CS-350 - Fundamentals of Computing Systems
Homework Assignment #3 - EVAL

Due on October 16, 2023 — Late deadline: October 18, 2023 EoD at 11:59 pm

Prof. Renato Mancuso

Renato Mancuso

EVAL Problem 1

In this EVAL assignment we will understand how seemingly insignificant changes in the input traffic distribution can lead to measurable differences in the behavior of queues and thus in the perceived service.
IMPORTANT: For this EVAL, make sure you download the latest version of the client binary! To confirm that this is the correct version, the client must print a line at startup that states:
`[#CLIENT#] INFO: CS350 Client Version 3.0`

a) In the previous EVAL assignment, you discovered that the client sends requests to the server with inter-arrival times that are exponentially distributed with mean $1/\lambda$ where λ is the arrival rate passed via the parameter `-a`. Also, the request lengths are exponentially distributed with mean $1/\mu$ where μ is the service rate passed via the parameter `-s`. Great! But as it turns out, that's just the *default* behavior of the client. Let's discover what else the client can do.

Which distribution the client uses is controller with an extra parameter `-d <dist. number>`, where `<dist. number>` is a number from 0 to 2. When passing `-d 0` you will be using the default exponential distribution. But what are the other two distributions? And does the `-d <dist.number>` parameter control both inter-arrival and service times?

To begin answering these questions, run your server and client with the following parameters (let it run, it will take about 5 minutes):

```
./server_lim -q 1000 2222 & ./client -a 4.5 -s 5 -n 1500 -d 1 2222
```

Notice that the client is requested to generate traffic according to distribution 1. Just like you did in HW2, plot the experimental data of inter-arrival times and request lengths and recover the type and parameters of the distributions used by the client when `-d 1` is passed.

Hint: *there is a some guesswork involved in recovering the distribution parameters. Start by looing at the shape of the distribution produced by the collected data and make a guess about which distribution might be. Setting the mean will be easy if you think about it. If there is a standard deviation to set, explore integer fractions or multiples of the mean.*

Problem 1 continued on next page...

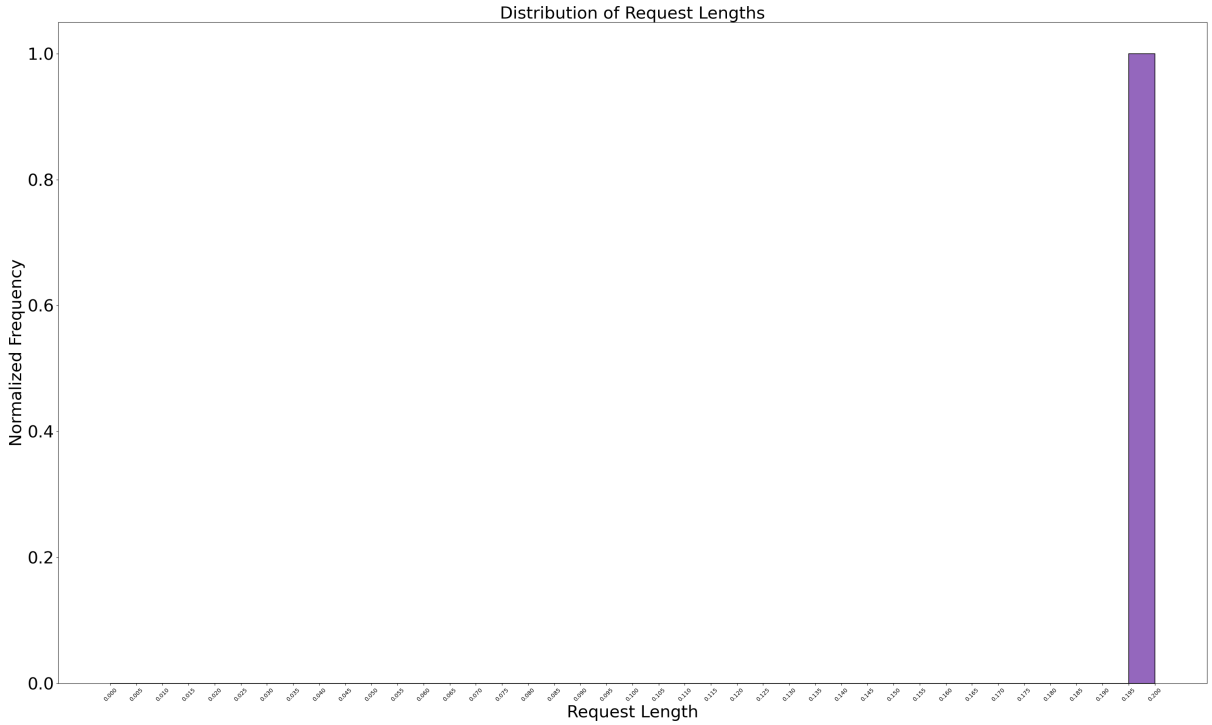

Figure 1: 1a: Distribution of Request Lengths

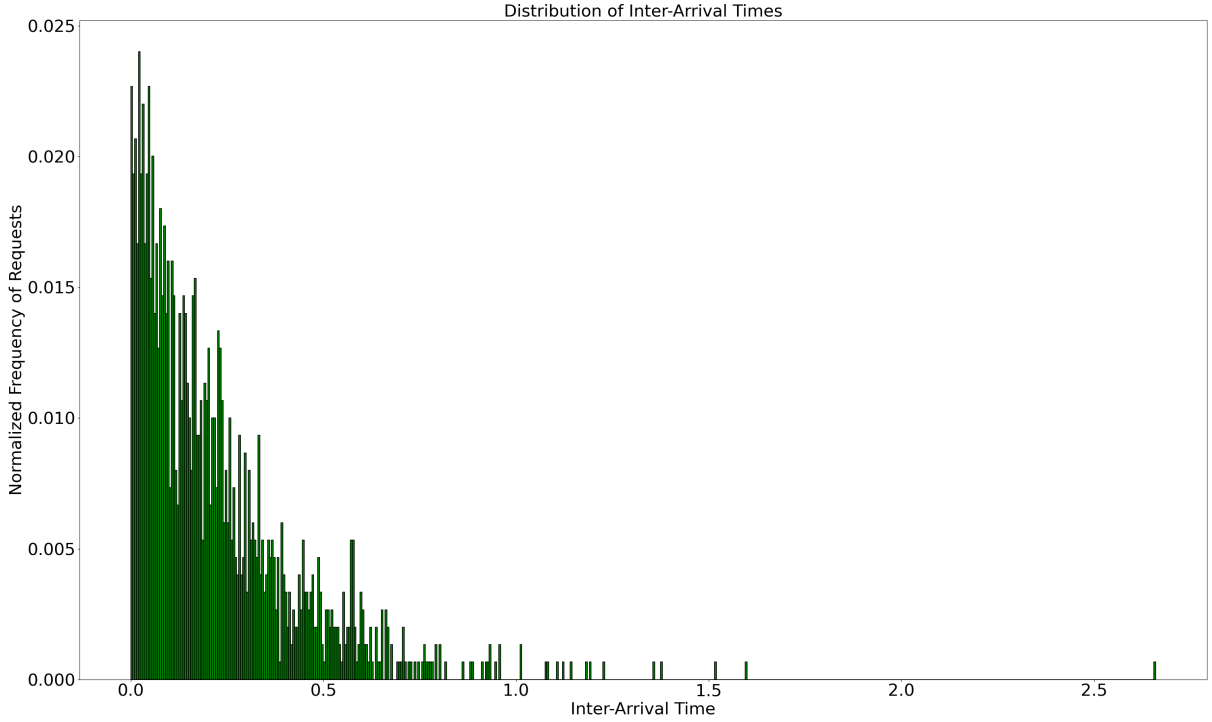

Figure 2: 1a: Distribution of Inter-Arrival Times

Problem 1 continued on next page...

The request length is constant for all requests and does not have a distribution. Inter-arrival times are exponentially distributed with mean = 0.21897906804536368 and standard deviation = 0.2182914272255854.

b) Do the same as above to recover the parameters of distribution number 2. In a similar way as above, collect and post-process the server output data generated by the following parameter:

```
./server_lim -q 1000 2222 & ./client -a 4.5 -s 5 -n 1500 -d 2 2222
```

When decoding the distributions used by the client and their parameters, make conclusive statements about the distribution type for both inter-arrival times and service lenghts, and explicitly state their mean and standard deviation parameters.

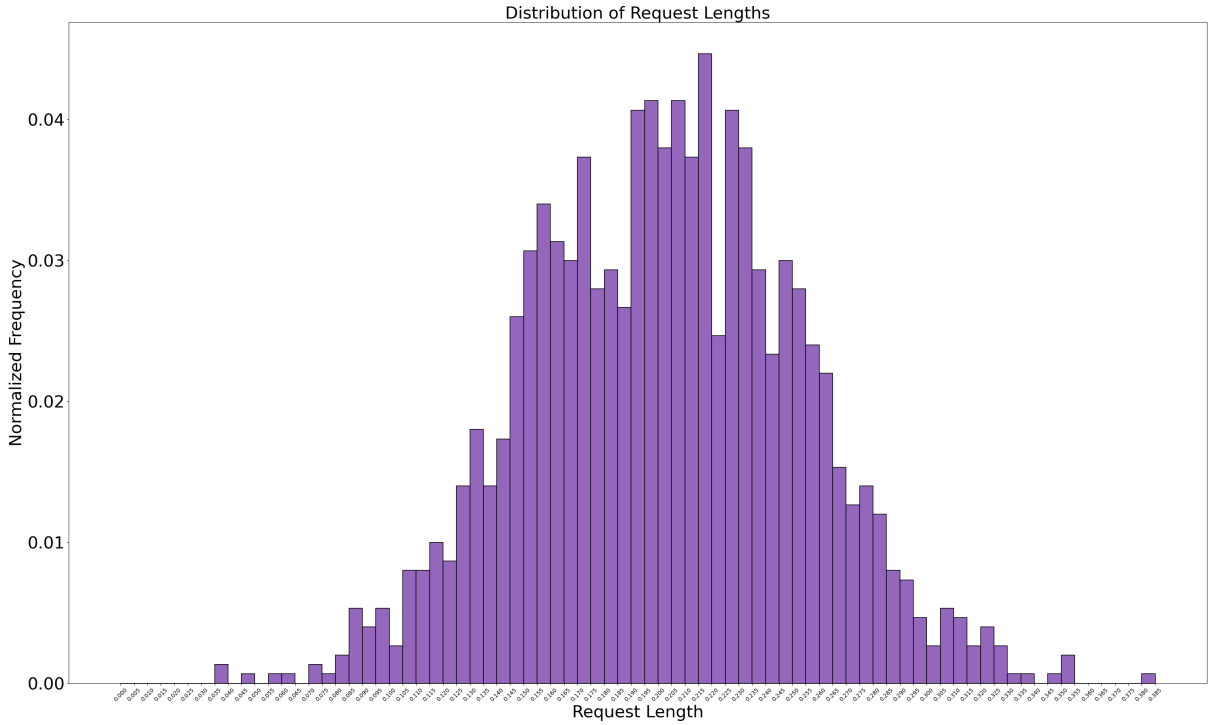

Figure 3: 1b: Distribution of Request Lengths.png

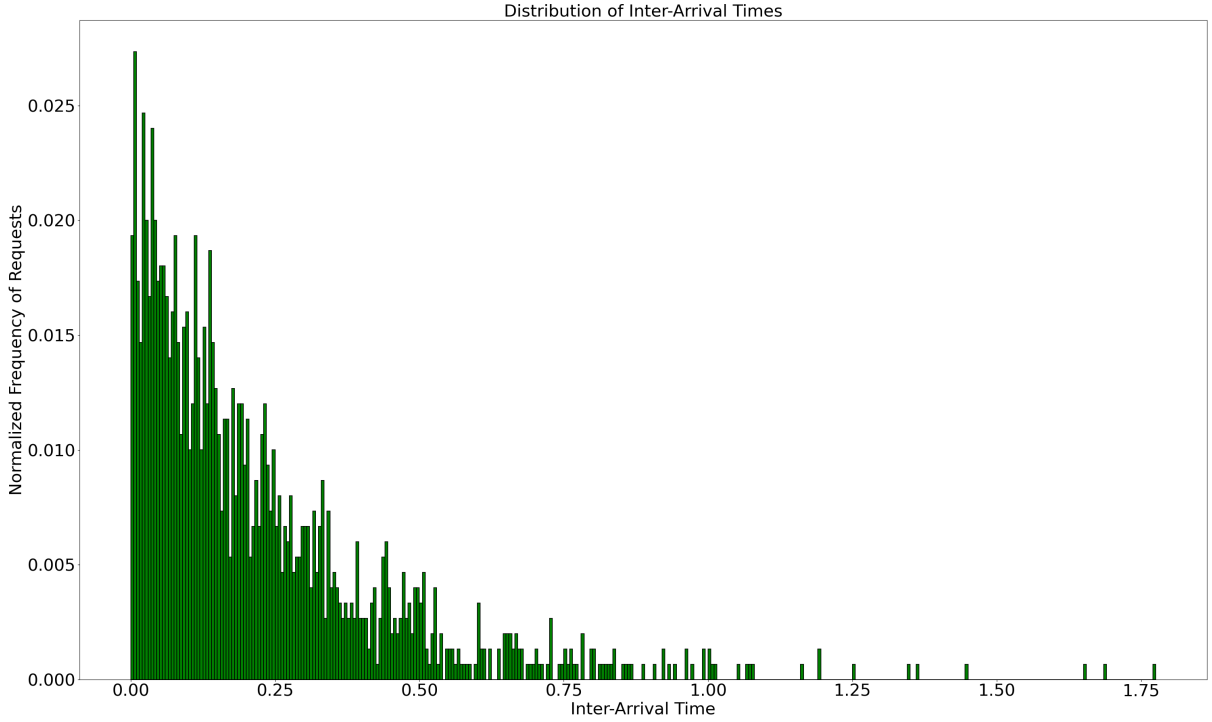

Figure 4: 1b: Distribution of Inter-Arrival Times

Service lengths are normally distributed with mean = 0.20219557866666665 and standard deviation = 0.05041654608061205.
Inter-arrival times are exponentially distributed with mean = 0.21936007138092045 and standard deviation = 0.22219159048170303.

c) We are now ready to see how different distributions affect the quality of service in our simple FIFO server. Let us focus on the comparison between an exponential distribution and whatever distribution 1 is. Run and collect experimental data for the following template command:

```
./server_lim -q 1000 2222 & ./client -a <arr. rate> -s 20 -n 1500 -d 0 2222
```

where `<arr. rate>` is varied from (and including) 10 up until 19. Notice that these experiments will ask the client to use an exponential distribution (`-d 0`). Use this set of experiments to produce a plot of the average response time as a function of the server utilization—in a way similar to what you did in HW1.

Overlap on the same plot produced above the curve obtained by post-processing in the same exact way the results coming from running the following template command:

```
./server_lim -q 1000 2222 & ./client -a <arr. rate> -s 20 -n 1500 -d 1 2222
```

where once again `<arr. rate>` is varied from (and including) 10 up until 19. What can you conclude about the quality service perceived by the user when the load (a.k.a. its utilization) is comparable and only the distribution of the traffic characteristics changes?

Problem 1 continued on next page...

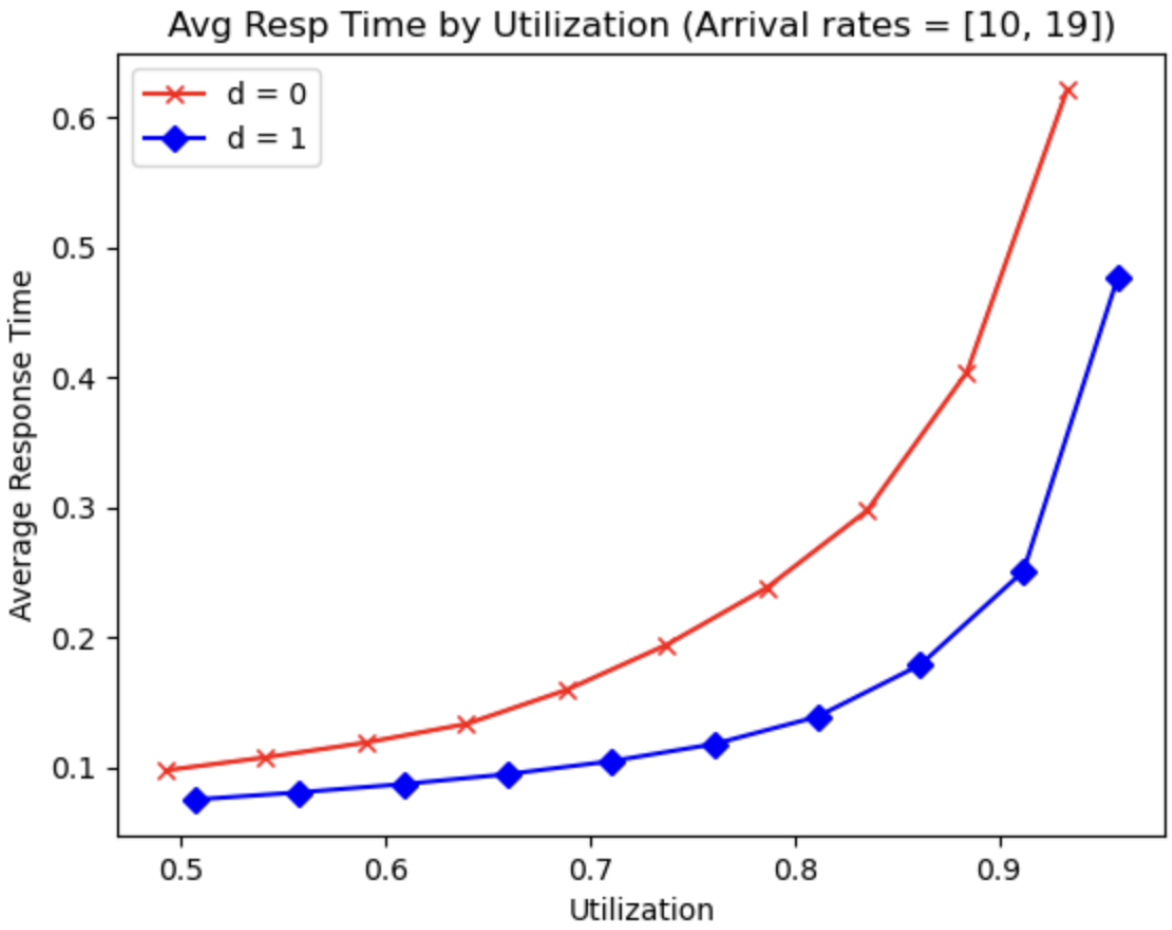

Figure 5: 1c: Average Response Times by Server Utilization.png

The quality of service perceived by the user increases when utilization is comparable but the traffic characteristics (-d) is set to 0. In other words, the traffic conditions of 0 is better than 1 in terms of quality service.

d) Let us now explore what happens when the queue has a constrained size. To do that, run the following command:

```
./server_lim -q 10 2222 & ./client -a 18 -s 20 -n 1500 -d 0 2222
```

Post-process the server output to understand what happened to our requests. First, calculate the ratio of requests that get rejected over the total. Next, plot the distribution of the inter-rejection time, i.e. the time that elapses between a rejection and the next. What does that distribution look like? Do not recover the parameters of the distribution, but simply share your insights on the shape of the distribution.

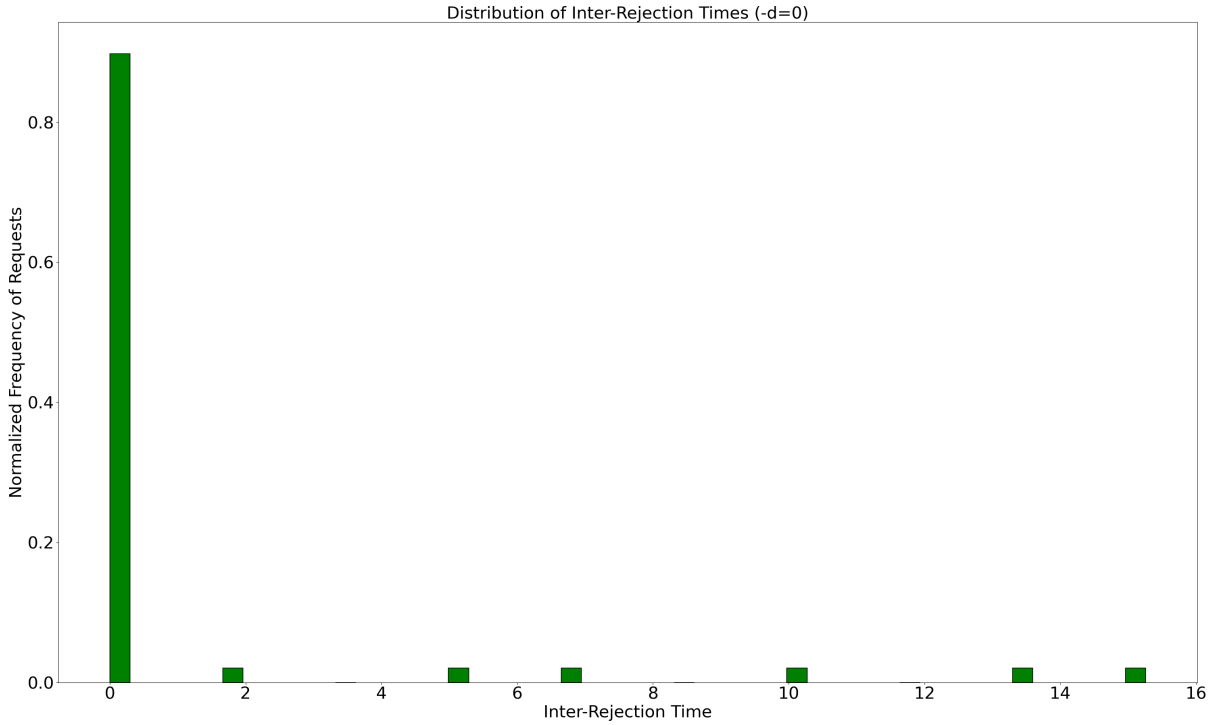

Figure 6: 1d: Distribution of Inter-Rejection Times (-d=0)

The shape suggests that inter-rejection times are exponentially distributed.

e) Repeat the same analysis as above, when distribution number 1 is used instead, thus by running the following command:

```
./server_lim -q 10 2222 & ./client -a 18 -s 20 -n 1500 -d 1 2222
```

If you compare the new rejection rate and shape of the distribution, would you conclude that the new system (the one that uses `-d 2`) offers a better or worse service to its users?

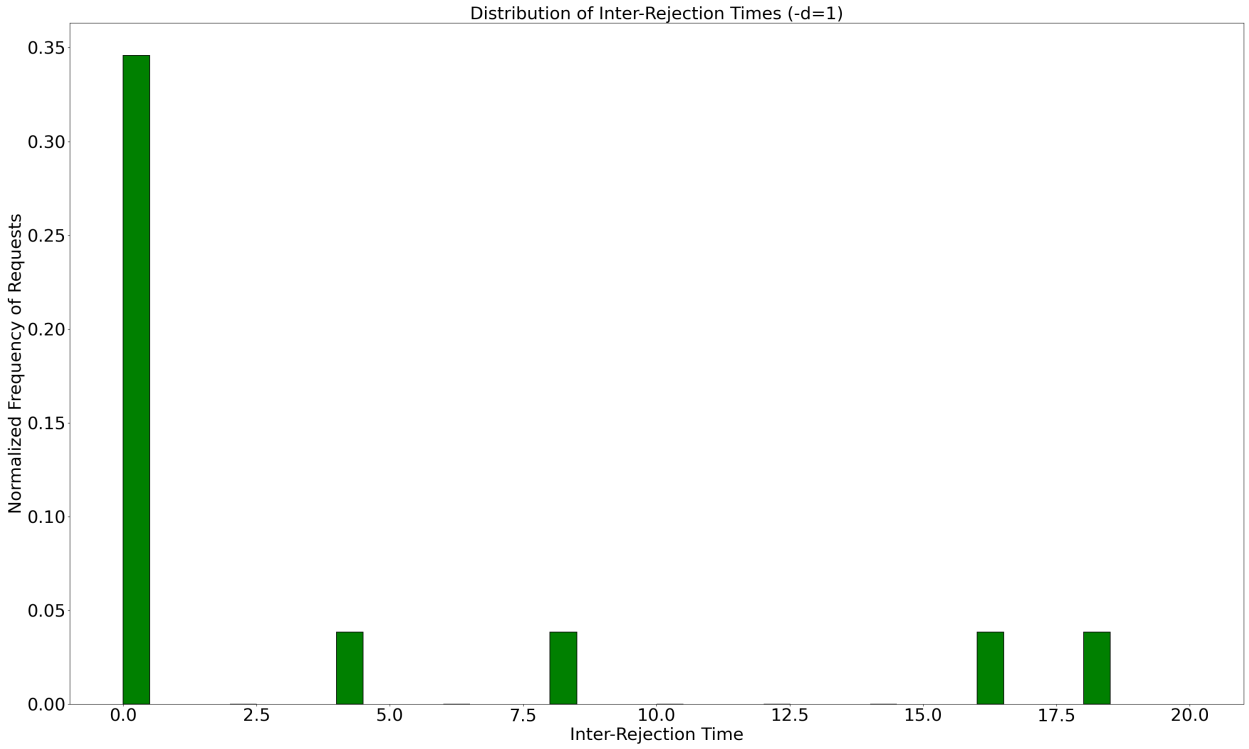

Figure 7: 1e: Distribution of Inter-Rejection Times (-d=1)

The shape suggests that when -d is set to 1, the inter-rejection times also follows exponential distribution. The new system has a generally longer (i.e. more spread out distribution) inter-rejection times. In general, a system with longer inter-request rejection times is better for users compared to a system with shorter inter-rejection times (-d = 0). Longer inter-request rejection times mean that requests are more likely to be accepted and processed by the system. Users send requests to a system with the expectation that their requests will be fulfilled. Moreover, a system with longer inter-request rejection times can handle a higher volume of requests without overwhelming the system.