Understanding Job Size Variance and Frequency Division Policies In Queues

Connor Haugh COSC-223 Kyler Kopacz

Introduction

Understanding queues is essential to understanding the performance of computer systems. Multithreading, Server performance, and networking all depend on the efficient processing of jobs. This project explores two fundamental issues in queueing theory. The first is the relationship between the variance of job size and mean time in system. Servers might need to be optimized based on the size of jobs they expect, and exploring variance can lead to helpful guidelines for that. This project asks the question: How does job size variability affect mean response time in a single-server queueing system? The second relies on a two server, two queue system, and explores how the distribution of arrivals to the system affects performance. This part asks the question: in two-server queueing systems, is a policy which places new arrivals. These experiments can be extrapolated into important understandings of networks. By tracing these two threads, this project attempts to uncover valuable insights into queueing theory.

Experimental Setup: Experiment 1

The system we'll study is a M/HyperExponential/1 system. Interarrival times are Exponentially distributed with rate λ, where λ ranges from 0 to 1. Service times (job sizes) are Hyperexponentially distributed with mean size E[S]. That the mean service rate is $\mu = 1/E[S] = 1$, so $\lambda < \mu$ and the system is stable. This experiment tests the variance of HyperExponentialy distributed job sizes (**Var(X)**) on the mean time in system (**E[T]**) across a range of arrival rates (**λ**). A random variable drawn from a HyperExponential distribution (X~Hexp(*p*)) comes from either an exponential distribution of rate μ_1 with probability *p* or an exponential distribution of rate μ_2 with probability *1-p*. In order to calculate **Var(X)** in terms of μ_1, μ_2, and *p*, we made several assumptions about our system:

a. The system has *Balanced Means,* or:
$$\frac{p}{\mu 1} = \frac{1-p}{\mu 2}$$
b. The Expected Value of the Job Size is 1
c. The Variance we test on is 1, 10, 20, and 50.

Using these assumptions, one can calculate the necessary parameters μ_1, μ_2, and *p,* in order to test the system. In order to solve for these parameters, use E[X] and the Balanced Means Equation to solve for *p* in terms of the two rates, and then put the Var(X) equation in terms of *p.*

We know the expected value of a HyperExponential distribution E[X] by conditioning on the probability a value is drawn from one of the two exponential distributions. We find $E[X^2]$ by the same logic. Using both those equations, Var(X) becomes apparent.

$E[X] = 1 = \frac{p}{\mu 1} + \frac{1-p}{\mu 2}$, $E[X^2] = \frac{2p}{(\mu 1)^2} + \frac{2(1-p)}{(\mu 2)^2}$, and $Var(X) = E[X^2] - 1$

Substituting in the *Balanced Means* equation for $\frac{p}{\mu 1}$ or $\frac{1-p}{\mu 2}$ into the equation of E[X], solves μ_1 and μ_2 in terms of *p* respectively.

$2p = \mu 1$

$2(1-p) = \mu 2$

Finally, by substitution of u1 and u2 into the Var(X) equation, we find the equation in terms of p. We then graph this to get the following table of parameters.

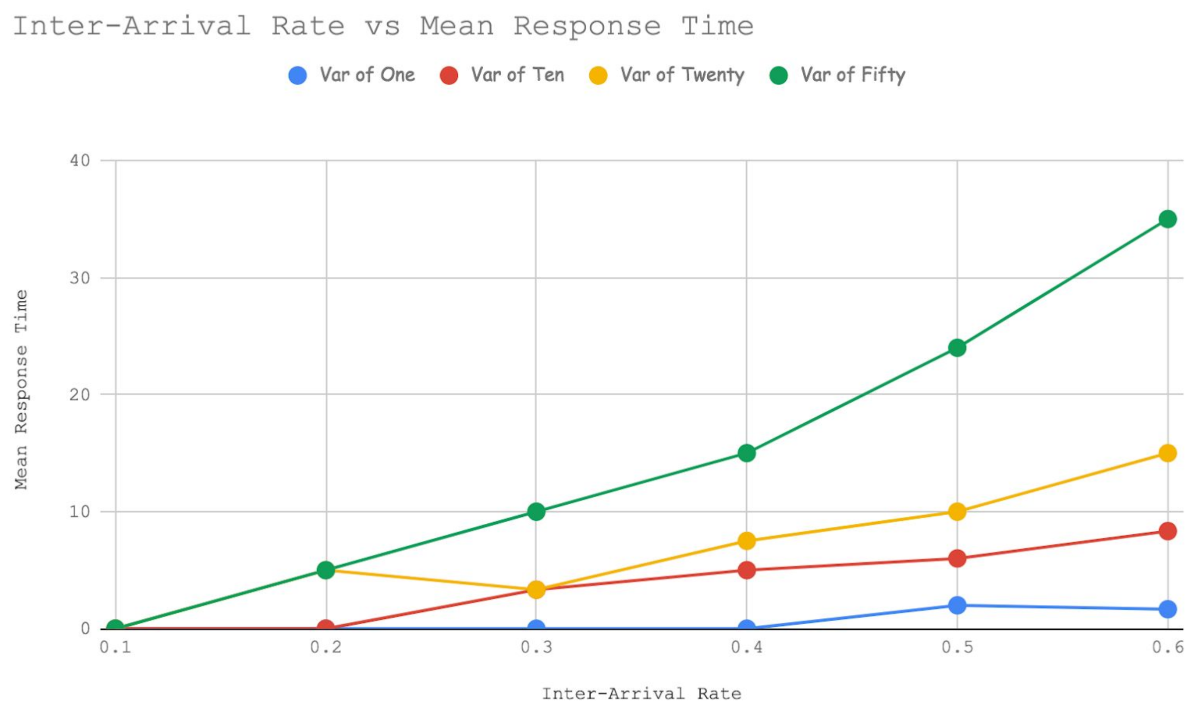
Var(X) = $\frac{1}{2p} + \frac{1}{2(1-p)} - 1$ Gives:

Var(X)	1	10	20	50
p	0.5	0.952	0.976	0.99
µ1	1	1.904	1.952	0.98
µ2	1	0.096	0.048	0.02

Based on these parameters, we establish a M/HyperExponential/1 server simulation by viewing only arrival and departure times. The system then simulates 10^6 arrivals, the first 10^4 of which are discarded. After that point, however, the system takes a snapshot of the number of jobs in the system at a given time step. These are then averaged over the number of snapshots, in order to determine the average number of jobs in the system E[S].

Results: Experiment 1

By Tracking the mean number of Jobs in the system using PASTA theory, we were able to determine the Mean Response Time E[T] using Little's Law. (The Average Number of Jobs in System divided by the Arrival Rate is equivalent to the the Mean Response Time)



Graph 1: The mean response time is directly exponentially proportional to the inter-arrival rate across all variances of job size. Those with smaller variance of job size, however, experienced dramatically reduced response times.

Discussion: Experiment 1

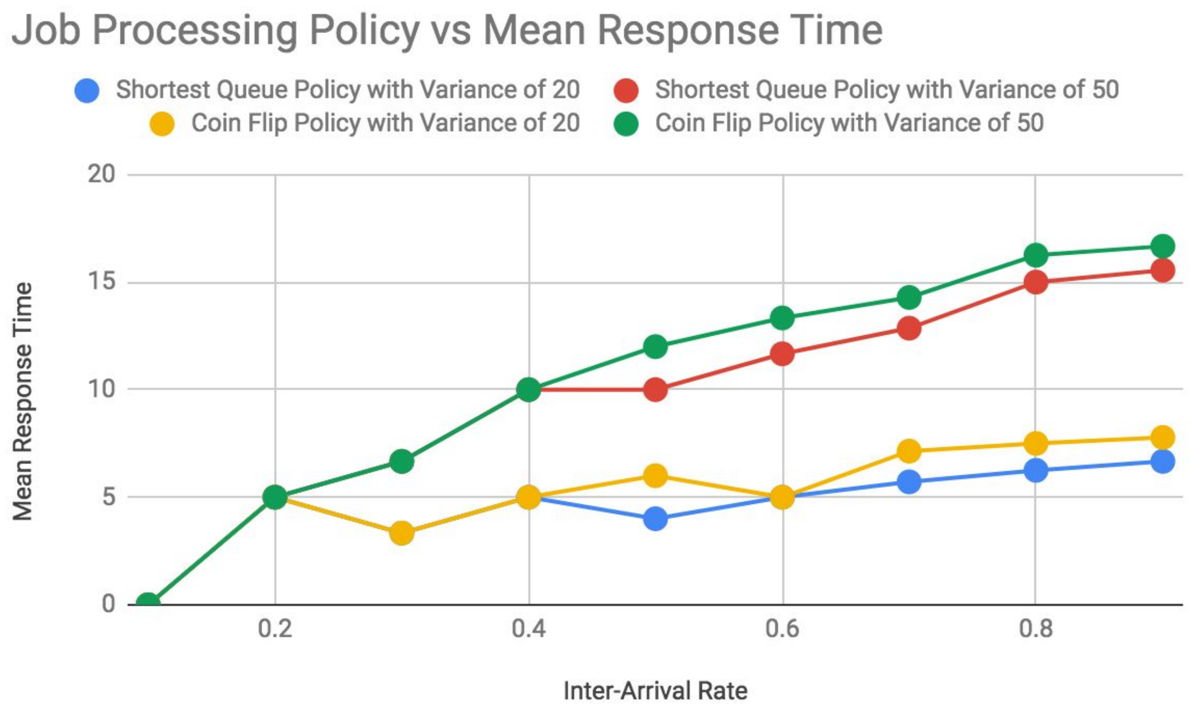
The results of this simulation demonstrate two things. First, the interarrival rate is proportional to the response time. Across every variance tested, as interarrival rate increases, the response time increases. This is quite obvious – if a server gets requests less often, jobs will spend less time in the Queue. This should not be a mind-blowing result, but it confirms the validity of our server simulation. As the frequency of arrivals increases, more jobs arrive in the queue per unit time, and therefore, those jobs must wait to be processed for longer periods of time.

The second fact the results show is that the variance of job size is directly proportional to the mean response time. The parameters which allow for greater variance in the HyperExponential distribution clearly result in greater mean time in system. It follows logically that increasing the size of unlikely large job sizes should somewhat increase the response time, although the interarrival rate should have some effect. The work in class to determine mean response time in continuous time of M/M/1 systems which showed $E[T] = \frac{1}{\mu - \lambda}$, however, that result obviously does not directly correspond to the M/HyperExponential/1 system of our simulation. The result that greater variance in the job sizes, which demonstrates a greater breadth of job size values, cause increased mean response times makes sense. Systems which see larger jobs more frequently will spend more time processing those jobs, therefore impacting the number of jobs in the system and the mean response time.

Experimental Setup: Experiment 2

This experiment attempts to understand frequency division in a M/HyperExponential/2 system. A system of this type has two queues, and arrivals are placed into one of the two queues. The point of interest is policy by which the system chooses which queue to place an arrival in. The first policy tested is the “Coin Flip” policy, in which with probability *p* the queue the arrival is placed is the first, and with probability 1-*p* the second queue. Another policy is the “Shortest Queue” policy, in which the arrival is added to the queue of the smallest length. In the event both queues are the same length, the policy looks at the next departure time of both queues, and places the arrival in the queue with the next departure. The parameter of arrival remains exponentially distributed, while two variances of job sizes, 20 and 50 were used to confirm that the new policy performs better across multiple variances.

Results: Experiment 2



Graph 2: The policy by which the job is placed into a Queue in a M/HypherExponential/2 system has effect on the mean response time. Across multiple variances of job size, and therefore multiple enumerations of job size parameters, the Shortest Queue policy consistently produced faster mean response times times than the Coin Flip policy as inter arrival rates increased.

Discussion: Experiment 2

In the experiment tested, the created frequency division policy division performed better than the coin flip. Across multiple variances of job size, as interarrival rate increased, the "Shortest Queue" policy outperformed the "coin flip policy. It did not matter for small values of λ because in those instances the Queue did not have enough jobs in it at any point for the "evenness" of the two to matter. At heavier loads, however, the shortest queue policy and the coin flip policy exhibited similar response times to their single-server peers with respect to variance, demonstrating the validity of the simulation through congruence in those results.

Conclusion

In order to effectively evaluate the results of the above experiments, first it is important to consider the assumptions the system models made about queueing which might not hold in practice.

In the first system model, the design used seems to be a fairly adequate abstraction of a scenario of a single-server queue. One assumption to note, however, is the relationship between exponentially distributed interarrival rates and the realistic flow of arrivals to a server. Although we modeled the arrival rates across different load sizes, arrivals will not always be exponentially distributed. For example, if a system is processing requests from a video game, which updates every second, jobs arrive in linear fashion. Alternatively, a server used to process human resource requests might be inversely exponential, as once you look at the number of sick days you have once, you are less likely to need that information for a while. To better account for this assumption, perhaps different distributions, like heavy-tailed, linear, or inversely exponential distributions could be tested. These would alter the relationship between mean response time and variance of job size, as they could provided different expected response times, and prevent the use of a simple formulation of Little's law. The other issue with the M/HyperExponential/1 system model stems from its use of the relationship of *Balanced Means* rates of the hyperexponential. Perhaps greater regularity, instead of disparity between $\mu 1$ and $\mu 2$, in the form of a balanced variance relationship of

$$\frac{2p}{\mu 1^2} - \frac{2p^2}{\mu 1^2} = \frac{2(1-p)}{\mu 2^2} - \frac{2(1-p)^2}{\mu 2^2}$$

could contribute to balance p $\mu 1$ *and* $\mu 2$ values which give in the current setup give varied distribution rates. This could lead to a model closer in relationship to what the class learned about M/M/1 systems.

The second system model relies on several assumptions in order to extrapolate itself to real life situations. The first is that servers share identical processing times, and it also only tests across two-server, two queue systems. Testing the model on more servers and queues would dramatically improve mean response time, but the limiting reagent might not be the frequency division policy, as the system with more servers will require shorter queues given similar arrival rates.

All questions about these queueing systems, however, were not fully elucidated in this project. The aim of the second experiment was to find *a* policy which provided improvement over a coin flip. The "Shortest Queue" policy demonstrated a faster mean response time, but only marginally so. A question which remains unanswered, therefore, is what policy is best. Perhaps some algorithm could forsee future timesteps, and make a judgement as to what queue could enter. Another question that remains in processing on multiple requests remains how to deal with varied service times between the two servers. How can a queue selection policy best handled varied service rates? Perhaps the next step would be to move the simulation from the abstract, and into the real. By monitoring the performance of the Romulus or Remus server cluster, one could test across variance in job size and other factors by simply finding matching patterns within the real system process.