Computer-aided Simulations Lab

Lab 9 Report

Luca Agnese s303382

Politecnico di Torino, Turin, Italy

23 November, 2022

Abstract

In this lab we want to add some routines to the queue system implemented in Lab 1 in order to detect the end of the transient and evaluate the accuracy of the simulation. In order to do so, I have implemented a technique which automatically detect the beginning of the steady state (based on visual inspection and heuristics). After having removed the samples collected in the steady state I evaluate the confidence intervals for the accuracy evaluation. Furthermore, a "batch means" technique has been implemented.

1. A short introduction

We are interested in simulating the average delay of customers in the queue and study its behaviour in function of the utilization of the queue. For this purpose, nine values of utilization are considered. Moreover, we have considered three scenarios for the service time of the queue: distributed as exponential with mean equal to one, deterministic equal to one and distributed as hyperexponential with mean one and standard deviation equal to ten. In order to evaluate the accuracy of the results, we consider confidence intervals. Theory on confidence intervals is based on two assumptions: the process is stationary and the observations are independent.

2. Transient Detection

The initial transient (or warm up period) is the time needed for the system to reach its steady state conditions. Since the estimated mean is influenced by the initial transient, we need to remove from our analysis the observations collected during the transient.

In order to detect the end of the transient I have implemented an heuristic method. It is based on the fact that during the transient the variance of observations is typically higher compared to the steady state. My method applies at 'run-time', meaning that while the simulation is running, I check iteratively if the end of the transient is reached. The method consists in analyzing the list of the delays of the customers, in particular I divide the analysis in windows (or batches) and check if the ratio between the mean of one batch and the mean of the previous batch is smaller than a certain threshold (i.e. if one mean is smaller than the other mean plus a certain percentage of the latter). If this occurs, the transient has been identified and it's set equal to the current length of the list of delays (minus one), otherwise the transient has not been identified and the simulation goes on and at next iteration the check is done again. The dimension of the windows is set depending on the value of the utilization of the queue.

To simplify the argumentation, the pseudo-code of the algorithm is reported here.

Algorithm 1 Transient Detection

```
1: if Transient not found then
2:     if We have enough data to identify a batch then
3:         Compute the mean of the current batch
4:         Store such mean in a list
5:         if We have two means to consider then
6:             if $max(mu_1/mu_2, mu_2/mu_1) < 1.1$ then
7:                 Transient identified
8:                 Transient=length of delays list-1
9:             else
10:                Lose infos about previous batch
11:            end if
12:        else
13:            Goes on in the simulation
```

Looking at the plots of the cumulative mean of the delays, we can say that this algorithm achieves good results especially in the cases in which the utilization is not too high. In the most critical cases, when the utilization is greater or equal than 0.9, since the system is not so stable, not always is possible to identify the transient.

3. Batch Means

In order to split a single run in non-overlapping sequences, a "batch means" approach has been adopted. This technique has been implemented in a way such that the number of batches is chosen adaptively in order to achieve a desired degree of accuracy. In order to do so, also this method is applied at 'run-time' i.e. during the simulation. Once the transient is found, the batch means method applies and only the delays after the transient are considered. The minimum number of batches is set to ten, in order to avoid producing intervals too wide, while the maximum number is set to thirty since considering more than thirty batches typically does not produce improvements. The algorithm I have implemented consists in computing, given a certain scenario for the service time and a certain utilization of the queue, the confidence interval whenever we have at least ten batches to analyze. The mean of the delays is computed for each batch. After that follows the computation of the confidence interval. The accuracy metric is strictly related to the width of the confidence interval. In particular, the simulation stops when the semi width of the confidence interval is not greater than a certain percentage of the mean. If this condition is not satisfied, the simulation goes on and the number of batches considered at the next iteration increases by one. Notice that not always the method will converge because in the critical cases, due to instability of the system, is not possible to find a confidence interval that satisfy the condition on the accuracy. Also in this case the dimension of the batches is selected according to the value of utilization, like for the transient. For simplicity I report the pseudo-code of the algorithm I have implemented for this part.

4. Short note on the hyperexponential scenario

Finding the parameters for the hyperexponential was not so trivial. Since we have two constraints, one on the mean and another one on the variance, we have to find the solution of the system of the two equations. This system has two equations and four unknown parameters. The four parameters are p_1, p_2, $lambda_1$ and $lambda_2$. With a graphical tool we can fix the two probabilities such that we can find the intersection between the two curves in the first quadrant (since $lambda_1$ and $lambda_2$ must be positive for definition). After having fixed two of the four parameters, we can find the proper solution of the system obtaining $lambda_1$ and $lambda_2$.

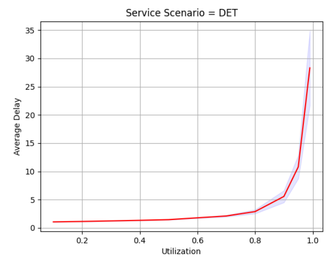
Algorithm 2 Batch Means

```
1: if Transient detected then
2:     if We have enough data to identify a batch then
3:         Compute the mean of the current batch
4:         Store such mean in a list
5:         if We've reached the min number of batches then
6:             Compute Confidence Interval
7:             Check the accuracy
8:             if Accuracy is ok then
9:                 store all useful values
10:                break the loop
11:            end if
12:            if Max number batches reached then
13:                batch means didn't converge
14:                Store values and break the loop
```

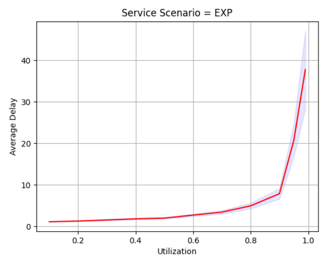
5. Results

Regarding the transient identification, I have obtained quite satisfying results since the transient is identified approximately well in most cases. Only for value large values of utilization (> 0.9), the method struggles to spot it, especially in the exponential and hyper-exponential scenarios where the system is more unstable. For this part, some results are shown in **Figure 2**, highlighting how the method behaves for small, intermediate and large values of utilization.

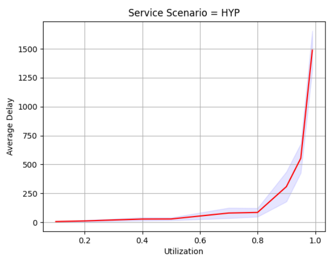
Regarding the evaluation of the accuracy of the results I have found that in the deterministic case the results are accurate. Especially for small values of utilization, the batch means converges as soon as the minimum number of batches is reached and the ratio between the semi-width of the confidence interval and the mean is tiny, resulting in very tight confidence intervals. Obviously, as the utilization increases the confidence intervals become larger. However, in this first case the batch means method tend to converge also for large values of the utilization. In the exponential scenario, the behaviour is more or less the same, but it may happen that, since the system is less stable, for large values of utilization the batch means doesn't converge and the confidence intervals are larger. On the other hand, in the hyper exponential scenario the system is even more unstable, leading to a situation in which the batch means rarely converge resulting in very large confidence intervals. The results are shown in **Figure 1**.



(a) DET scenario

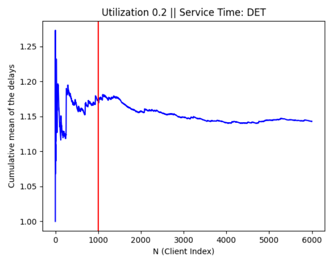


(b) EXP scenario

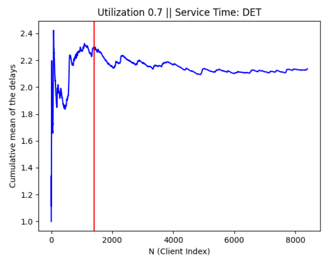


(c) HYP scenario

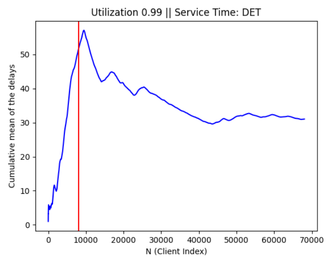
Figure 1: Average Delay in function of the utilization with 95 percent confidence intervals



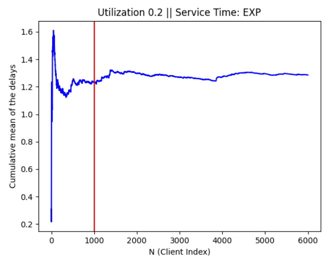
(a) DET case, Utilization 0.2



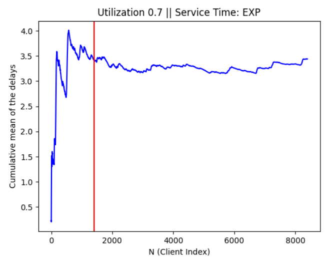
(b) DET case, Utilization 0.7



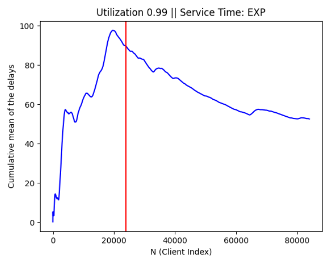
(c) DET case, Utilization 0.99



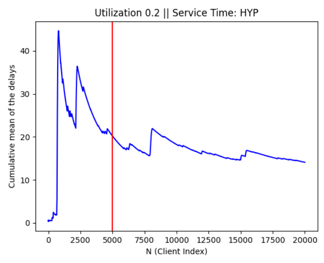
(d) EXP case, Utilization 0.2



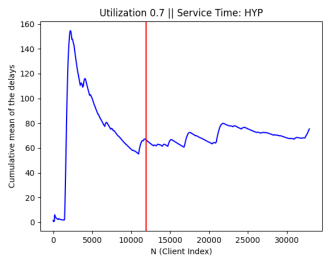
(e) EXP case, Utilization 0.7



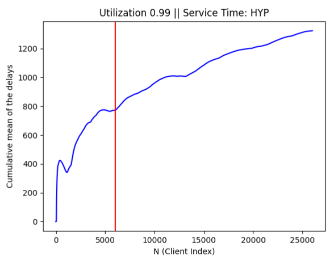
(f) EXP case, Utilization 0.99



(g) HYP case, Utilization 0.2



(h) HYP case, Utilization 0.7



(i) HYP case, Utilization 0.99

Figure 2: Transient identification for the several scenarios and values of utilization