# CS-350 - Fundamentals of Computing Systems
# Homework Assignment #2 - EVAL

Due on September 28, 2023 — Late deadline: September 30, 2023 EoD at 11:59 pm

*Prof. Renato Mancuso*

**Renato Mancuso**

# EVAL Problem 1

In this EVAL assignment we will start to explore how to discover characteristics of incoming workload, we will dive into how to measure the overheads of our systems, and we will also study the properties how queues evolve in response to varying traffic conditions.

a) First thing first, let's try to make sense out of the workload that is coming from the client. You might recall that by invoking the client with various values of the `-a` and `-s` parameters significantly impacts the load seen by your server. Now it is time to reverse-engineer the characteristics of that traffic. Start with the following to collect the report of 1,000 packets handled at the server:

```
./server_mt 2222 & ./client -a 6 -s 10 -n 1000 2222
```

Now, isolate only the lengths of the requests as they are sent from the client. With that, produce a plot of the distribution of the request lengths you have collected. The distribution plot should have on the $x$-axis a set of time bins, e.g., from 0 (included) to 0.005 (excluded), from 0.005 (included) to 0.010 (excluded), and so on in steps of 0.005 increments. Given each transaction, look at its length. If it is in the range between 0.005 and 0.010 seconds, it falls in the second bin; if it is in the range between 0.010 and 0.015, it falls in the third bin and so on.

On the $y$-axis, plot how many requests fall in each bin! But do not plot the raw count. Rather, normalize that value by the total number of requests you are plotting. In this case, 1,000. Hooray! You have produced a distribution plot.

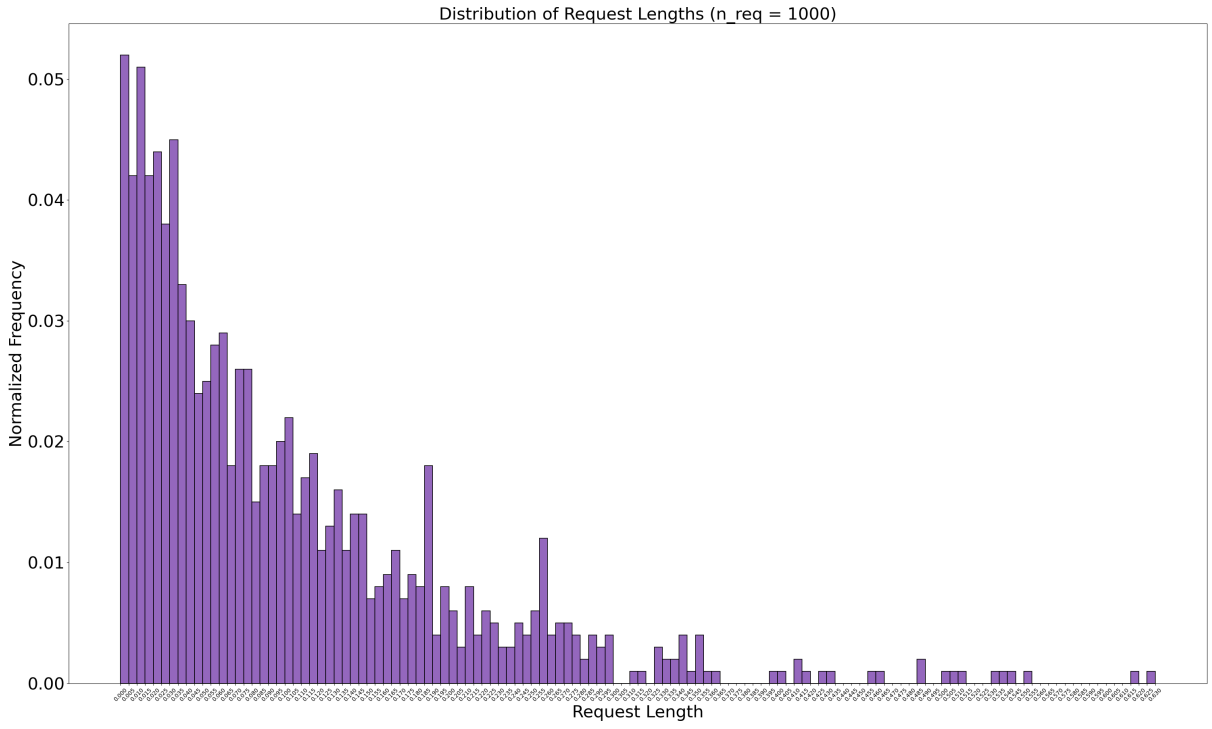


Figure 1: 1a: Distribution of Request Lengths

b) By using the same procedure used in the previous part, produce a distribution plot of the inter-arrival time between any two subsequent requests. Say that request R0 is sent (look at the sent timestamp!) arrives at $t_0 = 10s$ and R1 is sent at $t_1 = 15s$, then the inter-arrival time between them is $t^{iat}_{0,1} = (t_1 - t_0)$.

Compute all the 999 inter-arrival times you have observed and plot their distribution just like you did above, except that this time you will normalize by 999.

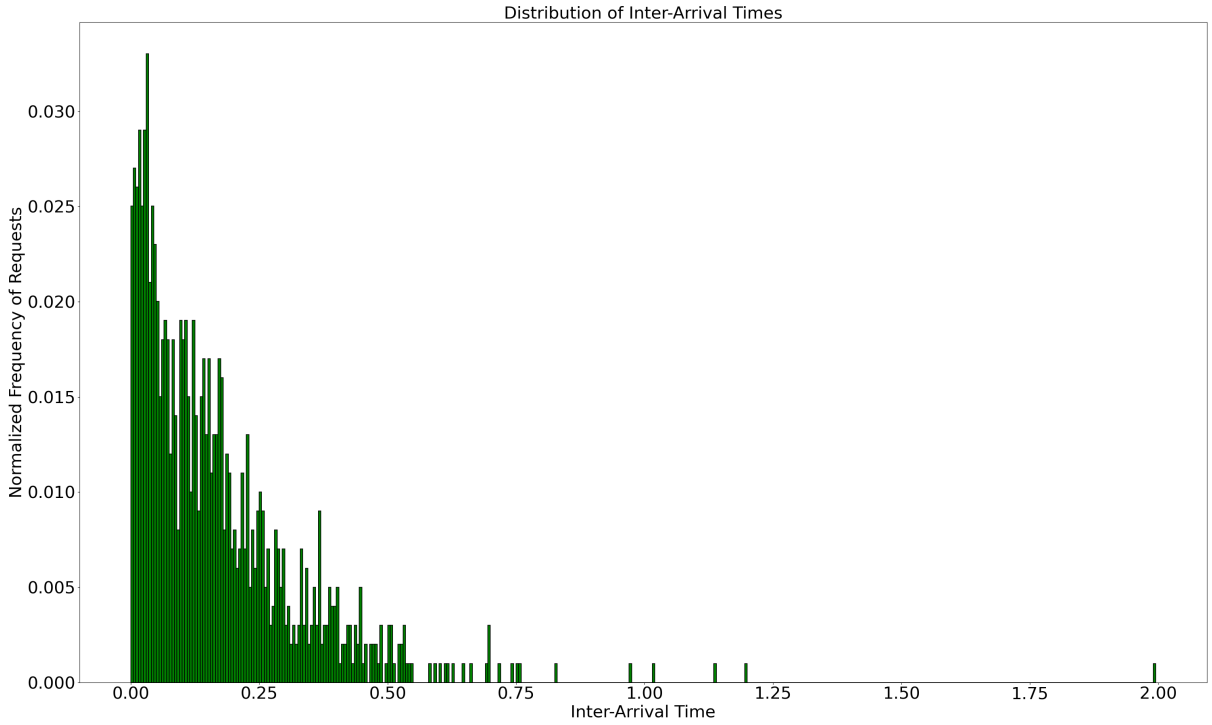


Figure 2: 1b: Distribution of Inter-Arrival Times

c) Time to reverse-engineer things! Let's start from the distribution of request lengths. Use your favorite programming language to generate 10,000 samples from the following theoretical distributions:

(1) A Normal distribution with mean 1/10 and standard deviation 1;

(2) An Exponential distribution with mean 1/10;

(3) A uniform distribution with mean 1/10.

Plot these distribution together (on the same plot, just different lines) with the distribution you previously acquired from your server run. Thus, your plot should have a total of 4 different lines (I suggest having lines instead of bars for this plot) in it. Then, comment on which ones of the curves matches more closely with the experimental data. If there is one of them that matches remarkably close, you have successfully reverse-engineered the characteristics of your input load!

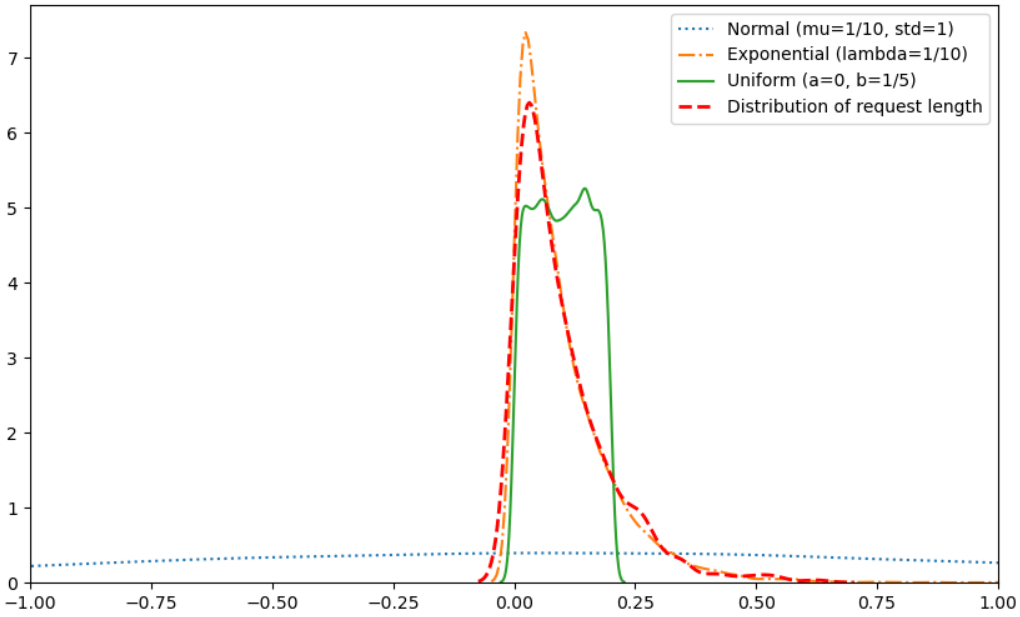


Figure 3: 1c: Distribution Comparisons (`req_len`)

**Among these distribution, the distribution of request lengths matches most closely with that of Exponential Distribution.**

d) Do the same with the inter-arrival times. But this time, compare it with the following three references:

(1) A Normal distribution with mean 1/6 and standard deviation 1;

(2) An Exponential distribution with mean 1/6;

(3) A uniform distribution with mean 1/6.

Produce the comparison plot and comment on the match between the experimental and theoretical curves. At this point, can you tell me what the `-a` and `-s` parameters control, exactly?

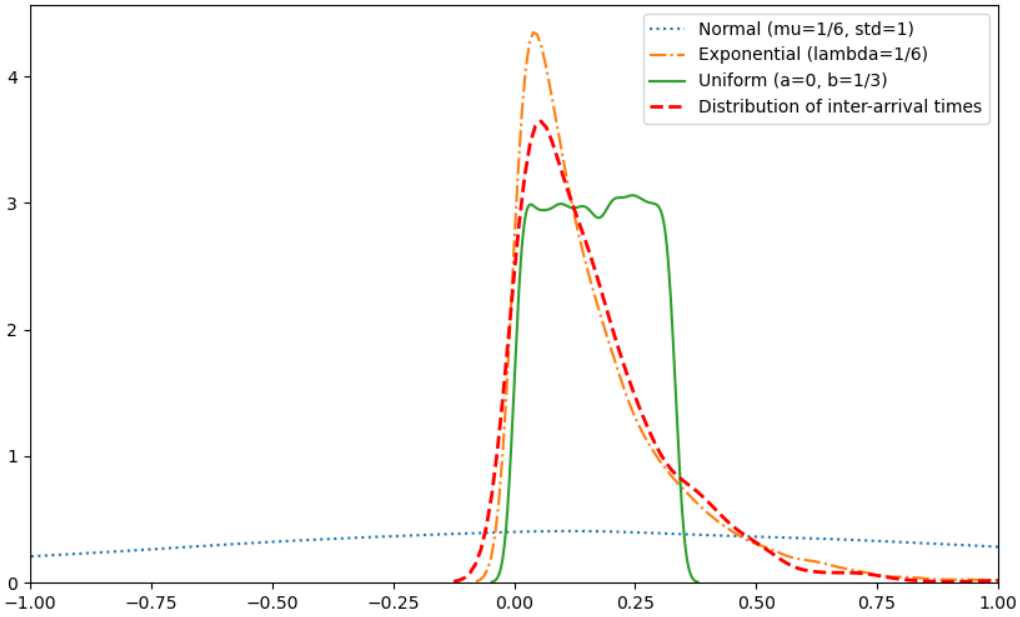


Figure 4: 1d: Distribution Comparisons (`inter_arrival`)

**Among these distributions, the distribution of inter-arrival times matches most closely with that of the Exponential Distribution. I can tell that parameter `-a` controls the arrival rate, and parameter `-s` controls the service rate.**

# EVAL Problem 2

In this EVAL problem, we will start to study the relationship between utilization, throughput, and queues in response to changing load conditions.

a) First thing first, learn how to take a *good* queue size average. A good queue average measurement should consider the amount of time the queue remains in a certain state, i.e., it should be a *timed* average of the queue length.

Let us make an example. Say that your queue at $t = 0$ has 3 elements in it, and it stays that way until time $t = 9$ sec, at which point the queue becomes empty. If you do not consider time, the average size would be $q = (3+0)/2 = 1.5$. But this is incorrect: most of the time we see the queue with 3 elements, and only at the end with 0. So an average of 1.5 seems wrong.

The right way to take the average is by weighting the queue state by the time it stays in that state. Thus, the right way to calculate $q$ in our example is $q = 3 \cdot (\frac{9}{10}) + 0 \cdot (\frac{1}{10}) = 2.7$. You can see how this is a better average of queue size over a 10 seconds time window starting from time $t = 0$.

Now, measure the queue length for the case where your queue-enabled server is invoked with the following parameters:

```
./server_q 2222 & ./client -a 14 -s 15 -n 1000 2222
```

Use the queue snapshots produced by the worker thread to measure the queue size as it was observed after each request was observed. Use the time elapsed between two subsequent queue snapshots to weigh that size towards the total average.

$$\textbf{Average queue length} = \frac{\sum_{n=0}^{998}(request(n+1).\textbf{completion\_time} - request(n).\textbf{completion\_time})}{\textbf{request}(999).\textbf{completion\_time} - \textbf{request}(0).\textbf{completion\_time}} = 6.190502$$

b) Now let us repeat the computation of the queue size average as in Qa) but this time sweep through the `-a` parameter passed to the server. In particular, run the first experiment for a value of 1; then a second time with a value of 2; and so on until and including the case where the value is 15. Thus, you will run 15 experiments in total. This might take a while, so try to automate the runs and dump the results into a file for later analysis.

By reusing what you learned in `hw1`, also extract utilization and response time averages from each of the 15 experiments. Now, you should have three sets of 15 values each: (1) utilization, (2) average response time, (3) average queue length.

Finally, produce a plot that depicts the trend of the average response time (on the $y$-axis as line 1) and average queue size (on the $y$-axis as line 2) as a function of the server utilization $x$-axis. What relationship do you discover between how response time and queue length averages evolve as a result of increasing utilization?

**Lists of utilization, average response time, and average queue length for arrival rate $\in$ [1, 15]**

**utilization**
= sum of $\frac{\text{request\_length}}{\text{total\_time}}$
= [0.06682265474553331, 0.13362848871954988, 0.2004214763983279,
0.2671996564286006, 0.3339647794673325, 0.4007271269480305,
0.4674871719441579, 0.5342076447247646, 0.6009053635041492,
0.6676545444256989, 0.734343163716828, 0.8010552431816064,
0.8602937769286202, 0.9163125686769626, 0.9699786875175643]

**average response time**
= the mean of each request's (completion_time − sent_time)
= [0.07075279399985447, 0.0764641399999964, 0.08273956500054919,
0.09085794099979103, 0.10008085300028324, 0.11262582400051178,
0.1277083510000666, 0.14559463300005882, 0.16753776500027742,
0.2019878000000608, 0.27179698700006705, 0.3687716470001324,
0.49416217199951645, 0.6614274130002595, 0.977861283000384]

**average queue length**
$= \frac{(\text{completion\_time}_{n+1} - \text{completion\_time}_n) \times \text{queue\_size}}{\text{total\_time}}$
= [0.004609311917769172, 0.019080345463133523, 0.04599441052870688,
0.09109495706706353, 0.16297779924308112, 0.26460885570972936,
0.40872348676235754, 0.6085979141677774, 0.8904177909128013,
1.346267867585406, 2.242808465561838, 3.3290580629853266,
4.546144576016295, 6.184263227963583, 11.025525378672038]

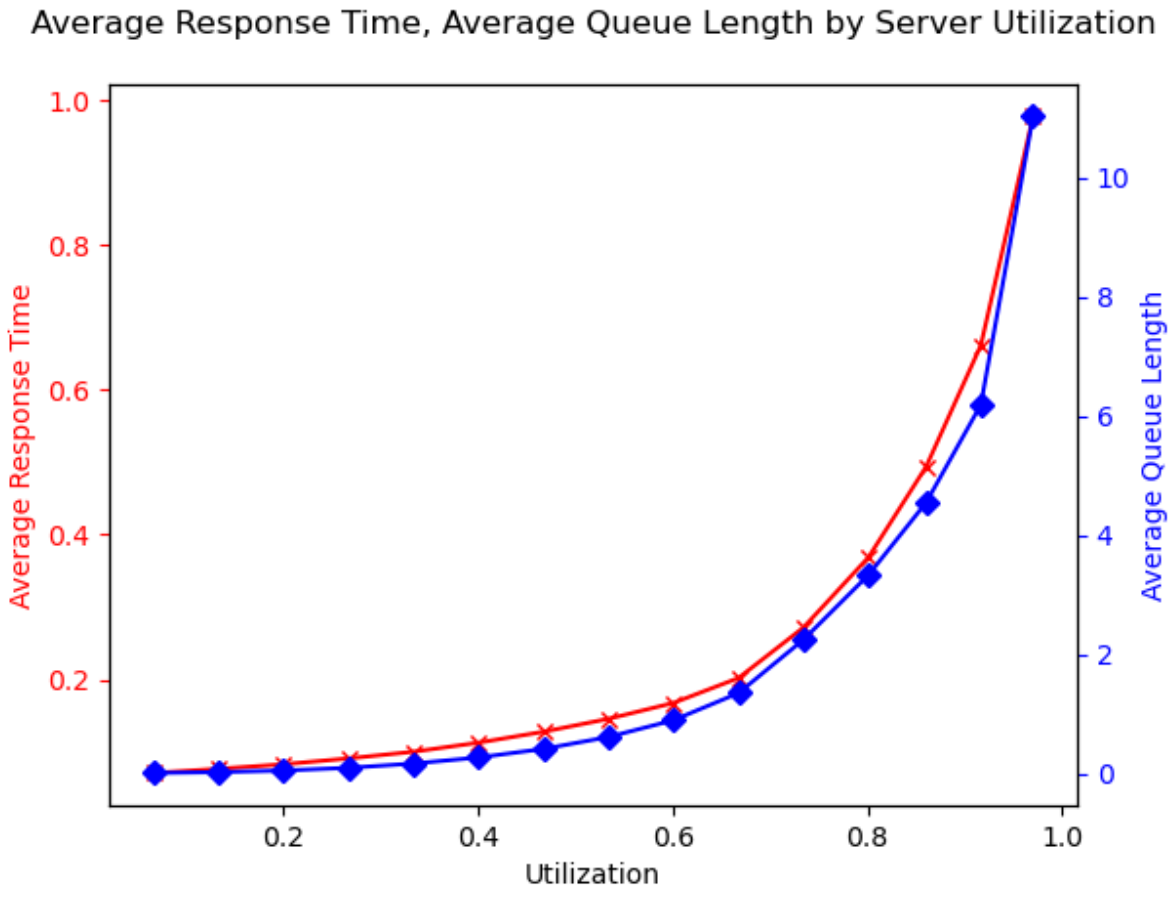


Figure 5: 2b: Avg Resp Time, Avg Queue Len by Utilization

**As utilization increase, response time and queue length both increase as well.**

Problem 2 continued on next page...

c) By looking at the plot produced above, can you conclude that there is some fixed proportional relationship between queue length and response time? Is there something in the theory covered so far capable of modeling this relationship?

The plot above suggests a proportional relationship between queue length and response time: **as utilization increases, response time and queue length both increase as well.**

Our program has a parent process delegating work to a child thread, which behaves as an **M/M/1 system**. In the context of the M/M/1 system, the relationship is given by utilization $\rho = \frac{\text{arrival rate}(\lambda)}{\text{service rate}(\mu)}$. Average queue length corresponds to the average number of requests currently in the system (i.e., $q$), which is derived by $\frac{1-\rho}{\rho} = q$. Average response time $T_q$ is derived by $\frac{1}{\mu-\lambda} = T_q$.

**Therefore, we can derive:**

**(a)** $T_q = \frac{q}{\lambda}$**, which implies that response time is proportional to the average number of requests in the system with the coefficient of the arrival rate.**

**(b) In an M/M/1 system, as $\rho$ tends to 1, average queue length and average response time go to infinity accordingly.**