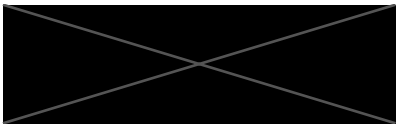
Priority Queue with k Serever



Abstract— This report is related to the simulation of priority queue with k server. In a sense that, when a client with high priority arrives, they should be served without any waiting time in the queue. If there is a client with low priority in the server, they should be discarded and place them at the end of the queue if the queue is not full.

I. STRUTURE OF SIMULATOR

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

A. Assumption:

Each arrival time is simulated based on Poisson distribution with different lambdas.

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:

- Number of servers
- Length of the queue
- Simulation time
- Arrival rate
- Service time rate

C. Data structure and algorithm:

There is a class for generating servers. We have one class to produce priority queue in which there are two functions:

Put: to simulate client arrival with Poisson distribution

Get: to simulate client departure based in service time

For storing the results of simulation corresponding to each distribution, python list data structure has been used.

In terms of algorithm, each server is busy until the client is served completely unless a client with high priority (which is generated by priority class) arrives, then the server is interrupted to serve the high priority client, and low priority client is dropped and place to the end of the queue if the queue has empty space. In order to plan service time of each element in the queue, we keep track of departure time for each server separately.

D. Output metrics:

We are looking for a relationship between combination of lambda/distribution and delays for different kind of client (low and high priority), the algorithm provides plots for each combination and also average delays in the function of lambda values [0.2, 0.4, 0.8, 1.4, 2.0, 2.4, 2.8].

II. RESULTS

Due to the large number of plots, we show one plot for average delay time which is in Fig1. More graphs are provided in delivered file.

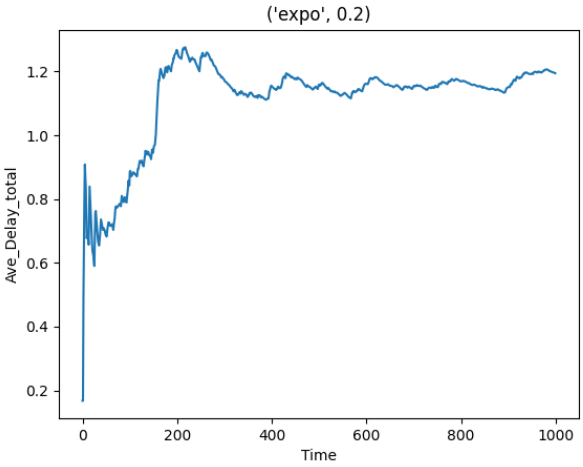


Fig1: Total average delay time for specific combination

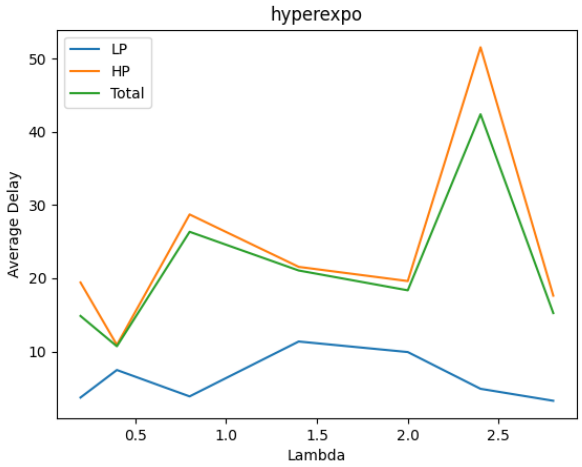


Fig2: Relationship between average delay and lambda for each specific distribution

Finally, Fig2 illustrates the relationship between average delays and lambda.