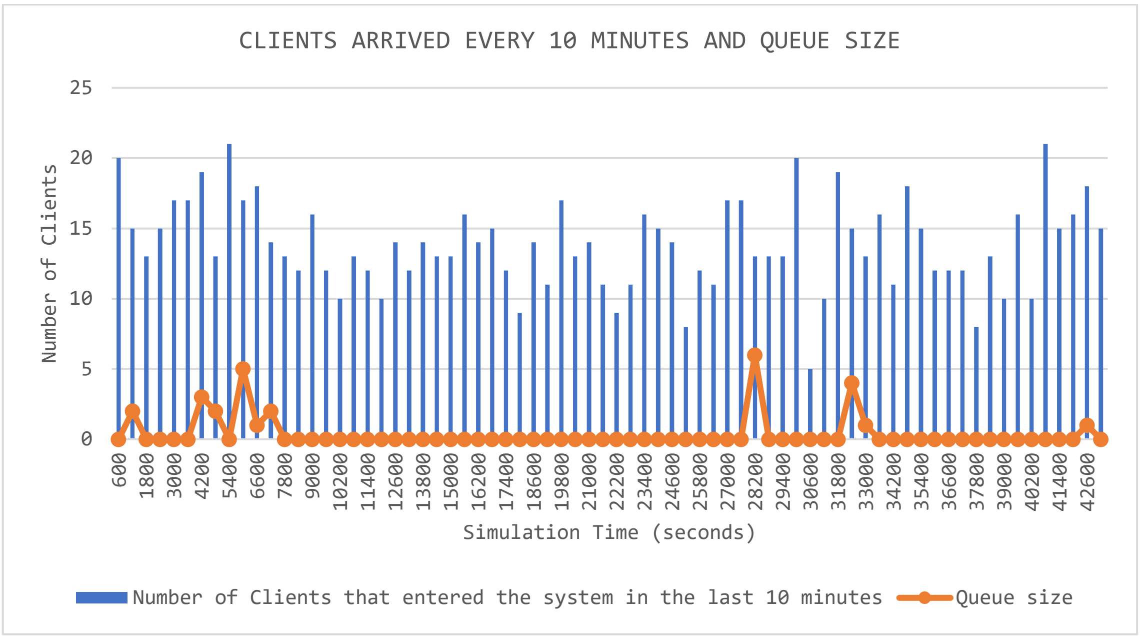

Figure 20 – System configuration: PT = 240s, IA= 45s, Tills = 10 – Utilization = 0,624

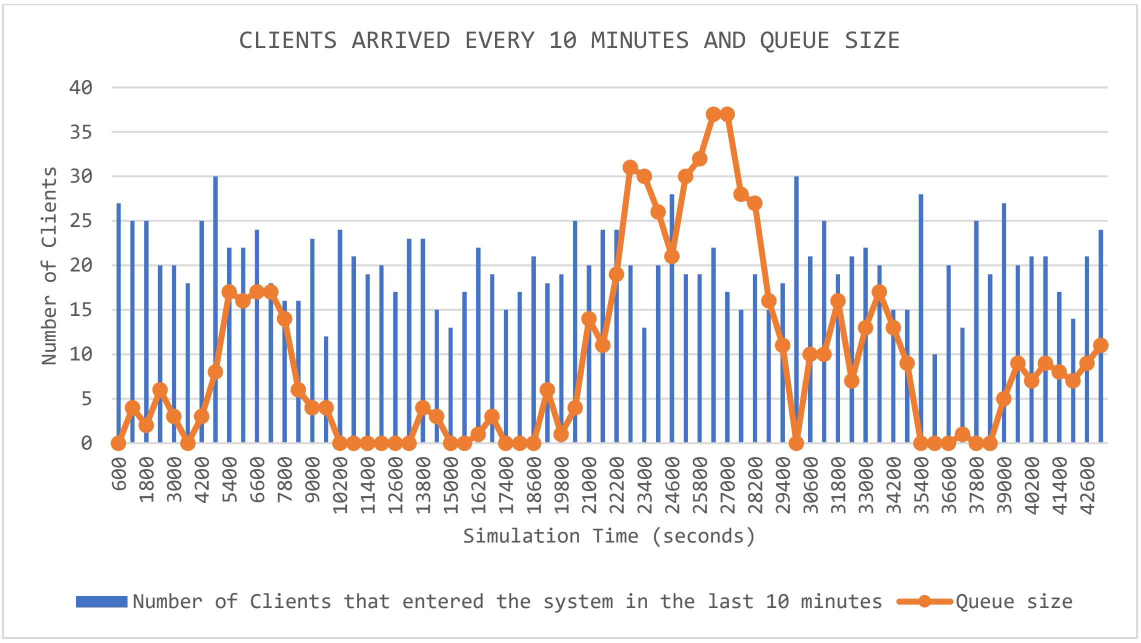

Figure 21 – System configuration: PT = 240s, IA= 30s, Tills = 10 – Utilization = 0,937

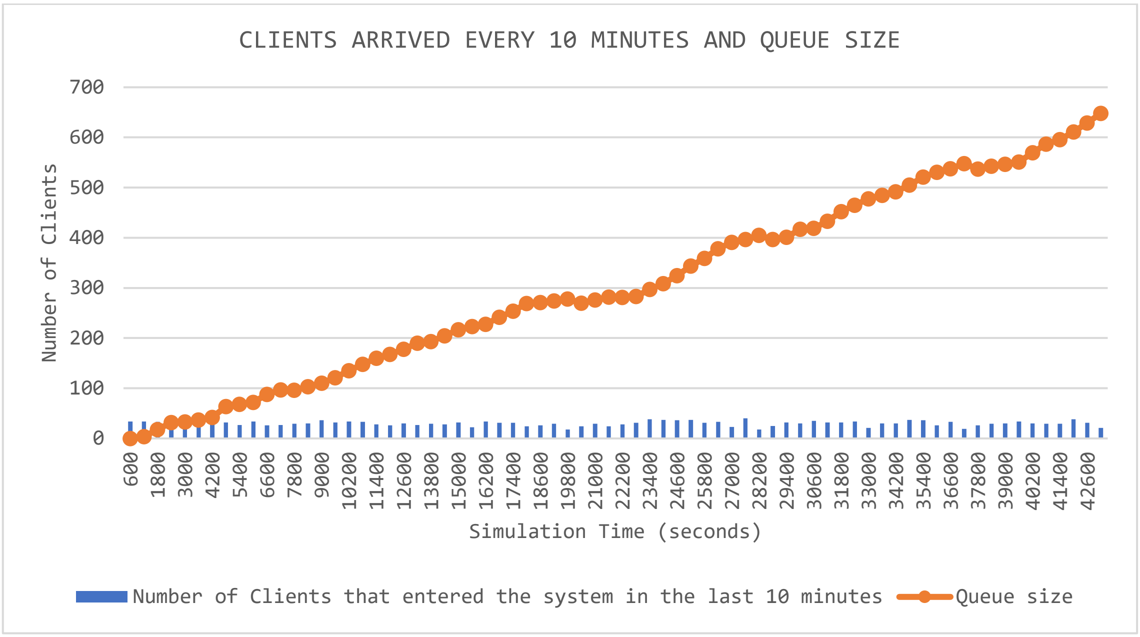

Figure 22 – System configuration: PT = 240s, IA= 20s, Tills = 10 – Utilization = 1,4

Complementing the general tests to evaluate the system response under different workloads, the information of how many clients were assigned and processed on each till was collected.

Following the optimization guidelines for this Policy implementation, the till assignment algorithm then prioritizes closer tills (less time to reach). Comparing the distribution of clients in Figure 23 with Figure 24, it is noticeable that when the utilization is low, the system will send more clients to the initial Tills, as often in this case more than one till will be free at the same time, demonstrating the correct implementation of this optimization.

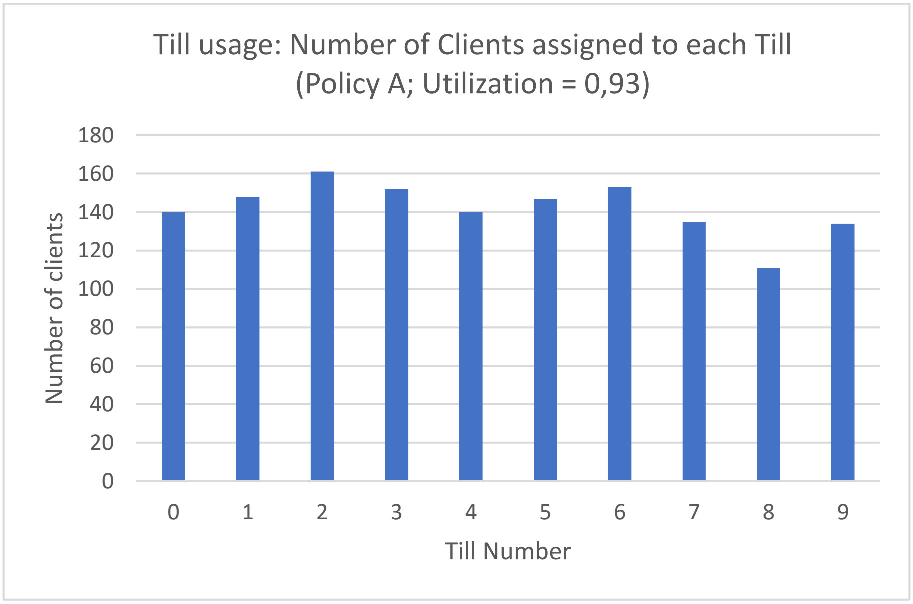

Figure 23 – Client distribution on each Till (PT = 240s, N tills = 10, utilization at 0,93)

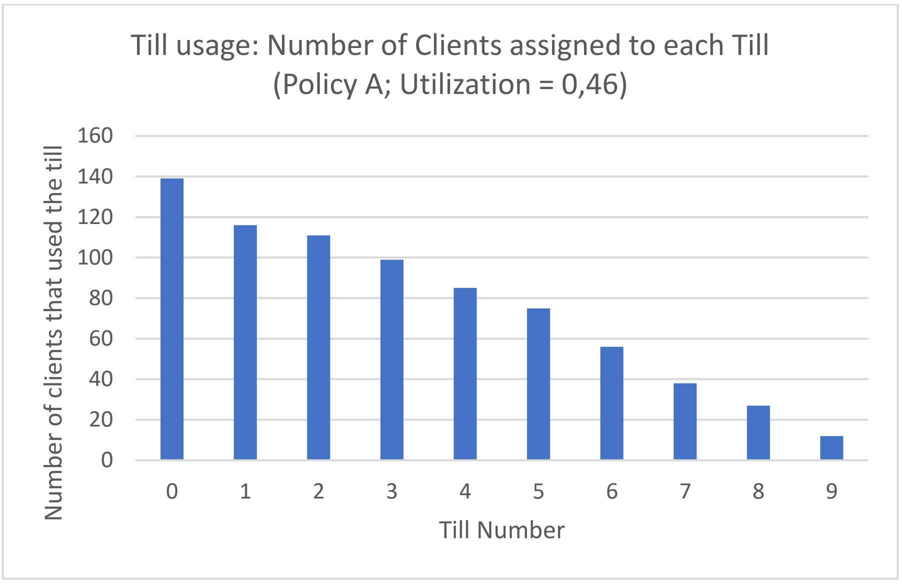

Figure 24 – Client distribution on each Till (PT = 240s, N tills = 10, with utilization at 0,46)

To complement and extend the tests performed on the Policy A scenario, a second mode of operation was designed in the simulator. This mode is defined as the Discouraged Arrivals mode, and it basically consists in changing the client's arrival rate based on the size of the queue. This behavior is observed in a real scenario, as often clients will avoid large queues, and wait for it to become smaller.

For this project, the parameters used for the discourage rate was to increase the client IA time in 10% for every multiple of 5 clients in the queue. That means that once the queue size reaches 5, the IA time mean value will be multiplied by 1,1. When the queue size reaches 10 clients, the IA will be multiplied by 1,2 and so on up to a maximum of 1,5. It is expected than that the queue never grow indefinitely, as once it gets larger, clients will not join it as often. The CTMC of this system and the results obtained in the simulations are shown in the following figures:

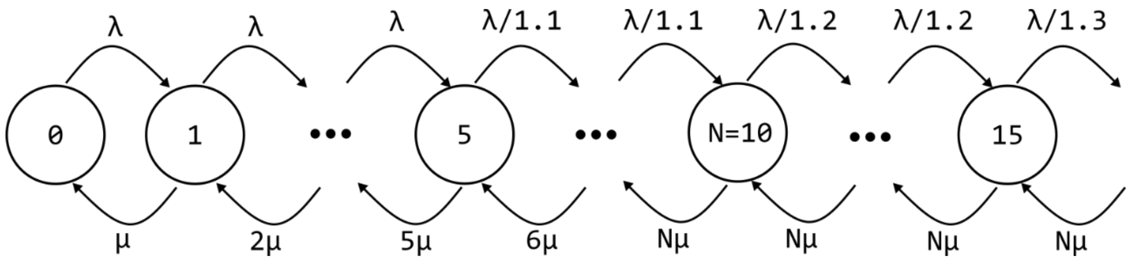

Figure 25 – CTMC of Policy A checkout using the discouraged queue mode

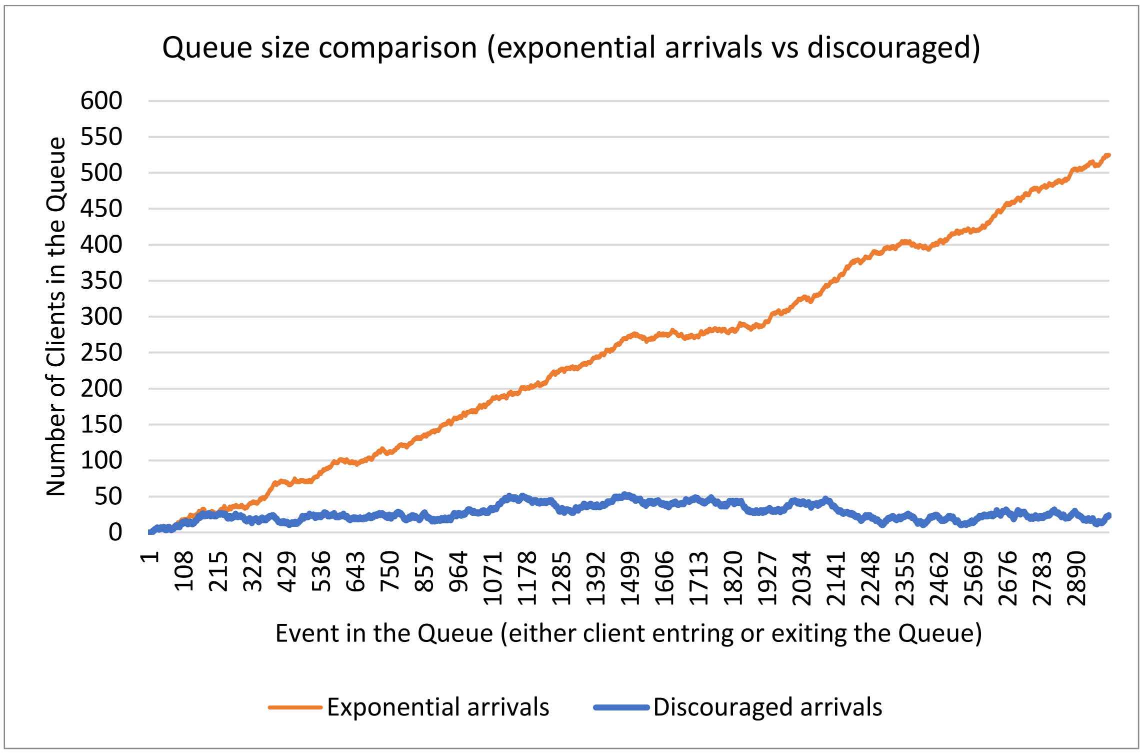

Figure 26 – Queue size comparison (sim. Parameters: PT = 240s, N tills = 10, initial IA value = 20s)

In Figure 26 it is noticeable that the discouraged mode did not allow the queue to grow indefinitely, even when the initial IA was set to 20s. These tests were performed using the same initial IA for both modes, and on the Exponential arrivals, the curve shows that the system was heavily saturated, yet, when using the discouraged mode, there was no indefinite queue grow.

Also, an extended test was performed on the discouraged mode, by running the simulation on 120 hours instead of 12. This example then shows how the queue oscillates during the test, as the IA is constantly altered, and the number of clients drastically drops as the queue grows, making it slowly decrease up to a point where the number of clients entering the system increases, increasing the queue again.

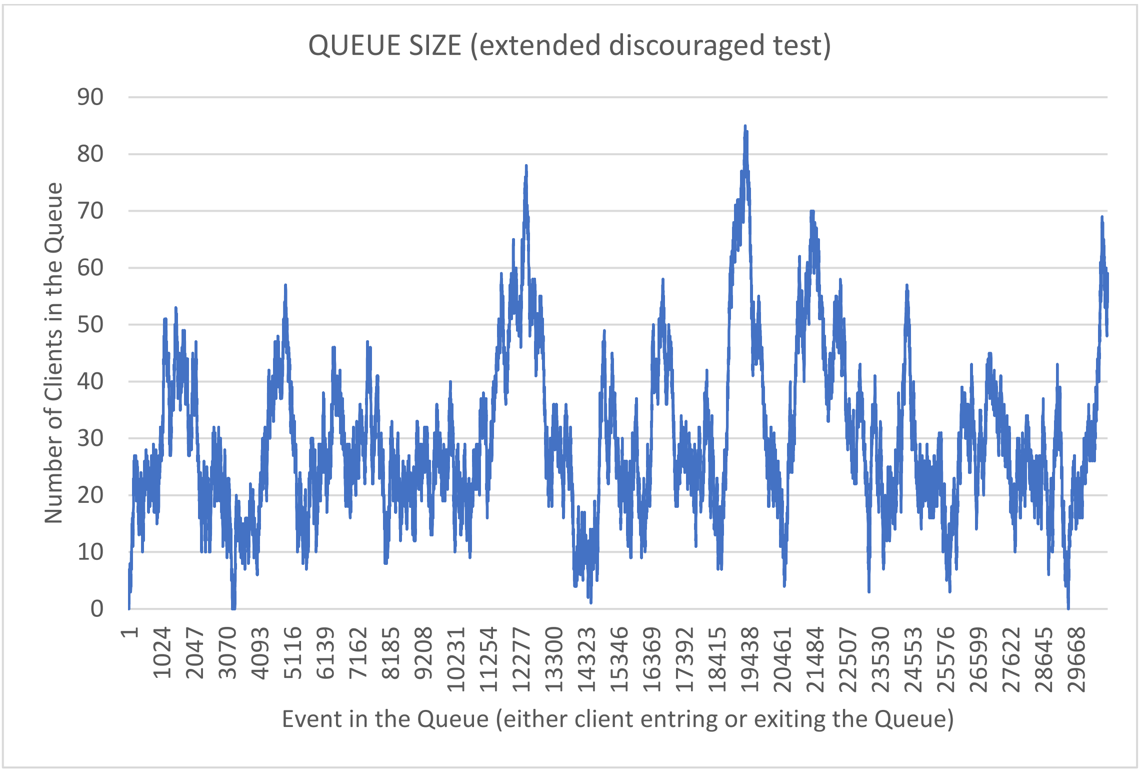

Figure 27 – Queue size on extended test using discouraged mode

Analyzing the behavior of the Policy A system simulation and all the results obtained from the exponential tests, it was possible to conclude that the implementation and design of this simulator was able to successfully represent the desired supermarket checkout system.

4.1.2 Policy A tests with Clients Demand as Lognormal RV and comparison

To observe the Policy A system using a different approach on how the clients are processed in the Till, the current exponential RV used to generate a value for the time that a client is processed was replaced by a Lognormal RV. To maintain the tests comparison consistent, the spread of this variable was adjusted in a way that the average PT was the same as the mean value of the Exponential RV used on the previous test. The bets parameter found was to set the spread value of this lognormal RV to 0,1, meaning that the deviation of the generated number was very low. The PT samples than are not distributed as in the Exponential RV but concentrated around the given mean value.

As an example, Figure 28 shows the result for a PT time generated with a mean value of 120s, adding the constant value of 30s. By using a low value for the spread it is noticeable that no sample generated had a value higher than 200s, and that most of the client PT values were concentrated at 150s. Next a dispersion comparing the randomly generated values, using both distribution is shown.

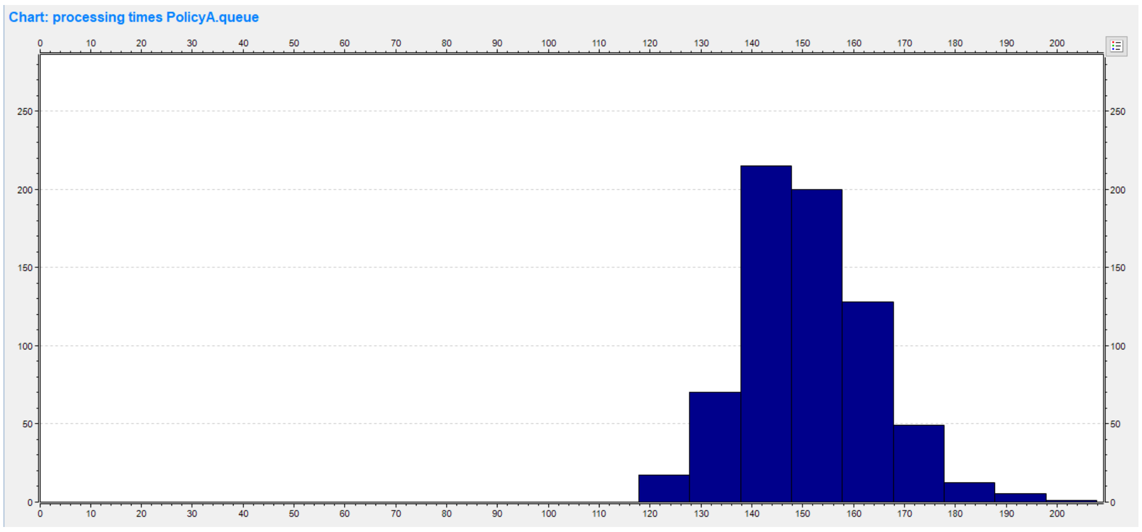

Figure 28 – Histogram of a sample of clients IA time generated using the Lognormal RV

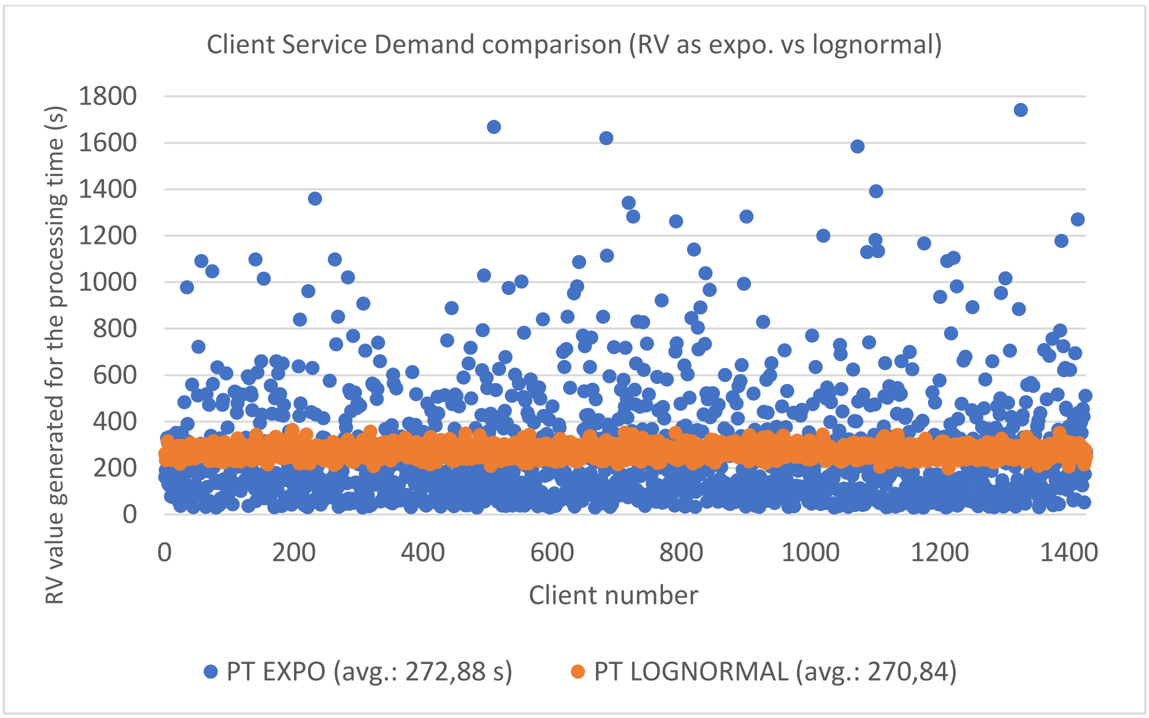

Figure 29 – Dispersion plot of the values generated by the IID RV used in the tests (mean = 240s)

Similar tests were performed using this RV, and by tuning the spread value, the utilization of the system was comparable with the tests using the exponential RV. Figure 30 shows the checkout time results for this scenario, plotted with the results from the exponential implementation.

Like the results observed in the previous configuration, the total time to go through the system when the utilization value is low tend to be close to the total PT (in this case a RV with mean value of 240s, plus the constant value of 30s and the average delay to reach the till, that is 11s, resulting a total PT of 281s).

However, due to the low spread of the IID RV, the curve has a delay to start growing, when compared with the tests using the exponential distribution. This means that once the workload increases, using a lognormal distribution for client arrivals, the system will take longer to start saturating the queue. From the plot, containing both curves of the system response time versus the utilization it is noticeable the gap between both implementations of the simulated checkout policy, indicating the mentioned behavior.

Once the utilization gets very close and over 100% of the system capacity, the checkout time then increases significantly, reaching similar values of the exponential distribution queuing behavior.

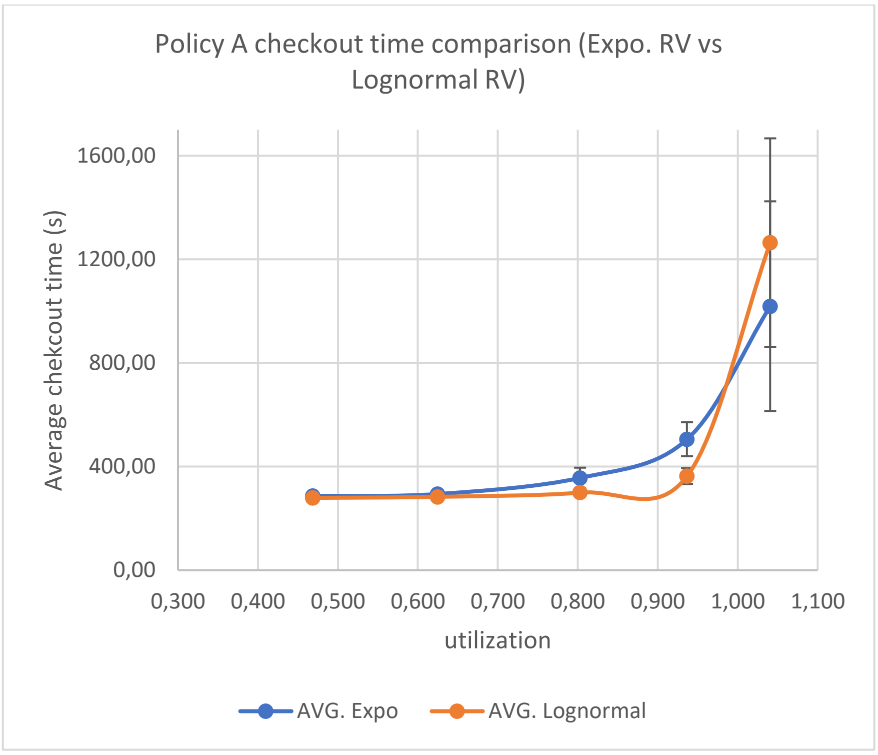

Figure 30 – Policy A checkout time plot – using a Lognormal RV for PT

Aside from the overall system modeling, the next analysis will focus on the evaluation of the KPIs defined for the performance of the systems.

When analyzing the average queue time results obtained in the simulations, it is possible to notice an exponential growth with the increase in the system utilization. As more clients enter the system, longer queues are formed, and so the time on average that a client must wait to be processed increases. With the defined parameter for this simulation, clients had to wait for about 4 minutes in the queue (using and exponential RV as the PT) when the utilization was at 93%.

In addition to the overall values, it is noticed the same gap between the tests using the lognormal distribution of PT. As expected, the exponential PT generated a higher waiting time for the clients, as it is more likely that clients with high demands enter the system and hold up the till while the other clients are waiting.

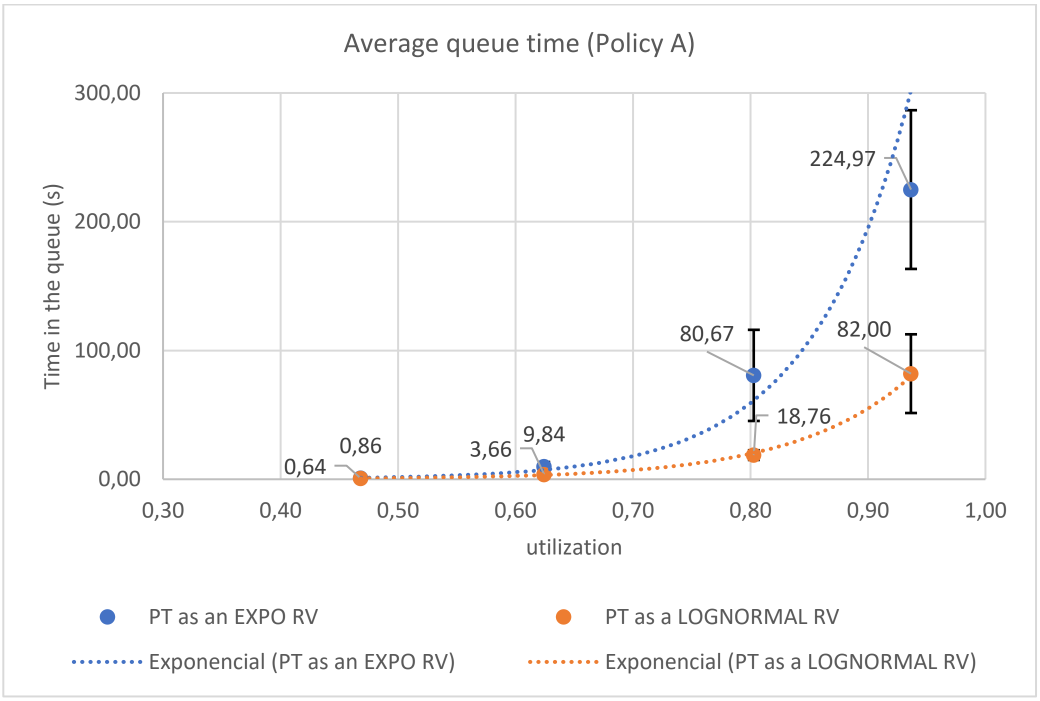

Figure 31 – Average queue time comparison

The last KPI parameter to be discussed when testing the Policy A implementation was the maximum queue time that a client had to go through. This parameter evaluation tries to identify possible scenarios where clients are taking to much time on the queue, even though on average the queue is fast.

Observing the results, the values for the maximum queue time are much higher than the average, specially in the case of the scenario using the exponential distribution, as its variance is higher than the lognormal distribution used.

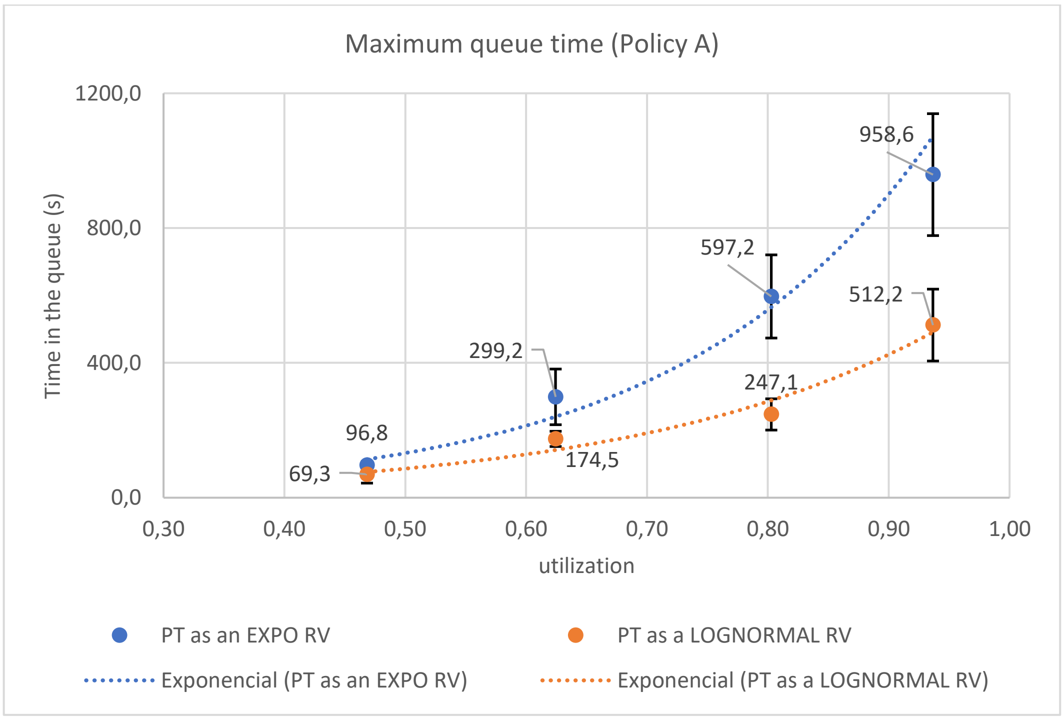

Figure 32 – Maximum queue time comparison

4.2 Checkout Policy B Testing

With the results obtained on the Policy A implementation, a similar approach was done for the tests of Policy B. However, the data collection mode had to be different, as from the simulator, it was not possible to collect the exact same type of data, as on the previous scenario, only one queue was used. In scenario B, where each till has its own queue, the values for the entry time, checkout time, queue time and maximum queue time had to be done independently for each queue-till subsystem. Then all the data was combined and analyzed, providing the results shown in the next sections.

4.2.1 Policy B tests with Clients Demand as Exponential RV

The first test presented is using an exponential RV for the PT of each till. Following the test order used to validate the scenario A implementation, the initial evaluation considered the system response under varying clients arrival rates. The values were consistent with the expected results, as the time values observed were like the previous tests and averaging a real checkout procedure. Figure 33 illustrates the checkout results for scenario B tests using an exponential RV for the generation of clients demands:

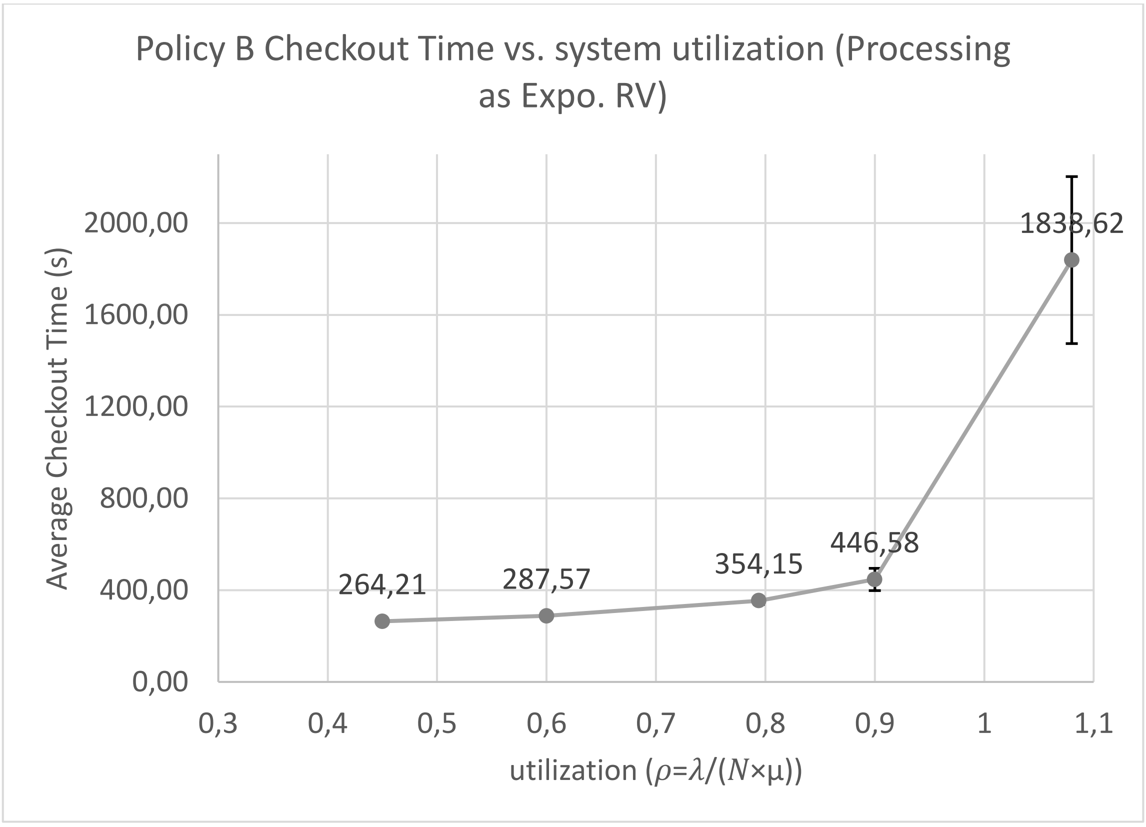

Figure 33 – Policy B checkout time plot – using a n Exponential RV for PT

As the previous scenario, the checkout behavior was as expected for the queueing system designed. As the utilization increased, the average checkout time also increased, and as the utilization crossed maximum system capacity, the queue time spikes and there is an indefinitely grow in the queue time.

Also, for this scenario, the parameters had to be tuned to be similar utilization rates as from the obtained on the previous scenario test. This then allowed a direct comparison between the two simulated environments.

Since there is no delay to reach a queue on checkout Policy B, and the clients IA were kept, the overall utilization values were lower when compared with the Policy A system configuration. Figure 34 illustrates the results obtained from both simulations, and demonstrating the similarity in the systems response when the workload increases:

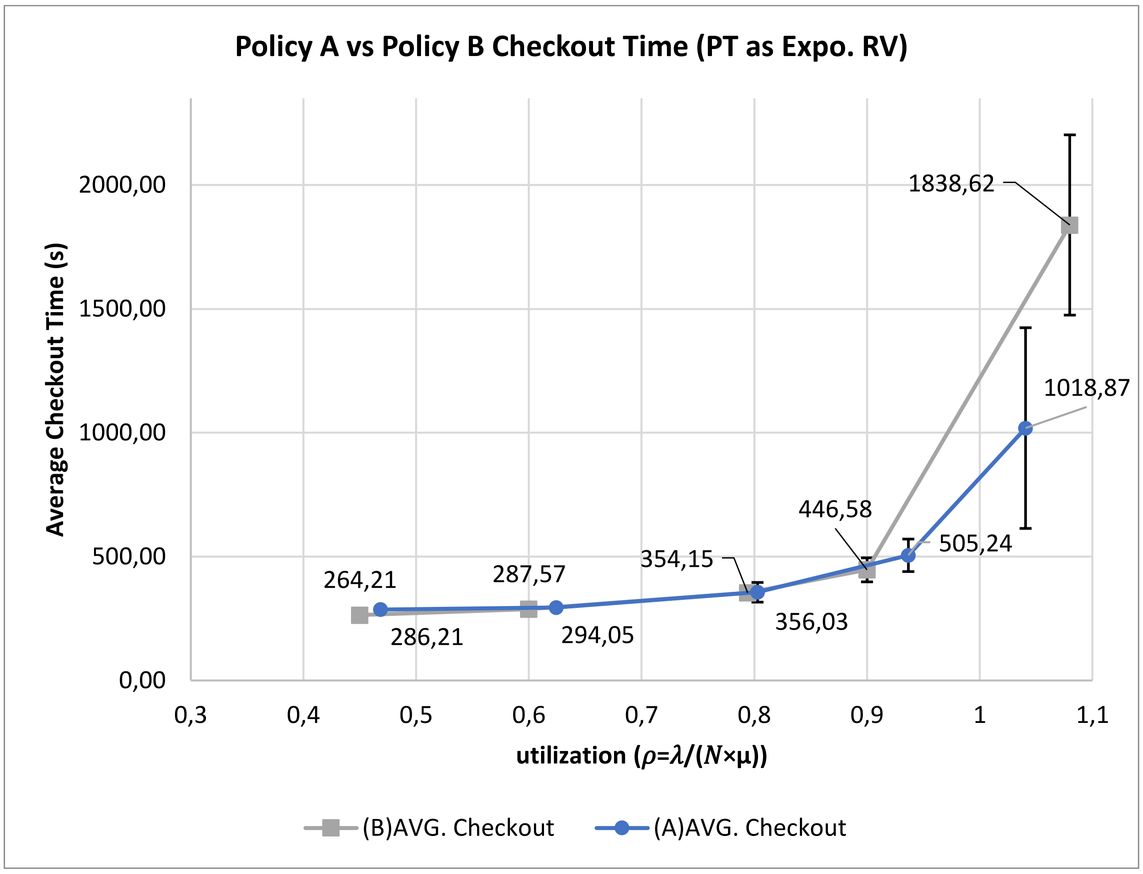

Figure 34 – Checkout time plot comparison: scenario A vs B (using PT as an Expo. RV)

Thought the response for both systems be very similar under this variation of clients IA time, the analysis of other parameters shows the differences between those implementations of the checkout policy.

By using an individual queue for each till in scenario B, it was expected that they would not grow with the same size as the single queue in Policy A. This is also an effect of the distribution algorithm used to assign clients to the smallest queues in the system. The following figures illustrates the queue behavior on one specific till of the simulated system (Policy B):

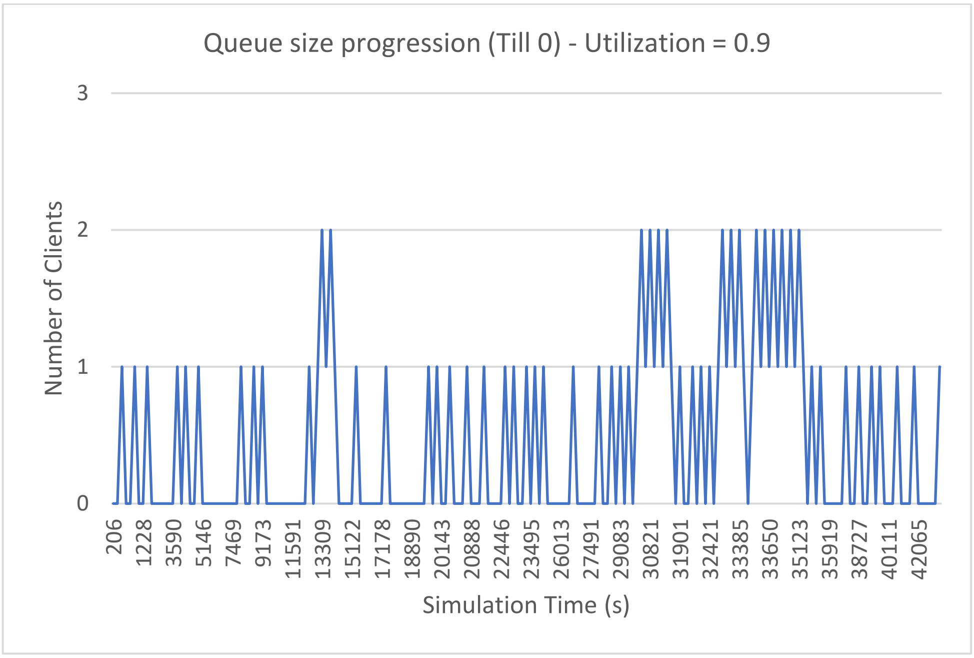

Figure 35 – Policy B queue size on Till 0 with system utilization at 90%

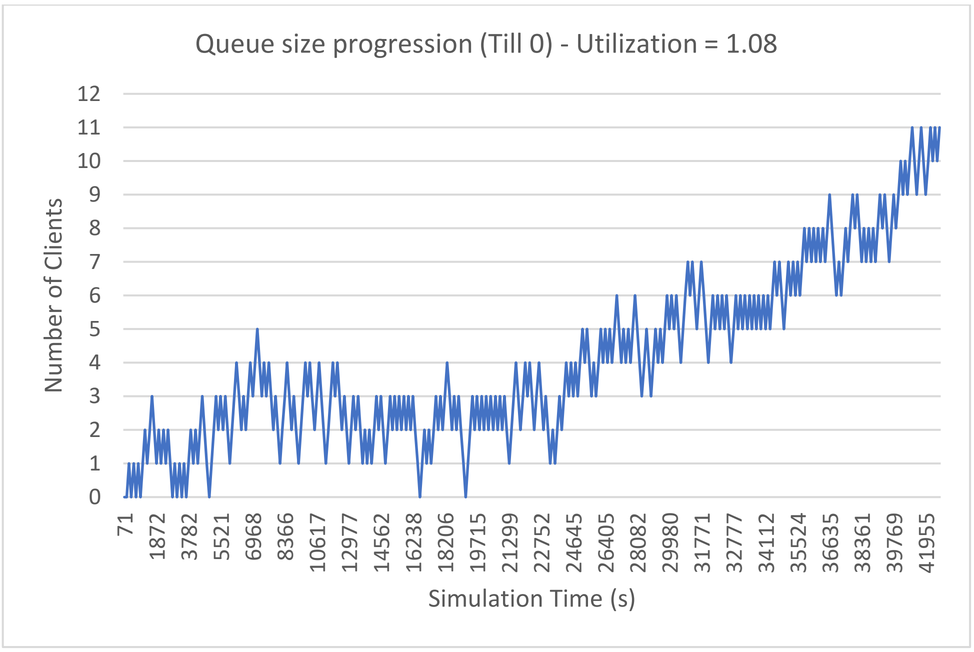

Figure 36 – Policy B queue size on Till 0 with system utilization at 108%

Following the system design testing used to evaluate scenario A, the distribution of clients among available queue/till subsystems was also accounted in this scenario. However, since the distribution algorithm ensures to distribute clients evenly (considering the individual queue sizes), it was expected that even in a low utilization rate, the clients would been equally divided across all tills.

Figures 37 and 38 show the results from the tests, demonstrating this behavior of the distribution algorithm:

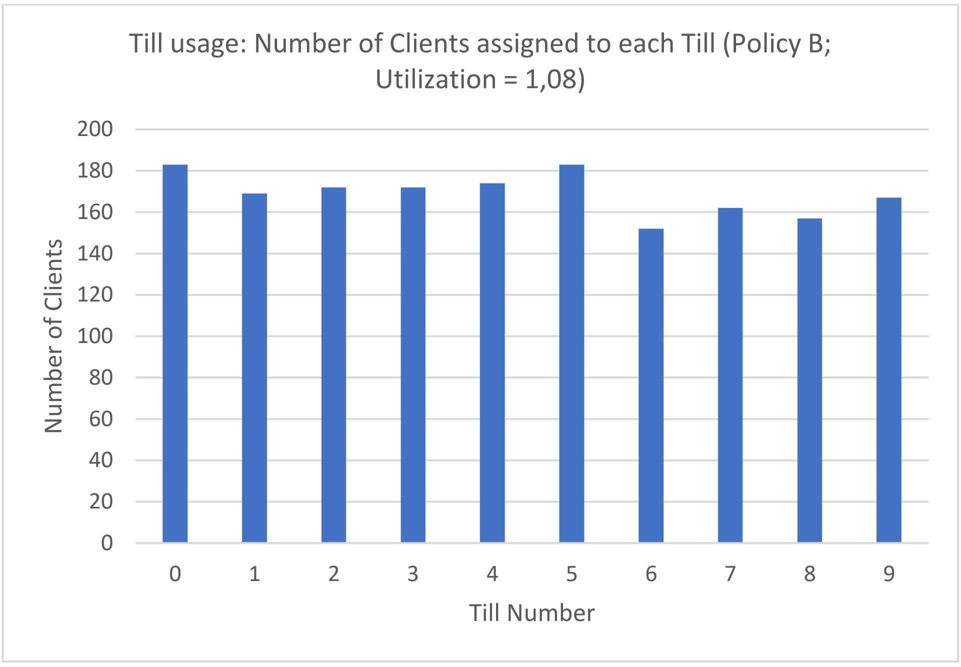

Figure 37 – Policy B client distribution on

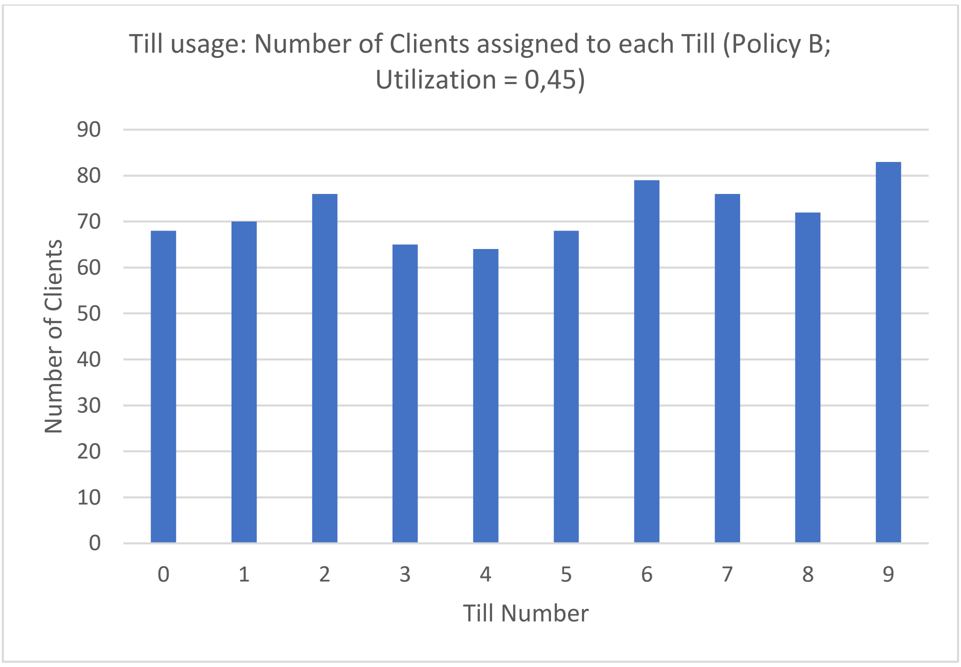

Figure 38 – Policy B checkout time plot – using a Lognormal RV for PT

From the results, we can conclude that even when the system utilization is low, the clients are, on average, evenly divided on the available tills, improving the overall efficiency rate of those tills. This could also be a considerable parameter to be a reference for the system quality evaluation, as when the clients are homogeneously distributed, the throughput of all the tills would remain the same.

With this section we then conclude that the implementation of scenario B was successful. Further details on the KPIs will be discussed on the next sections.

4.2.2 Policy B tests with Clients Demand as Lognormal RV and comparison

Following the guidelines for the project specification, scenario B was also tested using a Lognormal IID RV as the generator of client demand, that is the processing time PT. As with scenario A, the variance of this distribution was set to be low, and so the results, when comparing both types of PT generations, the results should resemble as the ones saw in the previous tests.

However, what was observed was a smaller gap, when comparing with the tests in scenario A. The following plot illustrates the checkout times observed when comparing the different types of random generated PT values:

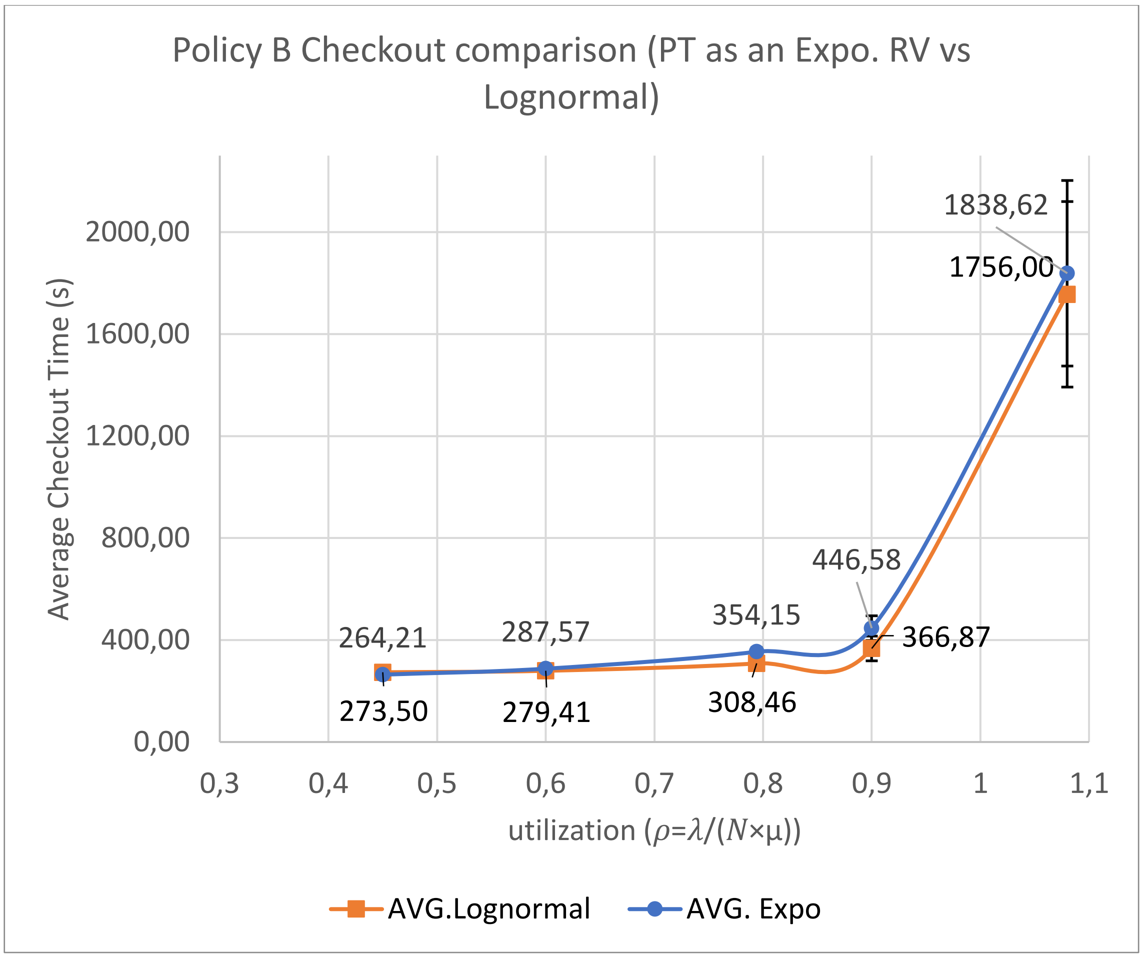

Figure 39 – Policy B checkout time comparison (PT as Expo. RV vs Lognormal)

After the model overall comparison, the KPIs were also evaluated. The two main parameters were again tested, and the results are shown in Figures 40 and 41. From the average queue results, it was possible to observe the variance impact between the two configurations using different types of RV. The exponential results then produced higher queue time on average, as it is more likely that a client enter the system with a large demand, thus stalling the queues.

Also, when analyzing the results of the maximum queue size a very important result was observed. The Policy B implementation of the system, when using an exponential variable to model the client demands, the maximum queue time were very high, when compared to every other scenario tested.

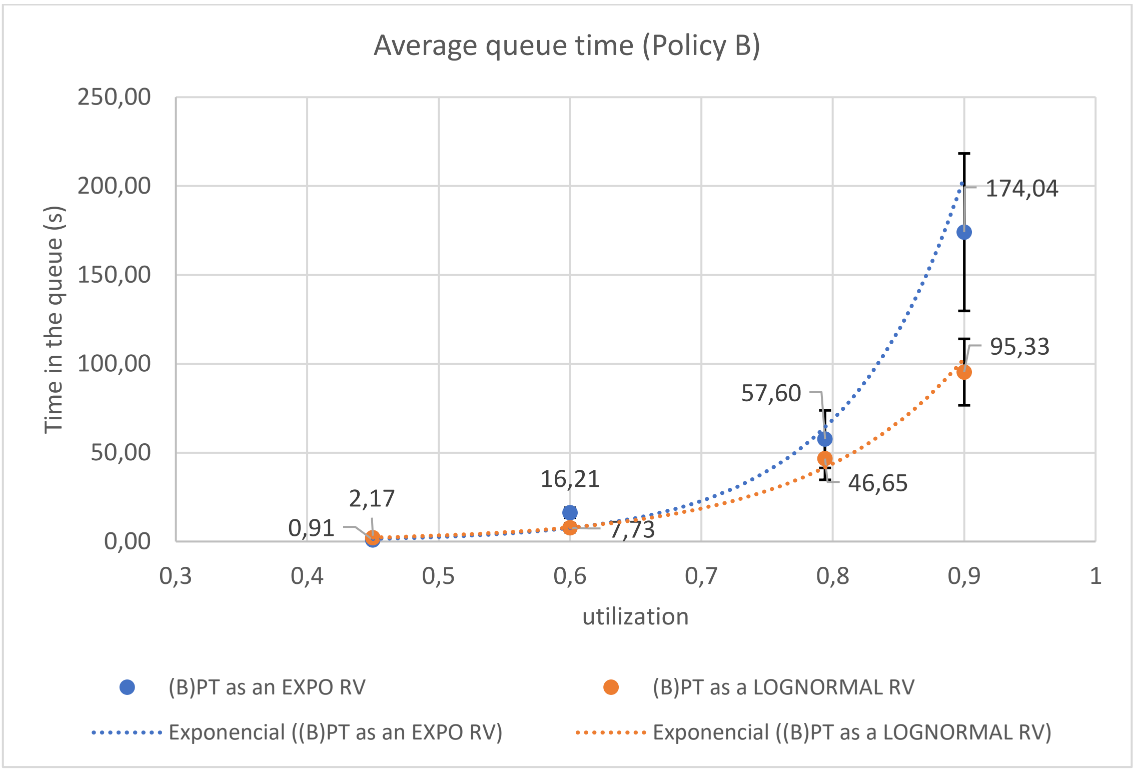

Figure 40 – Policy B average queue time comparison (PT as Expo. RV vs Lognormal)

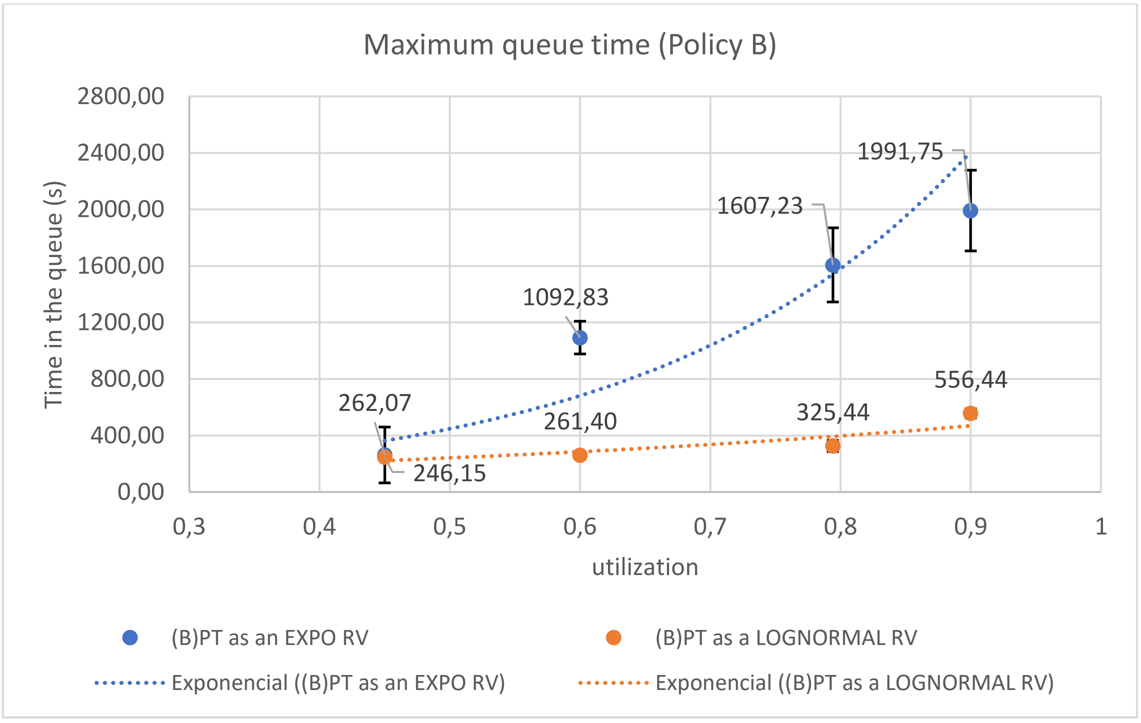

Figure 41 – Policy B maximum queue time comparison (PT as Expo. RV vs Lognormal)

This extreme behavior could be explained by the fact that, clients choose the queue based only on the current size, and thus, disregarding the clients demand that are on that till. The result of this implementation is that once a client enters a till, he is not allowed to leave the queue, and move to another one. Then when a relatively large queue is formed, a client, or even a sequence of clients, may have a high service demand. Then the clients already assigned to that till will experience a high queue time, even though the results show that on average the queue time remained low (as an example, on utilization 0,8: the average queue time is about one minute, while the maximum time recorded was up to 26 minutes).

4.3 Policy A vs Policy B comparison and Fast queue mode

Initially, it was thought that no policy system would be slower or, on average, worse than the other. However, from the test results observed it was possible to conclude that Policy A can deliver better KPIs that Policy due to the high maximum queue time values observed on the simulations. To demonstrate these insights, a comparison between scenario A and scenario B, the following plots shows the obtained results with the simulations:

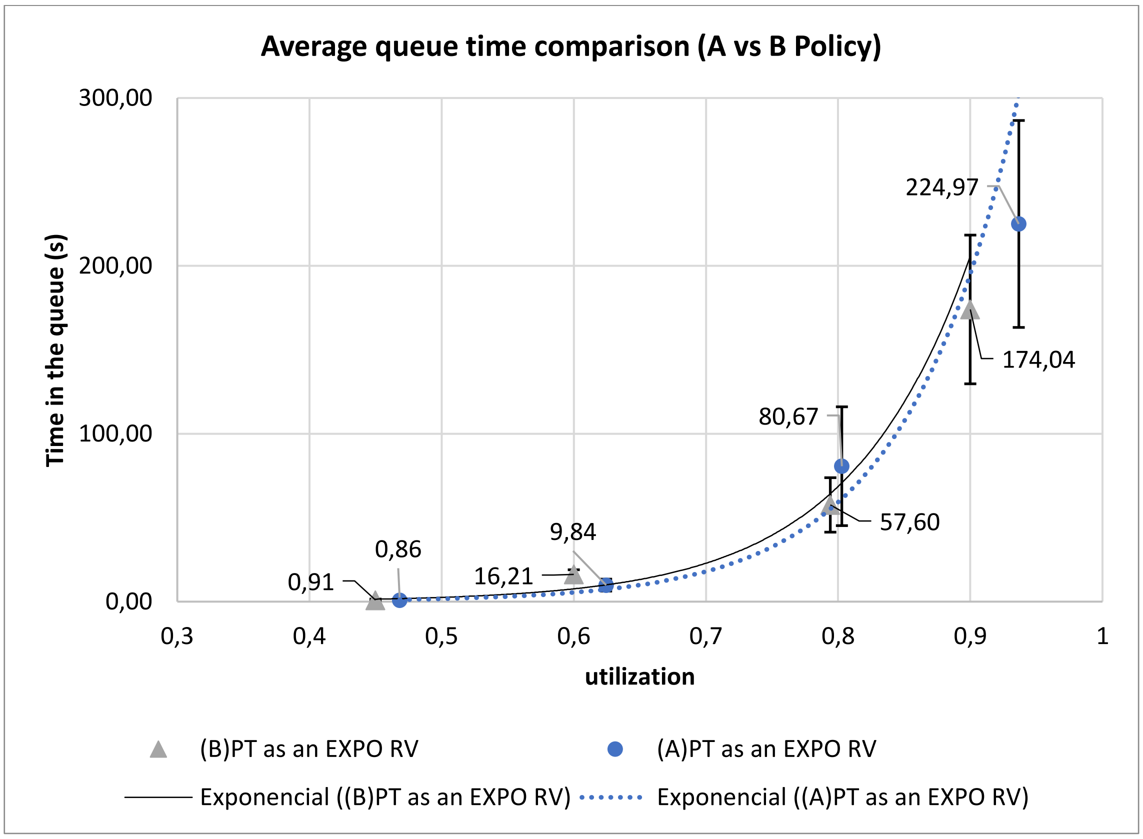

Figure 42 – Average queue time comparison (Policy A vs Policy B, with PT as Expo. RV)

As described, the results for the average queue time were similar between the different implementation of the supermarket checkout systems. However, when comparing the maximum queue time, there is a clear discrepancy between the scenarios.

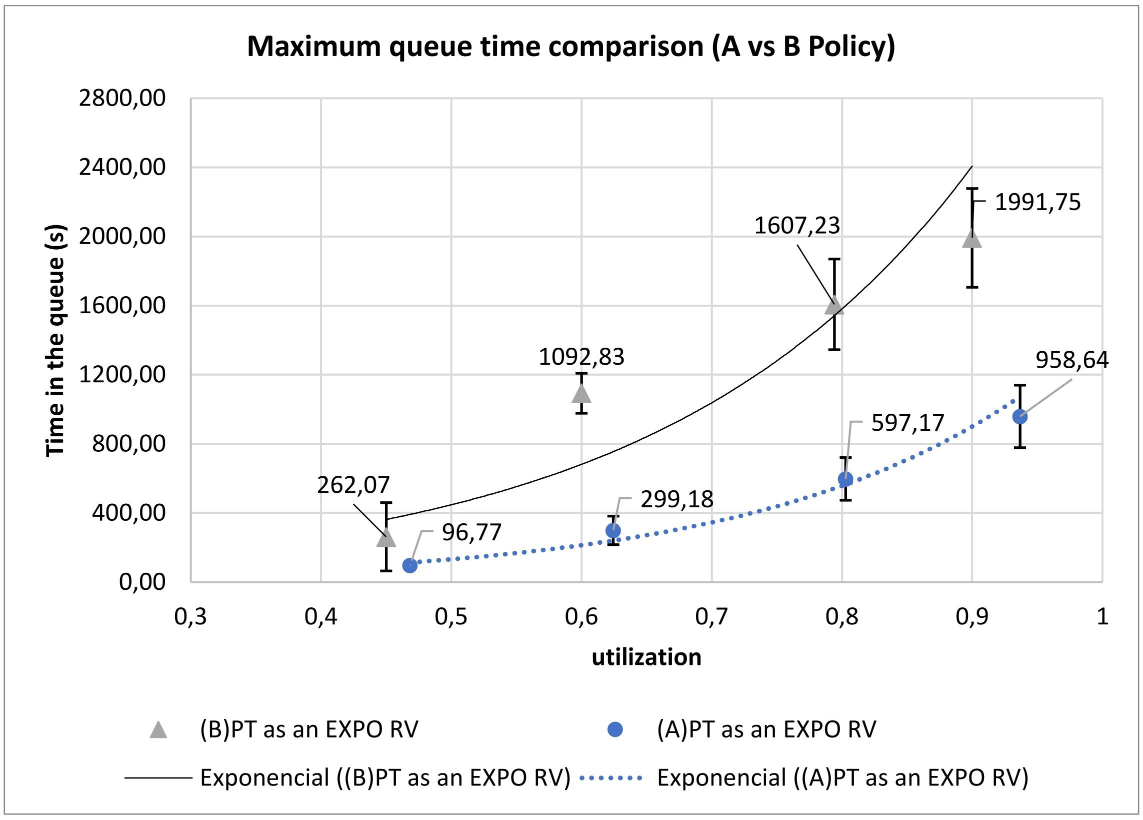

Figure 43 – Maximum queue time comparison (Policy A vs Policy B, with PT as Expo. RV)

With the discovery of this important discrepancy, it was proposed a solution to this issue on the policy B modeling. To compensate the high maximum time on queues, a system where clients are divided into priority queues was designed. This then allows a certain number of tills in the system to receive only clients with small demands, denominated as Fast queues. This new mode of operation then tries to reduce the chance of a client to be stalled in a normal queue, as the Fast queue can attend more clients, lowering the average queue size of the normal tills.

On the simulator, the configuration of the tills was then separated, and for this specific test, 3 of the 10 available tills were converted to Fast tills. The random generated PT for these tills was then set to a lower mean value (on the tests shown in Figure 44 and 45, the normal tills have a RV mean value of 240s, while the fast tills have a mean of 120s) and made in a way that when the generated RV was higher than a threshold (300s), it would be reduced artificially by 1/10 of its value. This changed the properties of the exponential RV, but maintained the demand of all clients in the Fast queues low.

With this design, the three fast tills would then be guarantied to not have high demands, and thus being often smaller, and preferred for the clients. The distribution of clients on one of the tests is shown in Figure 44. Noticeably the fast queues received more clients, as now the normal queues will likely queue up faster (the distribution algorithm remained the same).

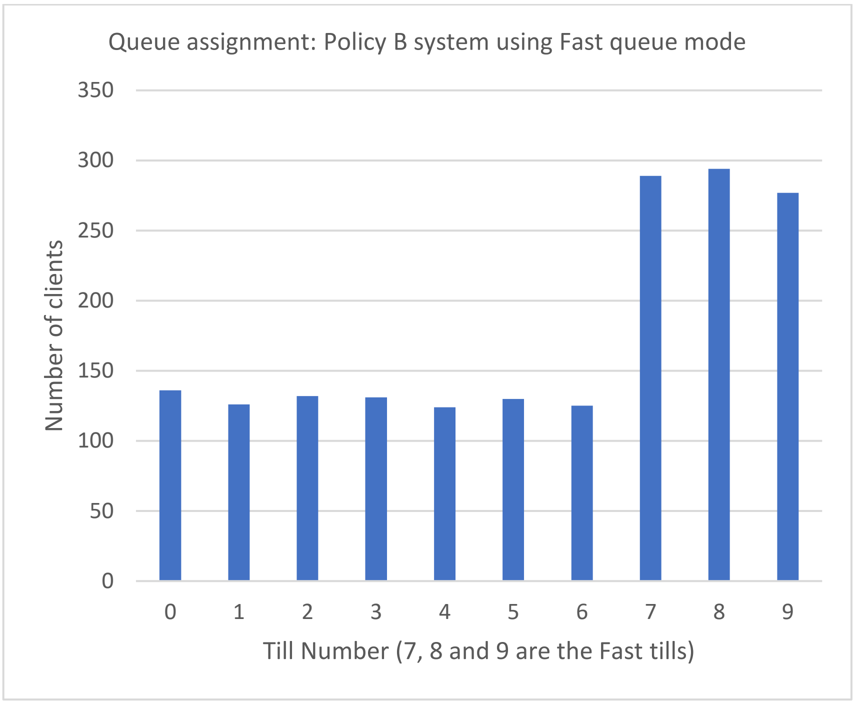

Figure 44 – Client distribution on the tills using the Fast queue mode (Policy B with PT as an expo. RV)

To test this implementation, the parameter if the system had to be tuned, as by reducing the PT for some queues, the utilization would naturally be lower, considering the same number of clients entering the system. Due to this factor, the IA was increased on this system to match the utilization of the previous tests. The results obtained with this solution then shows a significant reduction of the maximum queue time, as well as the average queue time (26,6% for the maximum queue time, and 38,4% for the average queue time) when compared with the system without the Fast queues.

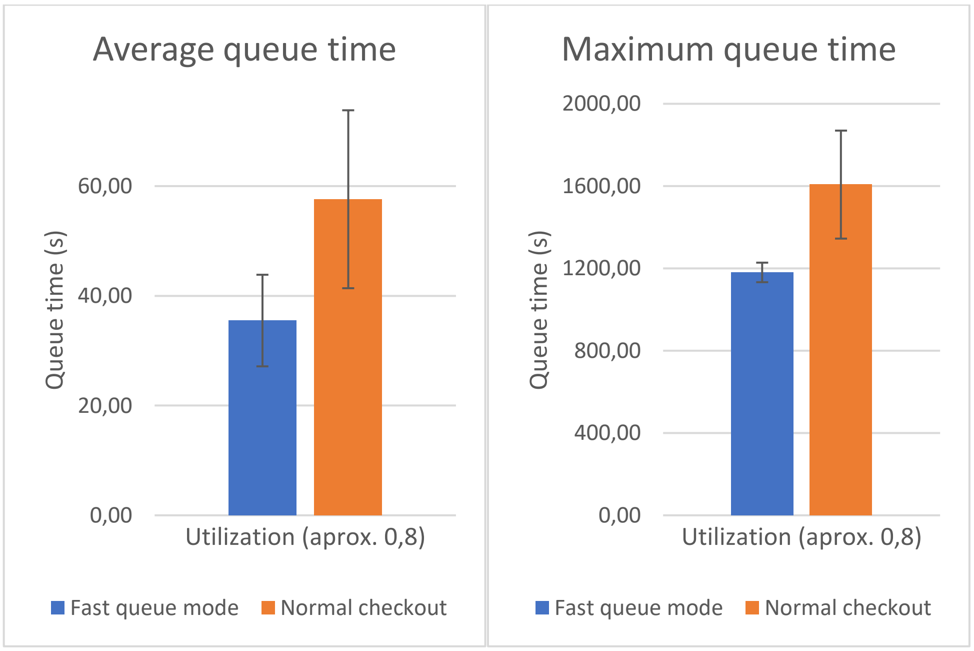

Figure 45 – KPI comparison in Policy B results with the normal checkout mode vs Fast queue mode

5. Conclusions

The Carrefour project proposed the simulation of a supermarket checkout system, considering the queueing of the clients to reach the processing tills, and to analyze the behavior of different implementations of the system.

Using the OMNet++ simulator environment it was possible to develop a set of modules (client source, queues, and tills), organized in two different types of policies for the checkout, resembling the most common supermarket configurations. This designed software then allows the proper configuration and tuning of the parameters of the supermarket as a way to test the response of the system under different workloads.

With all the results obtained with the simulations performed, it was possible to conclude that the simulated checkout policies were able to represent close enough a real supermarket system, with a consistent time response from it. The modeling considered important parameters to simulate the clients, the queues and processing unit, and the system time delays; meaning that the results can be a reliable representation of the behavior of such systems.

The modeling also evaluated the policies using exponential and lognormal types of RV generated for the client demands, producing important results regarding the variance of the distribution and how would the system behave under such conditions. Finally, the comparison between the single queue system vs the individual queue system demonstrated an important discrepancy considering on of the most important KPIs analyzed, the maximum queue time. Under similar conditions it was demonstrated that the multiple queue system produces higher maximum values for this evaluated parameter.

To explore the complexity of the queueing system for this application, a more sophisticate client distribution algorithm could be used for further simulations (maybe considering the already queued clients demand before assigning a new client to a specific queue), meaning that the system would be smarter, being a more accurate representation of a real supermarket checkout system.