# Project 2: Queuing

Investigating Scheduling Policies in Queuing Systems

Hope Tsai Sara Kao Lemara Williams

May 24, 2021

## Introduction

Queueing is an inevitable annoyance of modern life. It is so pervasive and inefficient that "Americans spend roughly 37 billion hours each year waiting in line".[1] This drain on time and emotional energy can be attributed to many factors, such as the speed of the processing system or the number of people in line. However, such factors may not be as malleable as one would prefer. For example, one may be hard-pressed to find a cashier who can process one hundred items in ten minutes, and one cannot simply remove people from the queue. The annoyance of queueing cannot be readily alleviated. However, there is one factor that is presupposed into almost every queue: the first-in, first-out scheduling policy. Even though society may deem this policy as "fair," it is not guaranteed to be the most efficient. One might question whether the prevalence of this policy may lead to the prevalence of lines. Perhaps humanity's adherence to fairness is queueing's "fatal flaw."

In this paper, we study the efficiency of four different scheduling policies for one single-server queueing system. The system has a queue that can contain infinite jobs, and a server that can process one job at a time. The first scheduling policy that we study, as described above, removes from the queue whichever job entered first; we refer to this policy as FIFO. We then study two scheduling policies that illuminate the potential benefits of leveraging additional information about the jobs in the queue. In the policy we refer to as minHeap, the system removes from the queue the job that will take the shortest amount of time to complete. In the policy we refer to as maxHeap, the system removes from the queue the job that will take the longest amount of time to complete. The final policy we study, the random policy, removes any of the jobs in the queue with equal likelihood, regardless of the job's size or arrival time.

In order to evaluate efficiency, we use two performance metrics: mean response time and mean number of jobs in the system. A low mean response time indicates that the server is processing jobs quickly. A low mean number of jobs in the system indicates that the server is not being stalled by certain jobs, so the queue of waiting jobs is not growing excessively in length. If both performance metrics are lower for one policy than for another, then we can conclude that it is more efficient on average. We also study the variance of these performance metrics for the different scheduling policies, with the goal of understanding the potential tradeoffs between consistency and average efficiency.

We add further nuance and generalizability to our experiments by utilizing three different probability distributions for the interarrival time: Erlang, Geometric, and Hypergeometric. Within

[1] Stone, A. (2012, August 18). *Why Waiting Is Torture*. The New York Times. https://www.nytimes.com/2012/08/19/opinion/sunday/why-waiting-in-line-is-torture.html?pagewanted=all.

the three probability distributions, Geometric serves as the baseline. The Erlang distribution yields a lower variance than Geometric, and an instance, with parameter $p$, is generated by summing two instances of the Geometric distribution (with parameter $p$). Hypergeometric involves three parameters: $q$, $p1$, and $p2$. It yields a higher variance than Geometric. To generate an instance, essentially, a coin is tossed with probability $q$ of coming up heads. If the coin toss results in heads, then an instance of the Geometric distribution with parameter $p1$ is generated. Otherwise, an instance of the Geometric distribution with parameter $p2$ is generated. By testing our scheduling policies with these three probability distributions, we can better generalize our results to the real world, which encounters interarrival times that vary both in expectation and variance. Ultimately, we aim to discern if a different scheduling policy can make queues more efficient, thereby diminishing a prominent annoyance of modern life.

In the realm of computers (that is, in a world without the notion of fairness), the FIFO and random policies are essentially equivalent. The random policy delays processing certain jobs that would be processed sooner under the FIFO policy, but also processes sooner jobs that would have remained in queue for longer. As a result of these opposing effects, we expect that these two policies will have similar mean response times and mean numbers of jobs in the systems. However, we expect that the variance of the performance metrics will be greater for the random policy than for the FIFO policy. For the latter, jobs are processed in the order in which they arrive, and thus, the number of jobs before an individual job's processing can be calculated, resulting in a higher level of consistency for response times. This is not the case for the former. As the former randomly picks jobs to be processed, the number of jobs before an individual job's processing is itself random and therefore unpredictable. This variance does not affect mean response time or the mean number of jobs in the system because these performance metrics are summary statistics. In order to distinguish between the two policies, we have to look at the distribution of response times and numbers of jobs in the system. We hypothesize that, although the random policy and the FIFO policy will perform the same on average, the random policy will for some jobs perform much better and much worse.

For the minHeap and maxHeap policies, we expect that the former will have the lowest values for mean response time and mean number of jobs in the system, due to the fact that in this policy, small jobs complete very quickly, thus freeing the server and the queue more quickly as well. We expect the maxHeap policy to result in the highest mean response times and mean numbers of jobs in the system. Because large jobs, which receive preference in processing under the maxHeap policy, can stall the server, the queue and therefore response times will grow at a faster rate. As such, we expect that the maxHeap policy will be the least efficient, and the minHeap policy will be the most efficient, out of the four scheduling policies.

Ultimately, we find that fairness does not imply efficiency. As hypothesized, the maxHeap policy is the least efficient, as processing the largest jobs first tends to stall the server. The maxHeap policy also yields a relatively high level of variance. The FIFO and random policies are both more efficient than the maxHeap policy and perform at approximately the same level of efficiency on average, though the random policy has higher variance in performance. Lastly, the minHeap policy is the most efficient for both performance metrics, though with a higher variance in performance than either the FIFO or random policy. Thus, we observe a tradeoff between expected efficiency and consistency.

These results suggest that the presumption of fairness into every queue may be hampering efficiency, for both computerized and human queues. In the realm of computers, applications of this finding are apparent and uncontroversial: Software engineers may simply implement different

scheduling policies. However, this is not true in the human world. For these results to have value in the human world, the desire for faster lines must overpower the desire for fairness, enabling scheduling policies that prioritize some jobs over others.

## Model

### The System

Before we could run analyses, we first implemented a single-server queueing simulation. Only one job can be serviced at a time. When a job arrives and the server is idle, the job automatically enters service. However, when a job arrives and the server is busy, the job enters the queue. When the server is idle and the queue is nonempty, the next job that enters the server depends on the simulation's specified scheduling policy. The queue does not have a size limit, so an unlimited number of jobs can enter the queue.

Below is a graphic representation of the single-server simulation.

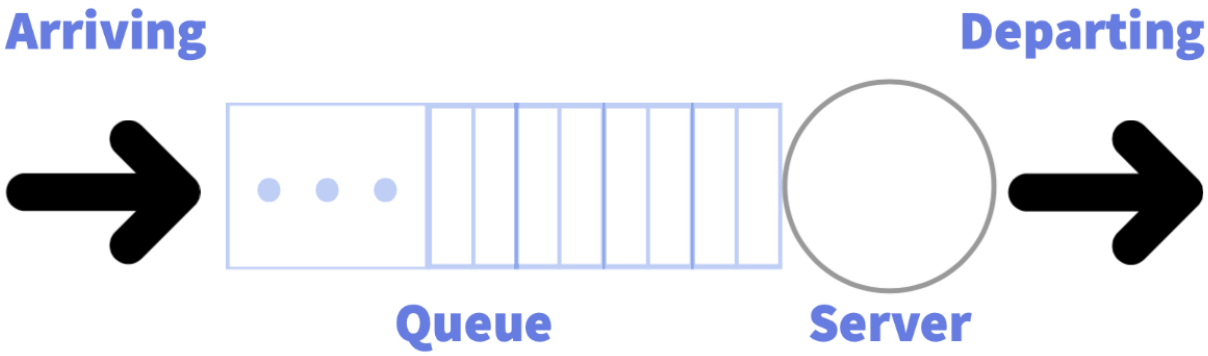



Figure 1: Graphic representation of the single-server simulation

### Assumptions

Throughout our experiments, we assume that the job size is drawn from a Geometric distribution, with parameter $p = 0.5$. For our initial experiments in which we compared the scheduling policies for mean response times and mean numbers of jobs in the system, we assumed that interarrival time was drawn from a Geometric distribution where we varied parameter $p$. To examine the distributions of mean response times and mean numbers of jobs, we assume that interarrival time is drawn from a Geometric distribution with parameter $p = 0.46$. In our later experiments, we examine the scheduling policies for different distributions of interarrival times. In particular,

when we drew interarrival times from the Hypergeometric distribution with parameters $q$, $p1$, and $p2$, we fixed $q = 0.25$ and $p2 = 0.5$ and varied parameter $p1$.

### Experimental Set-up

We had two rounds of experiments. In our first round, we evaluated mean response time. The response time is the difference between the departure time and the arrival time for each job. For our first experiments, we assumed that interarrival time was drawn from a Geometric distribution where we varied the parameter $p$, which effectively varied the expected value of the distribution. For each scheduling policy, we started at $p = 0.0025$ and incremented by 0.0025 until we reached $p = 0.4974$. For each value of the parameter $p$, the program returned the mean response time for the first 100,000 arrivals, discarding the first 2,000 arrivals in order to eliminate noise. Then, in order to examine the distribution of response times for the different scheduling policies, we fixed $p = 0.46$. The program returned the response time for the first 5,000 completed jobs after discarding the first 2,000 arrivals. Finally, we repeated our experiment varying the expected value and finding a range of mean response times with different distributions. For the Erlang distribution, we started with $p = 0.0025$ and incremented by 0.0025 until we reached $p = 1$. For the Hypergeometric distribution, in order to vary the expected value of the distribution, we fixed $q$ and $p2$ and varied $p1$. We started with $p1 = 0.0025$ and incremented bye 0.0025 until we reached $p = 1$.

In our second round, we evaluated the mean number of jobs in the system. The number of jobs in the system is the sum of the number of jobs in the queue and the number of jobs in the server. The process for these experiments was very similar to those evaluating the mean response time. For the experiments in which we varied the expected value of the distribution, for each scheduling policy, the program instead returns a list of the number of jobs in service that corresponded to each $p$ used in the given distribution. Similarly, for our experiments to examine the distribution of the numbers of jobs in the system for different scheduling policies, we fixed $p = 0.46$ and the program returned the the number of jobs in the system for the first 5,000 departures after discarding the first 2,000 arrivals.

### Description of Evaluation Metrics of Results

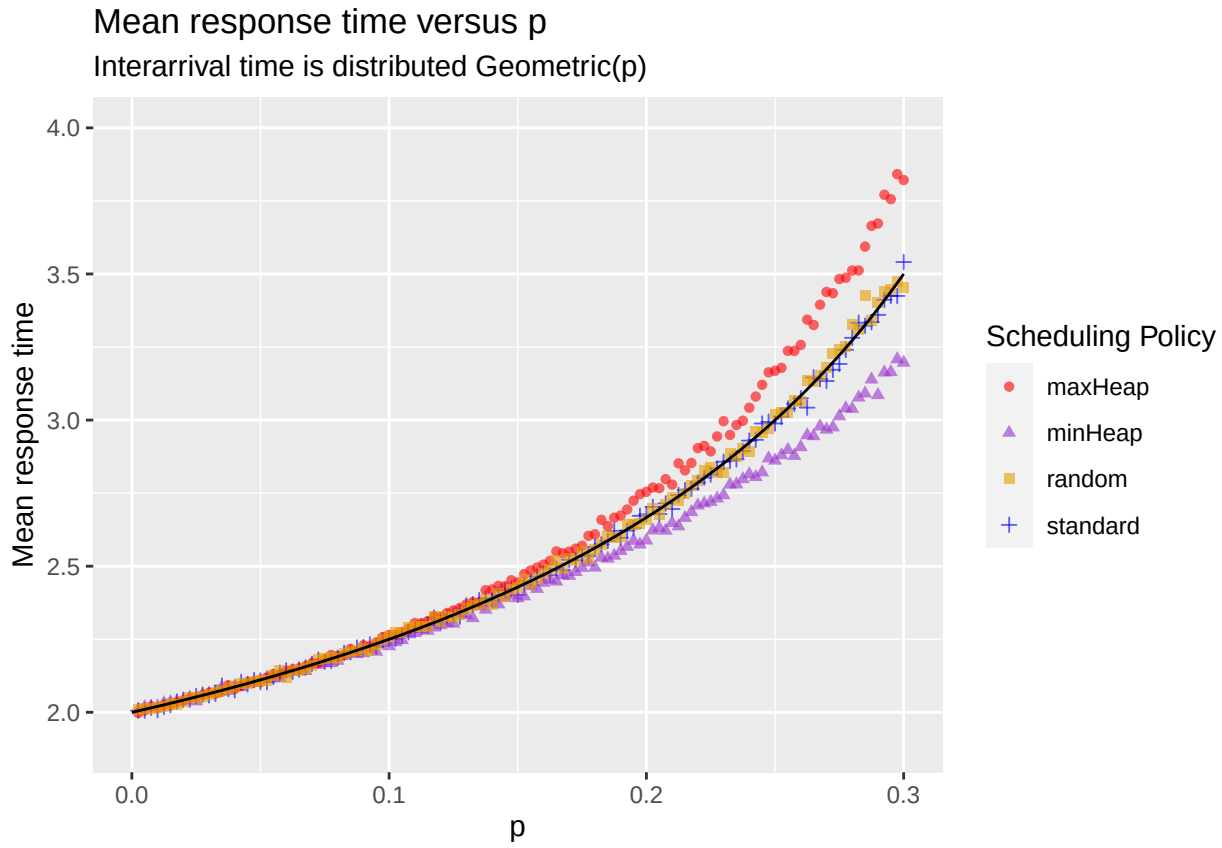
We evaluated our results through the values we obtained for E[N], mean response time, and E[T], the mean number of jobs in the system. Lower mean response times and lower mean numbers of jobs in the system means that a scheduling policy is performing more efficiently on average. A lower variance in the distribution of a performance metric for a scheduling policy means that the corresponding scheduling policy is performing more consistently.

We analyzed our results using a variety of graphical analyses. We first focus on the results of a Geometric distribution since it serves as our baseline. In our first two scatterplots, we graph mean response times and mean numbers of jobs in the system against the parameter $p$ from the Geometric distribution from which the interarrival times were drawn. These graphs enable us to evaluate the average efficiency of the different scheduling policies for a variety of different expected values of interarrival times. For each scheduling policy, we have a histogram of response times and a histogram of the number of jobs in the system where interarrival time is drawn from a Geometric distribution. These graphs enable us to evaluate the relative consistency of each scheduling policy across our performance metrics. In order to generalize our results to interarrival times drawn from different distributions, , we also plotted mean response time and mean number of jobs in the system against the expected value of interarrival time over the Erlang, Geometric, and Hypergeometric

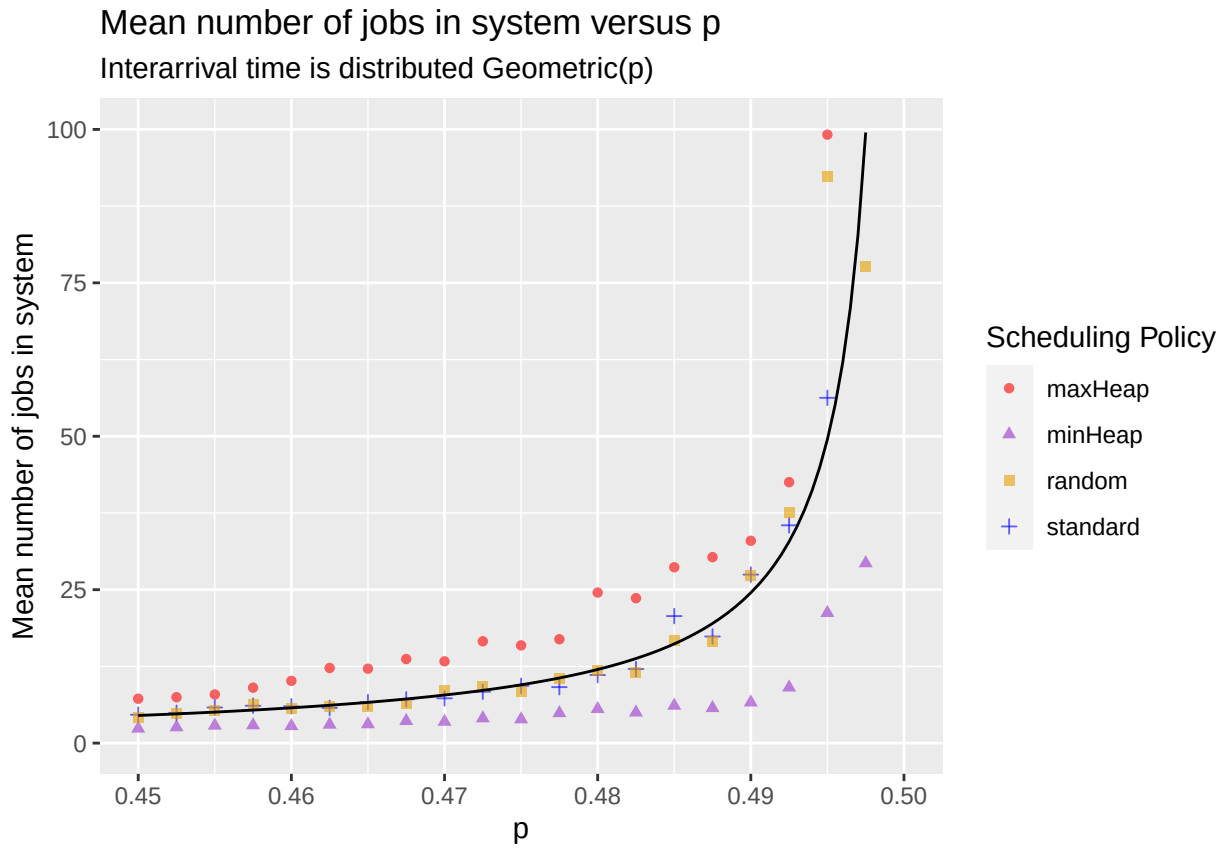
distributions In order to convert the information we had about the parameters in each distribution into expected values, we found that the expected value of a random variable $X \sim \text{Erlang}(p)$ is given by $\text{E}[X] = \frac{2}{p}$; the expected value of a random variable $Y \sim \text{Geometric}(p)$ is given by $\text{E}[Y] = \frac{1}{p}$; and the expected value of a random variable $Z \sim \text{Hypergeometric}(q, p1, p2)$ is given by $\text{E}[Z] = \frac{q}{p1} + \frac{1-q}{p2}$. We calculated the expected value for each mean response time and each mean number of jobs in the system and plotted against the expected value of interarrival time in order to compare our distributions more easily.

## Results and Discussion

```
## Warning: Removed 316 rows containing missing values (geom_point).
```




```
## Warning: Removed 718 rows containing missing values (geom_point).
```




In these graphs, we plot mean response time and mean number of jobs in the system against the parameter $p$, where interarrival time is distributed Geometric($p$), for each scheduling policy. Job size is distributed Geometric($p = 0.5$). Note that for the Geometric distribution with parameter $p$, the expected value of the distribution is $\frac{1}{p}$. Thus, $p$ is inversely related to the expected value of the distribution: As $p$ increases, the expected value decreases. We can map any observation about the mean response time and the mean number of jobs in the system regarding $p$ to an analogous observation about the expected value of interarrival time. For example, we observe that as $p$ approaches 0.5, mean response times and mean numbers of jobs in the system approach infinity; consequently, we can conclude that as $p$ approaches 0.5, the expected value of the interarrival times approaches 2. Since job size is distributed Geometric($p = 0.5$), the expected value of job size is 2, so as the expected value of interarrival time increases to 2, we observe very high response times and very high numbers of jobs in the system, leading to the asymptotic behavior for the all scheduling policies. From this point onward, we make all our observations in terms of expected value rather than in terms of $p$.

We observe that for any expected interarrival time, the minHeap scheduling policy was the most efficient, yielding the lowest mean response times and the lowest mean number of jobs in the system. We attribute this result to the fact that the policy tended to process very small jobs very quickly, leading to a higher proportion of completed jobs with very low response times. Moreover, since many jobs are processed very quickly, the queue tends to clear more quickly, leading to very low mean numbers of jobs in the system.

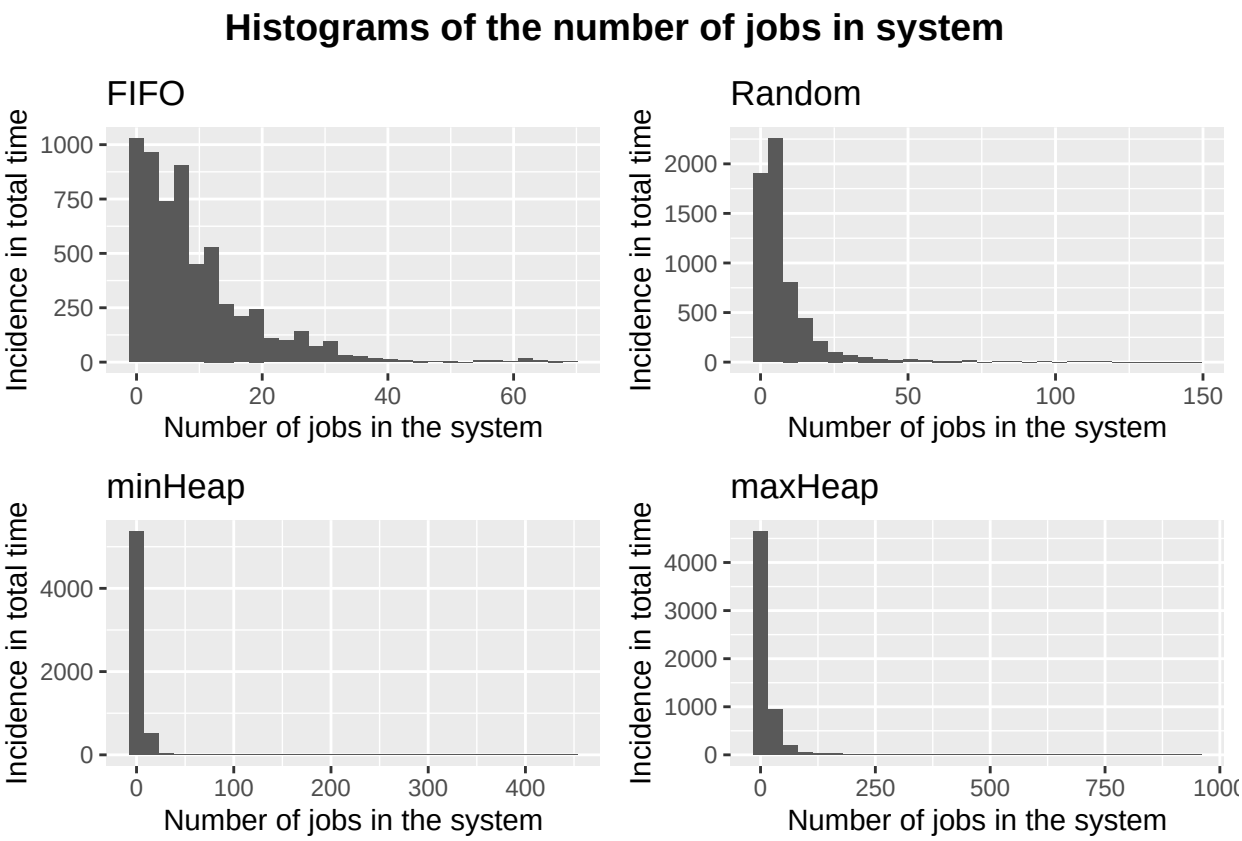
Meanwhile, the FIFO and random scheduling policies performed approximately equivalently for both metrics across all expected interarrival times. Our intuition for this result is as follows:

Although the random scheduling policy leaves jobs in the queue that would have been processed first with the FIFO policy, the randomness also has the potential to process later jobs that would have remained in the queue for a longer amount of time. As such, these opposing effects negate each other, resulting in equivalent expected performance-metric values for both policies.

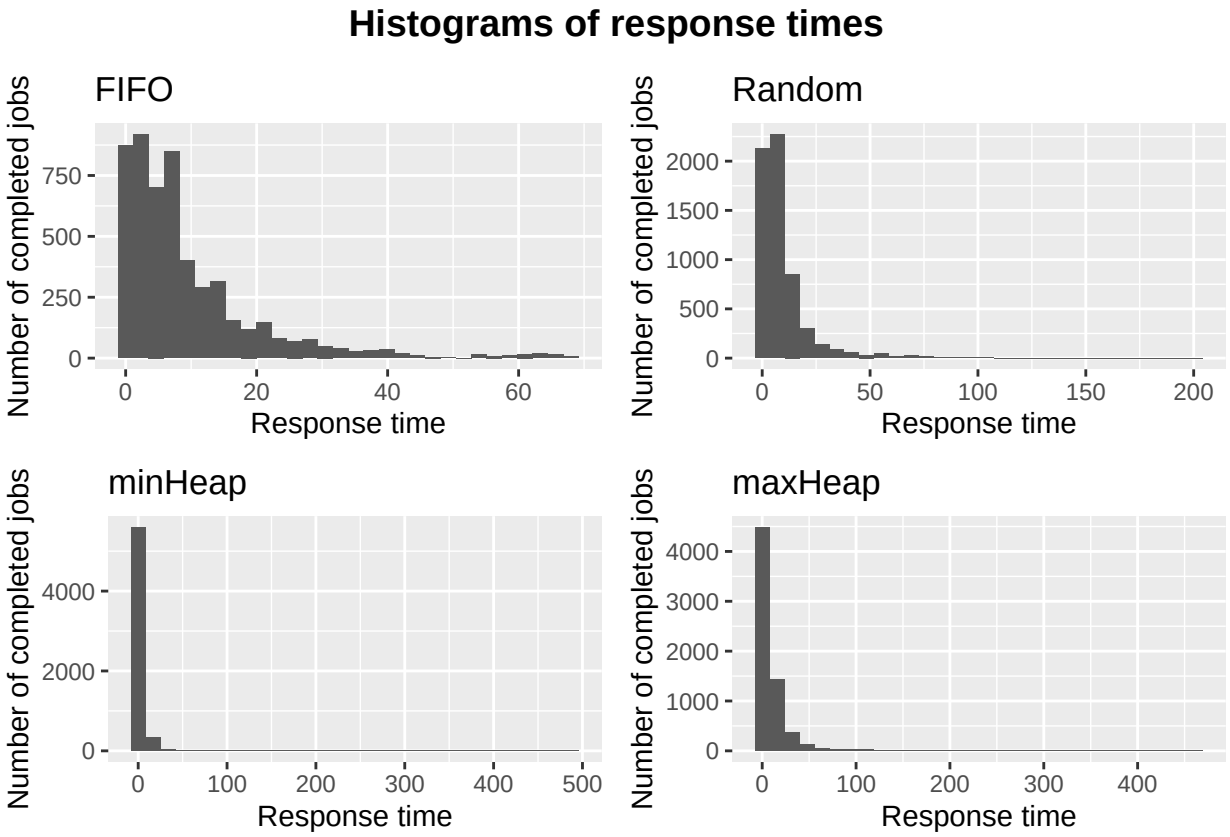
The maxHeap scheduling policy was the least efficient, yielding the highest mean response times and highest mean number of jobs in the system for each expected interarrival time. This result may be due to the fact that the policy always selects larger jobs to be processed, thus stalling the server and allowing other jobs to accumulate in the queue. As the larger jobs take longer to be processed, the jobs in the queue must wait longer than they would have in other scheduling policies.

We note that the inclusion of graphs of both mean response time and mean number of jobs in the system is somewhat redundant because the two performance metrics are related. As response times increase, the queue clears more slowly, resulting in higher numbers of jobs in the system. As the number of jobs in the system increases, jobs remain in the queue for longer, resulting in higher response times. Because these variables move together, we do not gain a great deal of additional information by the inclusion of both. However, we wanted to analyze as many performance metrics as possible to most conclusively determine which scheduling policy was most efficient.

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



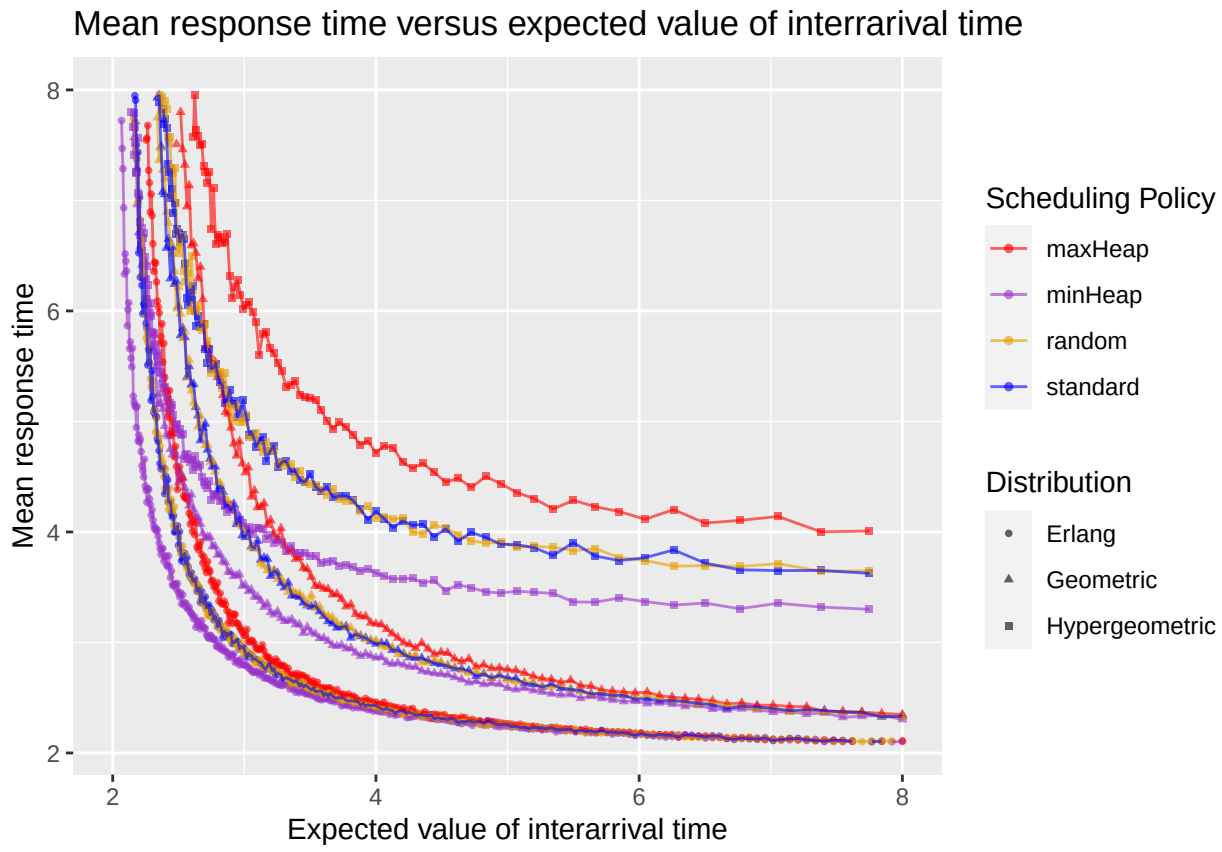

By disaggregating the means we analyzed in the previous section, we can see how the distribution of response times and numbers of jobs in the system was affected by different scheduling policies. Each histogram is for a sample of 5,000 arrivals, where interarrival time was distributed Geometric($p = 0.46$) and job size was distributed Geometric($p = 0.5$).

For the histograms of both performance metrics, the scale on the $x$-axis is the smallest for the FIFO scheduling policy, larger for random, even larger for minHeap, and largest for maxHeap. All four scheduling policies had some high outliers. Given that the Geometric distribution can generate job sizes from 1 to infinity, this result was not surprising. However, the different scales of the $x$-axis suggest that the largest response times were higher for the random scheduling policy than the FIFO scheduling policy, even higher for minHeap, and highest for maxHeap. Thus, the jobs that experienced the longest response times under the random scheduling policy took longer than the jobs that experienced the longest response times under the FIFO scheduling policy, and similarly for minHeap and maxHeap. Moreover, we observe that the spreads of both performance metrics' distributions for the random scheduling policy was much greater than those for the FIFO policy. As the random scheduling policy could process certain jobs more quickly or slowly then they would have been under the FIFO policy, the random policy resulted in many data points that are further from the mean. Thus, although the random and FIFO scheduling policies yielded equivalent, averaged performance metrics, the FIFO was much more consistent than the random scheduling policy.

In addition, the distributions for the minHeap scheduling policy had larger spreads than both the random and the FIFO scheduling policy; the distributions for the maxHeap scheduling policy had even larger spreads than those for minHeap. As the minHeap policy ignores larger jobs in favor of smaller jobs, a large job might remain in the queue for a very long time, resulting in some very high response times and some very high numbers of jobs in the system. In fact, because we terminate the program after a set number of arrivals, it is possible that some large jobs never complete, and we are missing key data points of jobs that would have had even higher response times than we observed. The distributions for the minHeap scheduling policy are heavily skewed right, indicating that most of the jobs experienced very low response times. However, the low mean response times and low numbers of jobs in the system that result from this skewness ignores, to an extent, the larger jobs that take extraordinarily long times to serve, as seen from the high outlier in the histograms. This effect is replicated in the maxHeap policy. However, because the high outliers would tend to be smaller jobs, which might be processed very quickly, the lack of consistency is not made up for in lower mean response times or lower mean number of jobs in the system.

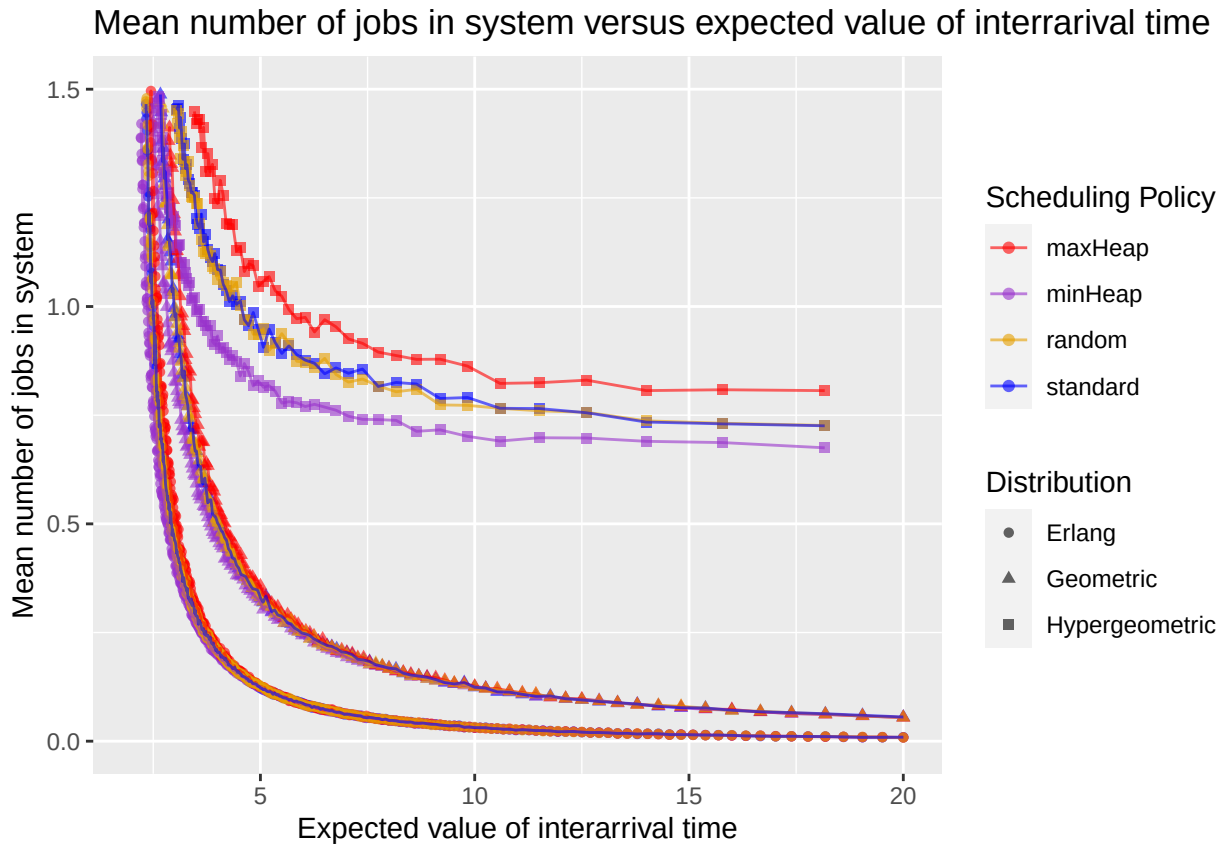
```
## Warning: Removed 1212 rows containing missing values (geom_point).
```

```
## Warning: Removed 1198 row(s) containing missing values (geom_path).
```




```
## Warning: Removed 1194 rows containing missing values (geom_point).
```

```
## Warning: Removed 1192 row(s) containing missing values (geom_path).
```




To generalize our results further, we plot mean response time and mean number of jobs in the system against the expected value of interarrival time over Erlang, Geometric, and Hypergeometric distributions. As established previously, the Erlang distribution yields a lower variance than the Geometric for a given expected value and the Hypergeometric distribution yields a higher variance. We also consider a lower variance distribution, which we refer to, somewhat erroneously, as the Erlang distribution. Note that job size remains distributed Geometric($p = 0.5$).

Most significantly, the relationships we observed in our analysis of the Geometric held across different variances of interarrival times. Within each distribution, for any expected value of interarrival time, the minHeap scheduling policy performed better in terms of both mean response times and mean number of jobs in the system. The FIFO and random scheduling policies performed approximately the same in both metrics, and the maxHeap scheduling policy performed the worst on both metrics for any expected interarrival time. There was no noticeable difference in how the scheduling policies evolved across variance. That is, the same patterns of upward translation hold within each scheduling policy across the Erlang, Geometric, and Hypergeometric distributions. For example, for each distribution, the maxHeap policy generated higher mean response times and mean numbers of jobs in the system for each expected value of interarrival time than the minHeap policy. However, the increase in the performance metrics moving from one distribution to another is not substantially different for the maxHeap relative to the minHeap policy.

These results demonstrate that our initial findings–The minHeap policy is the most efficient, the FIFO and random policies performed approximately equally, and the maxHeap policy is the least efficient–do not depend on the variance of the distribution used to choose interarrival times. We can now conclude with greater certainty that the minHeap policy is more efficient than the FIFO

and random scheduling policies, which are in turn more efficient than the maxHeap scheduling policy.

## Conclusion

We found minHeap to be the most efficient policy on average, both in mean response time and mean number of jobs in the system. For each distribution we examined and for each expected value of interarrival time, the minHeap had the lowest mean response time and lowest mean number of jobs in the system, though it resulted in higher variance than either the FIFO or the random policy. The FIFO and random policies both performed worse than the minHeap policy, but as we predicted, they had similar mean response times and mean numbers of jobs in the system. However, the random policy resulted in higher variance in both performance metrics. MaxHeap performed the worst overall in terms of both average efficiency and consistency. These results show that fairness is not always the most efficient. Although the FIFO and random policies seem more "fair" from a human perspective, they both performed worse than a policy that would not be deemed as the fairest queuing process. However, this surprising result is logical. When in a grocery store, the express lane, which would only checkout customers who had ten items or less in their cart, would be able to check out customers faster than a lane that takes in any random number of items.

Our results are not perfect because they were not made in a perfect environment. Our project had a few limitations. Our examination of different distributions allowed us to conclude that our results generalized for interarrival times drawn from different distributions; however, it only yielded data for three different variances for each expected value of interarrival time. A possible follow-up experiment would involve fixing the expected value of interarrival time and varying the variance, perhaps through creative use of the Hypergeometric distribution. Furthermore, we only used one server with one set speed of processing; the results could have been very different if we examined it from the perspective of a multi-server system.

Another limitation of the project is that we assumed that job size was drawn from a Geometric distribution with parameter $p = 0.5$. Altering the distribution from which job size was drawn or altering the expected value of job size may have impacted the performance of a scheduling policy in a way that our experiments would not have allowed us to observe. Also, the results could be different if we had more information about jobs than just their size. For example, an interesting scheduling policy to study might process first the jobs that are both the smallest in size and the highest in priority. Incorporating more information about the jobs that we are studying would enable us to study a greater variety of scheduling policies.

Finally, there is no guarantee that all the jobs will be processed in our simulation. Because we ran our simulations with limits on the number of jobs that enter the system and because jobs continuously enter the system as long as the simulation runs, now all the jobs had been processed by the time each simulation ended. Both minHeap and maxHeap are biased towards either the smallest or largest job, so some jobs will probably be in the queue forever. For example, under the minHeap policy, if a sufficiently large job entered the queue, it might be the case that smaller jobs continue entering the queue before the large job can be processed. In our simulation, the essentially infinite response time to this large job is never taken into account. Thus, our analysis of the performance metrics for each scheduling policy is somewhat limited; we could only make observations regarding the jobs that do complete.

While these limitations are prevalent and should be accounted for in future experiments, our findings interrogate the ideas of fairness from the human perspective. Realizing the weaknesses of

the FIFO scheduling policy compared to other policies provides the opportunity for a more efficient way of queueing in our everyday lives.