Queuing system
Transient analysis

Giuseppe Esposito
s302179

I. Preliminary Study

A. Overview

In this laboratory we were asked to simulate a single server queuing system (queue discipline: FIFO). It is a system where as soon as the arrival time of a client is scheduled, if the server is already busy, the client is added to the queue. That job for the next customer will start when the server will not be busy anymore. In the meanwhile the delay increases. Then, when the server is idle a random service time is generated for the next customer in the queue and at the end of this process the client will be labeled as "departure".

Once the simulation time has expired and all the delays are collected, I computed the cumulative average delay which has a specific trend: in particular it will increase at the beginning (warm-up transient) and then it will become stable at a certain level. The aim of this laboratory is to perform the batch means algorithm on the values that do not belong to the warm-up transient.

II. Assumptions

Let us assume that:

- The algorithm to generate the schedule the different events in the FES is the same used in the lab01;
- The service time can be: deterministic (always set to 1), randomly generated from an exponential distribution or from an hyper-exponential distribution;
- The inter-arrival time is randomly generated from an exponential distribution
- The parameters of the distributions are set such that the plot of the metric under analysis becomes stable at a certain time_instant;
- The queue has a FIFO policy that means that the first customer that arrived is the first served.

A. Input parameters

The input parameters set to study the different cases are:

- Utilization levels (U): list of possible values of utilization which is the average amount of work requested to the server from customers arriving in the time unit. From the definition it can be expressed as relation between the parameters of the exponential and hyper-exponential distribution. In particular: if $\lambda = 1 => \frac{1}{u}$;
- Confidence level of the different confidence intervals always set to 95%;
- Distribution type (distribution): possible distributions by means of which I generate the random instances of the service time;
- Maximum simulation time (SIMULATION_TIME): maximum time the simulation can lasts.

B. Output metrics

The metrics under analysis are: the average delays for each level of utilization and the average delay for each batch of the plotted average delays array for each departure event.

III. Main Algorithm

Algorithm 1 Hyper-exponential instance generation

$p \leftarrow 0.5$
$\lambda_1 \leftarrow \frac{1}{6}$
$\lambda_2 \leftarrow \frac{1}{8}$
Sample from a uniform distribution U(0,1)
if u $\leq$ p:
  Sample from an exponential distribution with parameter λ_1
else
  Sample from an exponential distribution with parameter λ_2

Algorithm 2 Warm-up transient removal

Given the array of the cumulative average delays and a utilization level
$window \leftarrow$ Empty list
$length\_of\_the\_windows \leftarrow u \times 10000$
Compute he number of windows over all the array
Define a adaptive threshold with respect to the value of the utilization
for $window\_idx$ **in** range(1; $num\_windows$) **do**
  $window \leftarrow$ slice of the initial array from the end of the previous window to the length of the current window times the window_idx
  Compute min_window and max_window
  ratio $\leftarrow \frac{min\_window}{max\_window}$
  if $ratio > threshold$:
    cut the initial array till the end
  return $cutted\_array, window\_idx$
end for

The parameters set at the beginning of the algorithm 1 are:

- λ_1 parameter of the first exponential;
- λ_2 parameter of the second exponential;
- probability p: probability to sample from an exponential with λ_1.

λ_1 and λ_2 are parameters computed from the system with 2 equations in 3 variables which contains the theoretical expression of the expected value and of the variance of the distribution and it is undetermined, so it has infinite solutions so I have to fix p in order to find the values of the parameters. Since I did not want to favour a parameter rather than another, I put p = 0.5 so that the probability to choose one or another parameter is the same.

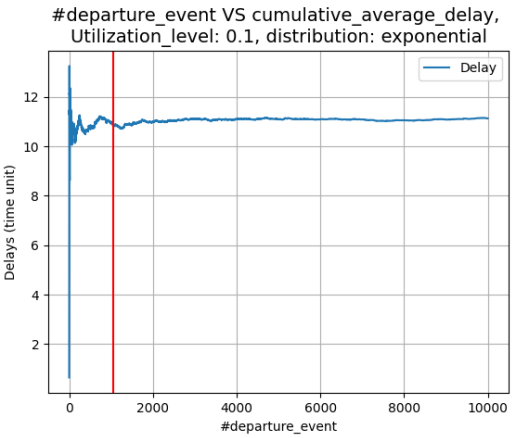
The idea of the algorithm 2 comes from the definition of "warm-up transient": it is the virtual time needed for the system to reach the steady state conditions after starting from a given initial conditions (I start from an empty queue and this is my initial condition, because the delays and also the cumulative average delay starts from 0). It consists in checking for each window of the array whether the gap between the min and max is greater than a threshold that is properly tuned with the setup of the simulator, if so just keep the remaining part of the array which is the part without the warm-up transient. I computed the ratio and not the difference in order to avoid the problem related to the absolute value of delay level the current window is looking.

After removing the transient I am able to perform the whole batch means algorithm. It consists in 2 main steps:

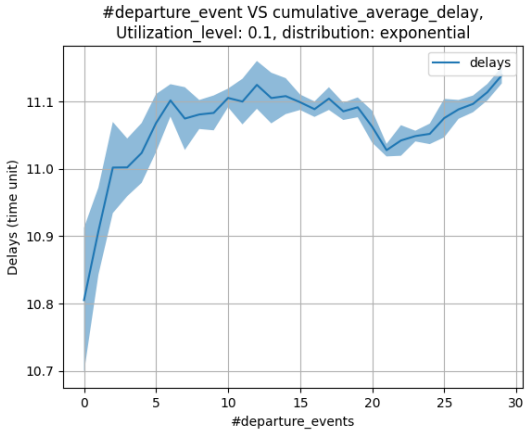
- Find the correct number of batches in which the part without the transient have to be divided. This is computed by checking whether the width of the confidence interval for a given number of batches is greater than a value that is a re-scaled value of the utilization level. If so collect another batch otherwise return the current number of batches in which the input array is split.
- Compute the mean for each batch and also the confidence interval for each value of the mean.

IV. Results

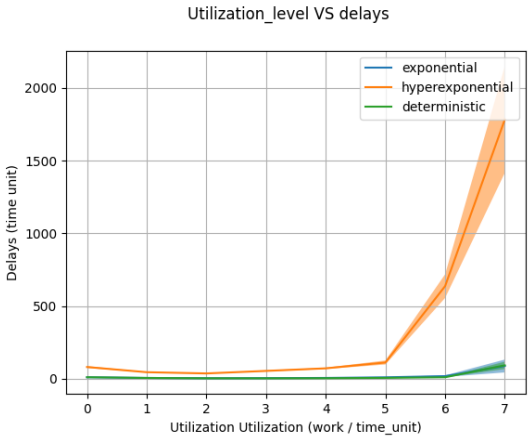
These are the results that came out from my simulation.



In figure IV we can see the number of departure events (I focused on that since the delays are computed as $current\_time - client\_departure.arrival\_time$ every time a client is popped from the queue list) and on the y-axis we have the corresponding values of cumulative average of delays. We can easily notice where is the warm-up transient and the red line determine the end of the transient. The result I obtained is quite accurate and the same holds for the other distributions and values of u that are stored in the subfolder 'graphs'. As aforementioned and as requested I decided to adapt the value of accuracy manually with respect to the simulator setup. So the accuracy level in my algorithm is the value over which I stop removing the transient (i.e. threshold).



After applying the batch means algorithm, we can notice from IV that the curve needed 30 batches which is a borderline case because when the number of batches used is higher that 30, it means that the algorithm is not producing significant improvement on the confidence interval because the samples that i consider dividing in batches may become dependent from eachother



In the end another noteworthy result is the comparison between the trend of the average delays versus the utilization level with respect to the reference distribution shown in figure IV. The trend of the delays when the simulation is performed

with an exponential distribution is almost the same when it is performed with a deterministic one, but the same does not hold for the hyper-exponential distribution which has a way faster increasing trend.

In the appendix you can find some other graphs which are representative examples of some other cases under analysis.

Appendix