Transient Analysis

Davide Di Mauro
Politecnico di Torino
Student ID: s306089
s306089@studenti.polito.it

***Abstract*—The objective of this homework is the simulation of a single server queue and then analysing its delay. This analysis is done implementing a method to automatically remove the transient and then use the batch means method to compute the confidence interval of the measure of interest.**

I. Assumptions

In order to simulate this system we needed the following assumptions:

1) In order to detect transient in a reasonable simulation time we consider the case of a **finite size queue**, if not the cumulative trend would be too unstable
2) Instead of using the actual delay of each customer in the queue we use the size of the queue at each event, this is justified by the fact that the two statistics are proportional
3) Since the trend of the delay is really unstable we use the cumulative mean to detect the transient

II. Queue Simulation

A. Simulation Parameters

The queue has 3 input parameters:

1) The maximum size of the queue: **set to 10** in all of the experiments
2) The utilization: modeled as an exponential random variable with rate λ
3) The service time scenario

Figure 1 shows the cumulative mean of the occupancy of the queue for different values of the utilization λ. We can see that after an initial spike (i.e. the transient) the cumulative mean stabilize at a value that is proportional to λ.

B. Service Time Scenarios

3 different scenarios were used to model the service time:

- Exponentially distributed with μ = 1
- Deterministic = 1
- Hyper-exponentially distributed with μ = 1 and σ = 10

III. Transient Detection

As described in Algorithm 1 the objective of this algorithm is to detect the end of the transient, in order to remove it from the data to utilize. The idea of this implementation is to take the cumulative mean of the delay and then see if the trend is decreasing, since usually there is a drop in the average occupancy of the queue after it reach saturation, and this can indicate with good enough approximation the end of the transient. To automatically detect if the transient is ended the algorithm iterate over the data, if more than n values in position $i+1$ are lower than the ones in position i the transient is detected. The efficacy of this method is shown in Figure 2; please notice that the green line (arrival rate = 0.7) is longer than the others because the collection of 5 more batches was requires to compute an accurate confidence interval, as described in Algorithm 2

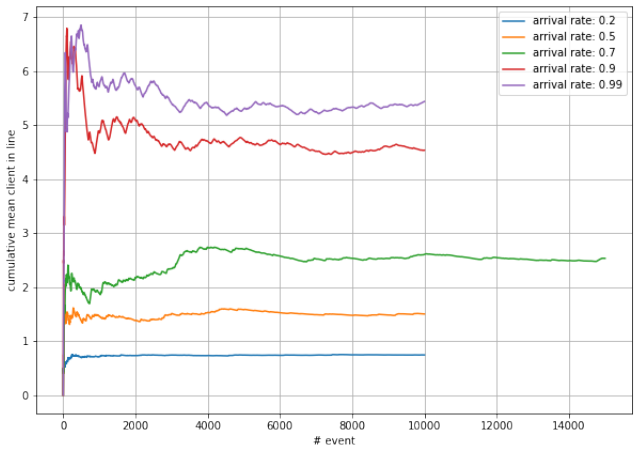

Fig. 1: Exponential service time

Algorithm 1 Transient Detection

Input: $Delay$ of the queue
Output: i index of the end of the Transient

```
cum_mean ← cumulative_mean(Delay)
count ← 1
n ← 1000 · λ                          ▷ λ = service rate
for i ← 1 to length(cum_mean) do
    if cum_mean[i − 1] > cum_mean[i] then
        count ← count + 1
        Collect a further batch
    end if
    if count > n then
        return i
    end if
end for
```

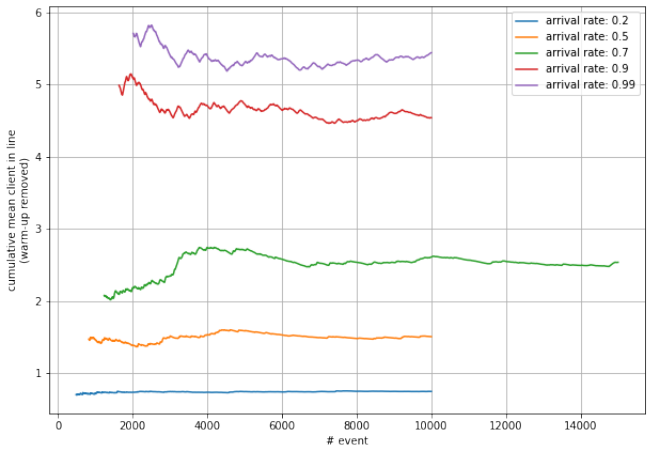

Fig. 2: Transient detection Exponential service time

IV. Batch Means

Batch mean (Algorithm 2) is a technique used to compute the confidence interval I_n of a selected statistic without doing multiple runs with different seeds. This is done dividing the data in batches and compute the average for each batch; then this average is used to compute the confidence interval. This approach is based on the assumption that in a long simulation each batch can be seen as independent and only the point at the boundaries are correlated

The results of the batch mean method for the **95%** confidence interval are shown in Tables I, II and III

Algorithm 2 Batch Means

Input: cum_mean of the queue with the Transient removed
Output: I_n, x Confidence Interval and mean of the input

```
n ← 10
batch_size ← 1000
Collect n batches of size batch_size
p ← 1 − α
while True do
    I_n = [x − z, x + z]        ▷ Compute confidence interval
    if 2z/x > p then
        n ← n + 1
        Collect a further batch of size batch_size
    else
        return I_n, x
    end if
end while
```

TABLE I: Batch Mean results Exponential *(95% c.i.)*

	$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 0.7$	$\lambda = 0.9$	$\lambda = 0.99$
I_n	(0.738,0.744)	(1.469,1.529)	(2.468,2.586)	(4.564,4.706)	(5.298,5.436)
$\hat{x}$	0.741	1.499	2.527	4.635	5.367

TABLE II: Batch Mean results Deterministic *(95% c.i.)*

	$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 0.7$	$\lambda = 0.9$	$\lambda = 0.99$
I_n	(0.724,0.736)	(1.193,1.208)	(1.877,1.916)	(4.099,4.296)	(5.322,5.485)
$\hat{x}$	0.730	1.201	1.897	4.197	5.403

TABLE III: Batch Mean results Hyperexponential *(95% c.i.)*

	$\lambda = 0.2$	$\lambda = 0.5$	$\lambda = 0.7$	$\lambda = 0.9$	$\lambda = 0.99$
I_n	(2.894,3.023)	(5.498,5.620)	(6.026,6.262)	(6.548,6.605)	(6.554,6.616)
$\hat{x}$	2.960	5.560	6.234	6.577	6.585

V. Conclusions

In conclusion, the following observation were made

1) The higher the arrival rate, the higher the average delay of the queue
2) Increasing the arrival rate makes the trend of the delay more unstable, consequently increasing the confidence interval
3) Comparing the different cases we notice that the Exponential and Deterministic cases are quite similar, with the difference that the Deterministic case is a bit more stable, hence the transient is better isolated.
   As for the Hyperexponential case, the transient of the case with arrival rate = 0.02 is much more unstable compared to the other two cases discussed above, and in general the queue stabilizes at a higher level of client in line, for all the values of arrival rate.

APPENDIX

In the Appendix are reported the plots for the same experiments described in the report but for the Deterministic (Figure 3, 4) and Hyperexponential (Figure 5, 6) service time.

The results with no limit on the queue size are not included in the report, due to problem in detecting the transient, specially for the case with arrival rate = 0.99, but are still available on GitHub at the following link: https://gitfront.io/r/user-3507277/7y8deXU57Z7N/ComputerAidedSimulationLab/tree/Laboratories/transient_analysis/ in the "res" folder.

Fig. 3: Deterministic service time

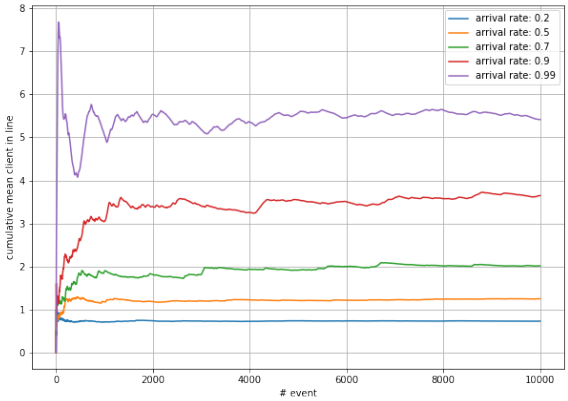

Fig. 4: Transient detection Deterministic service time

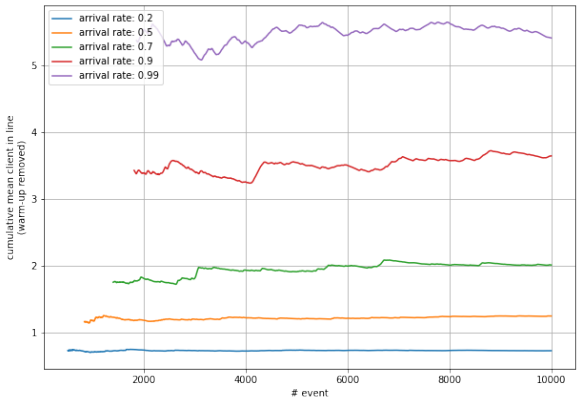

Fig. 5: Hyperexponential service time

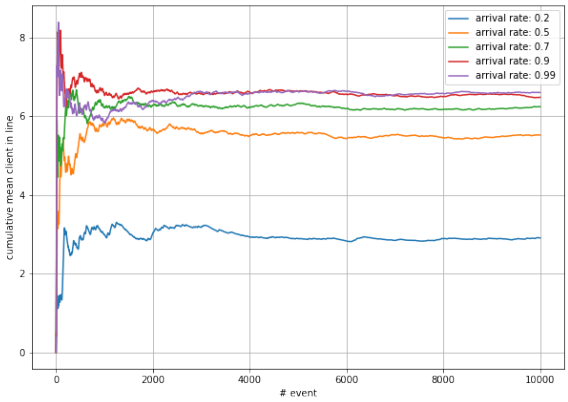

Fig. 6: Transient detection Hyperexponential service time