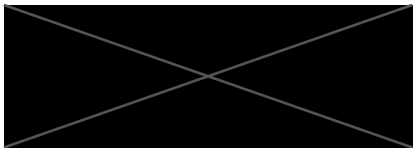
Transient Analysis for Queues



Abstract*— This report is related to the simulation of long term queues and determine the transient point also known as warm-up point based on their delays. To do so, we consider three different distributions (deterministic, exponential and hyper-exponential) for service time and a list of utilization numbers. To find the transient point and number of batches to reach specific accuracy, we use automated algorithm to find the latency in the function of utilization.

I. STRUTURE OF SIMULATOR

There are multiple implementations for queue problem with different scopes but we use single server queue case as a backbone of our simulator.

A. Assumption:

To have a dependency on service time, each random arrival time is generated by an exponential distribution depending on service time with three different distributions as follows:

- Deterministic with constant 1
- Exponential with mean 1
- Hyper-exponential with mean 1 and standard deviation 10. Due to the inefficient implementation of the actual algorithm, we use an alternative algorithm as below:

```
input exponential parameter 1 lambda1
input exponential parameter 2 lambda2
input p
generate uniform sample u
if u >= p:
        return exponential(lambda1)
else:
        return exponential(lambda2)
```

In order to find λ_1 and λ_2 for hyper-exponential case, we need to solve a linear system with its parameters. After solving following system, the positive values are $\lambda_1 = \frac{1}{6}$ and $\lambda_2 = \frac{1}{8}$.

$$\begin{cases} \frac{p}{\lambda_1} + \frac{1-p}{\lambda_2} = 1 \\ \frac{2p}{\lambda_1} + \frac{2(1-p)}{\lambda_2} = 100 \\ p = constant \end{cases}$$

B. Inputs

The parameters are:

- Initial batch number to perform batch means
- Maximum simulation time
- Accuracy level
- Transient removal threshold

C. Data structure and algorithm:

To simulate the queue, there are two main functions client's arrival time and departure time. The generated events are stored in FES which is as FIFO queue. Other functions to perform the whole simulation are explained as follows:

1. Transient Removal: this function finds the knee point of the delay's plot automatically:

```
input starting array
input starting windows number
input window length (prop. to u)
input threshold (manually tuned)
divide array into #windows
for each window:
 compute min and max for each window
 if min/max > threshold:
        go to next window
        break
```

2. Batch Means: in order to have dependency in simulations, there two approaches, using different seeds and batch means. In this code, we benefit from batch means strategy in which there is independency between each batch. To do so, following steps are deployed:

```
Run 1:
#find the right number of batches
input random threshold
input delays array
input number of batches = k
divide delays array into k batches
for batch in array batches:
 if batch_size > threshold:
        increase number of batches
        log batch width,
        break
Run 2:
#adjust threshold and perform batch means
set threshold = average batch width
recompute number of batches
#using new threshold
divide delays array in batches
```

compute batch's mean and 95% C.I.

D. Output metrics:

Having said that we are looking for a relationship between combination of utilization/distribution and delays, the algorithm provides plots for each combination with their transient points and also average delays in the function of utilization values [0.1, 0.2, 0.4, 0.7, 0.8, 0.9, 0.99].

II. RESULTS

Due to the large number of plots, we show one plot for transient part which is in Fig1. it can be seen that the algorithm finds the knee point with acceptable precision. More graphs are provided in delivered file.

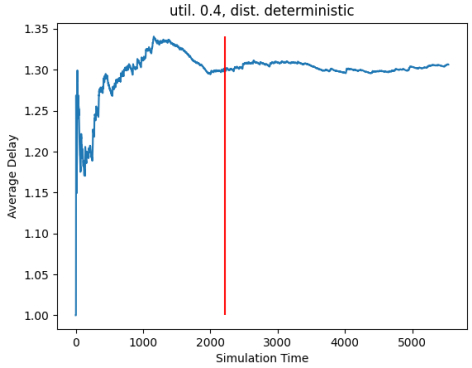


Fig1: warm-up point identification for a specific combination

Finally, Fig2 illustrates the relationship between average delays and utilization. From this plot, we can infer that hyper-exponential distribution is not a good choice to implement the algorithm.

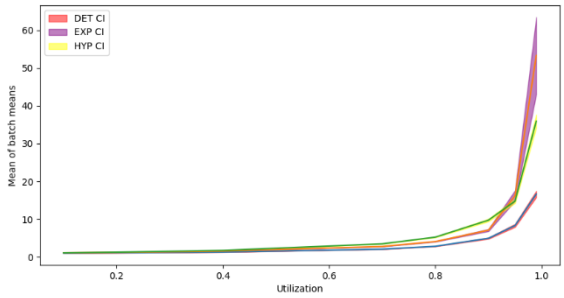


Fig3: not transient_cleaned delays vs utilization for all three distributions