1.Queuing Theory

1. Average Arrival rate = 3 per min; Average Service rate = 10/3 per min M/M/1 Queue

p = 3 / (10/3) = .9

Average length of queue = (.9 ^ 2)/(1-.9) = 8.1 people
Average time in queue = .9/(10/3 – 3) = 2.7 min or 162 seconds

2. Average Arrival rate = 40 per min; Average Service rate = 20 per min; Servers: 3 M/M/N Queue

N = 3

p = 40/20 = 2

p/N = 2/3 < 1

Probability of zero users = 1/(1 + 2 + 2 + 8/(6*(1-2/3))) = 1/9

Average length of queue = (((1/9) * 2^4)/18) * [1/((1-(2/3))^2)] = 8/9 people
Average time in queue = (2 + 8/9)/40 – 1/20 = .022 minutes or 1.333 seconds

5 > 3

Probability of having 5 people in system = (2^5 * 1/9)/(3^2 * 3!) = .065844 or 6.58%

3. Average Arrival rate = 9 per min: Exact Service rate = 10 per min M/D/1 Queue

p = 9/10 = .9

Average length of queue = (.9^2)/(2(1-.9)) = 4.05 people

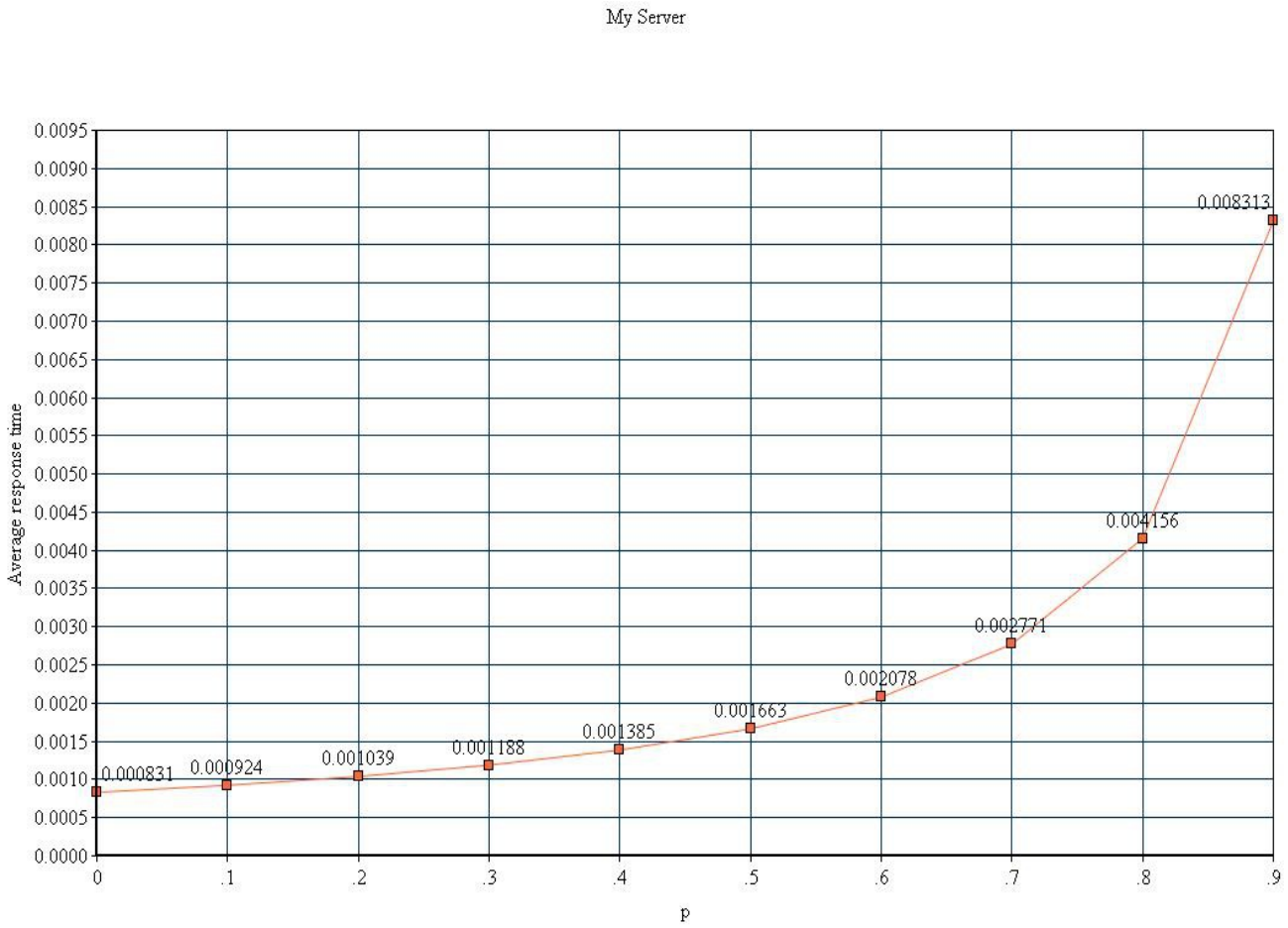
Average time spent in system = (1/(2*10))*((2-.9)/(1-.9)) = .55 min or 33 seconds

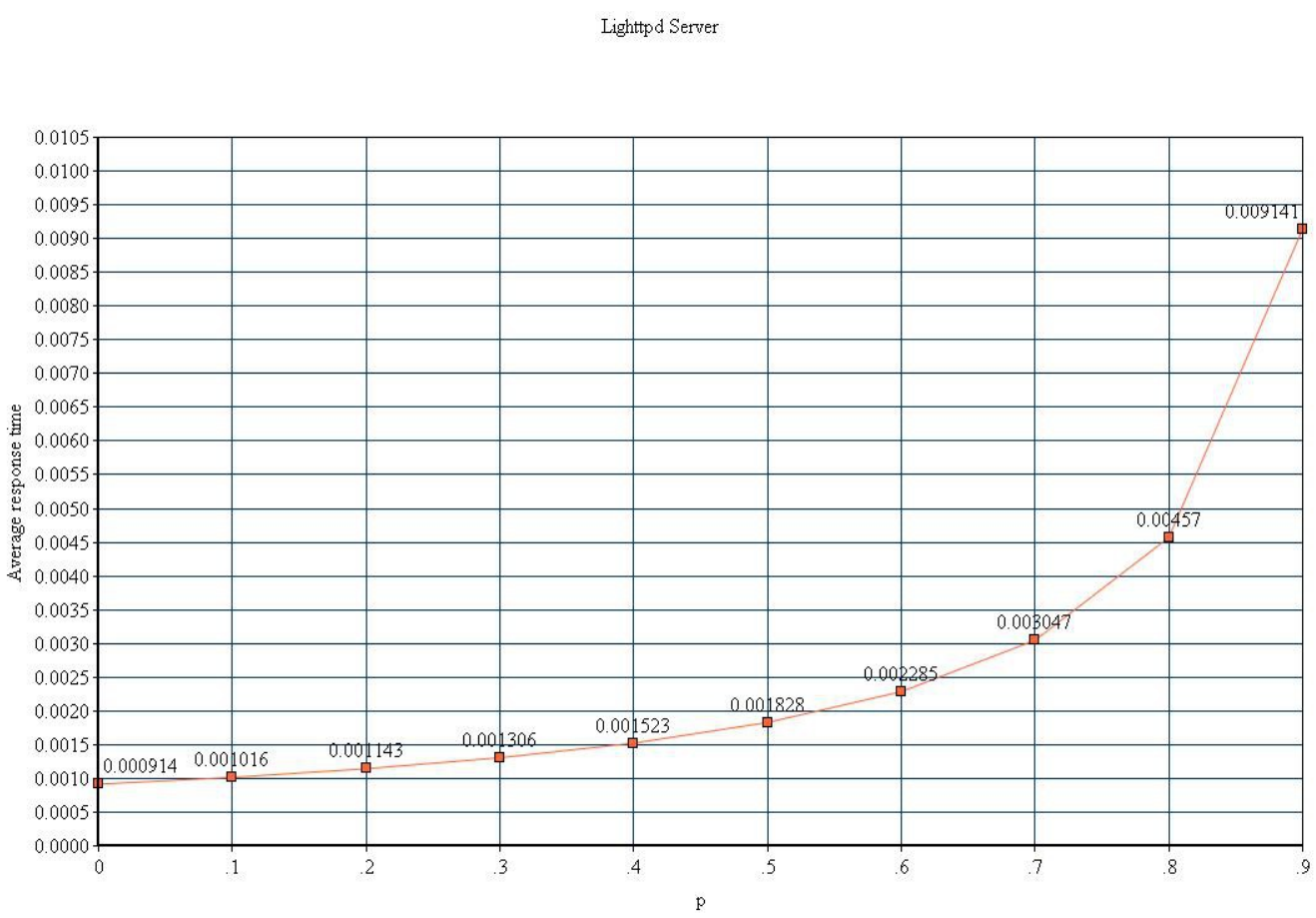
2. Queueing Analysis

Lighttpd μ: 1 request per 0.000914032936096 seconds or about 1094 requests/sec
My Server μ: 1 request per 0.000830649137497 seconds or about 1203 requests/sec

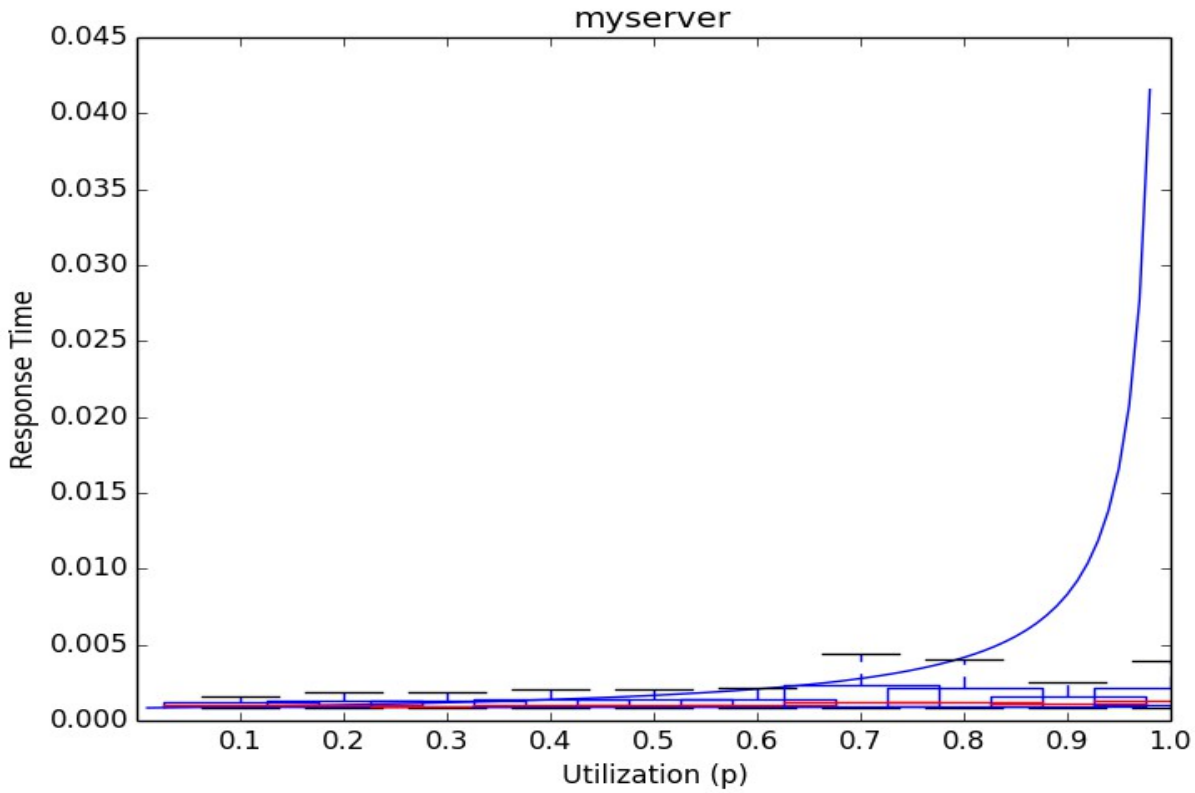
Average time spent in system for M/M/1 Queue: $t = 1/(\mu - \mu p)$

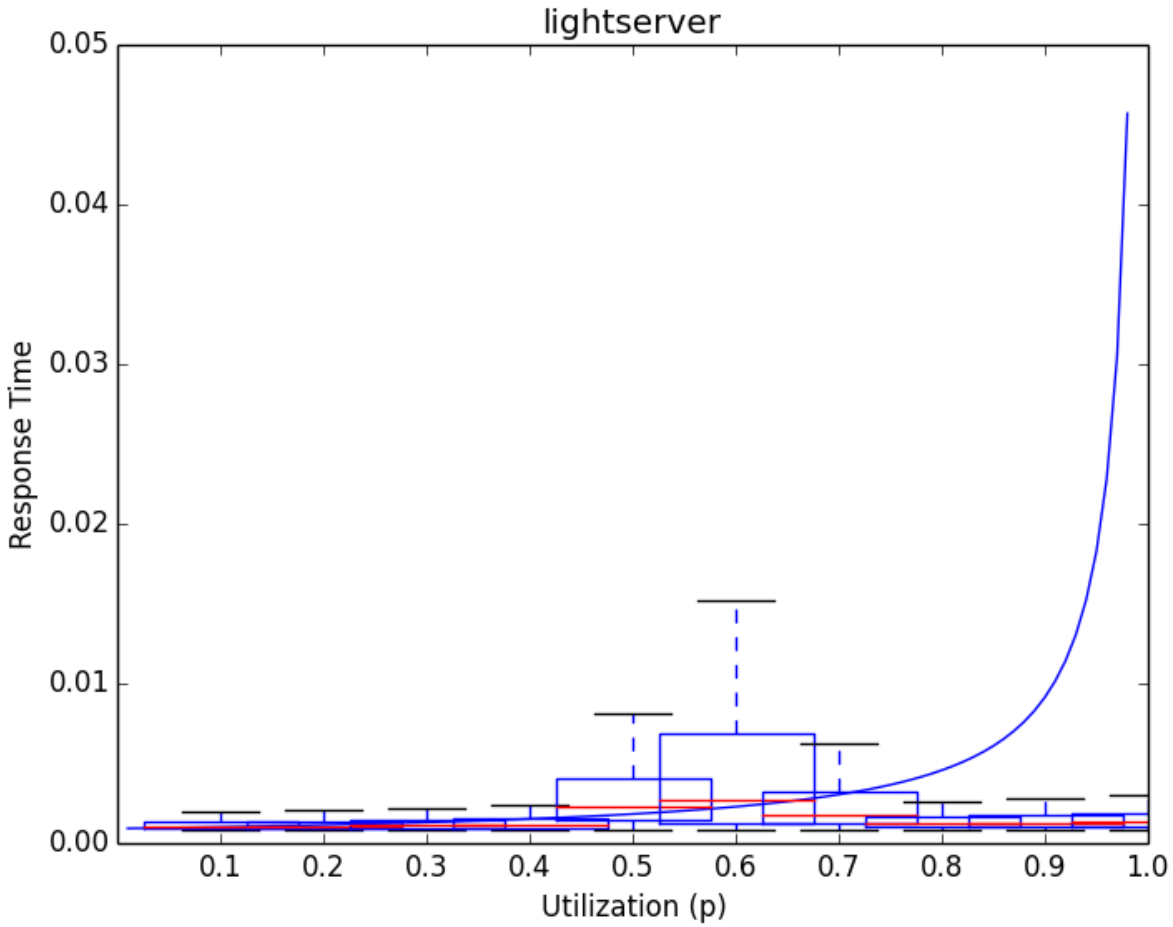
In order to calculate my average service rate for my server we wrote a script which found the time before a request was sent to a server and the time after a response was received and then found the difference. This difference was added to a list and the process was repeated 100 times. We then took the sum of the list and divided it by the length of the list in order to find the average time it took a server to get a request, process it, and send a response. We then took this average and converted it into requests per second. From the slides the average time spent in a M/M/1 queue is 1/(μ- λ). Putting this equation in terms of p we get 1/(μ – μp). Two graphs show my server compared to lighttpd server. The average response time for my server will be consistently smaller than lighttpd because mine can handle more requests per second than light can.





3. Performance Evaluation





We picked 10 utilization points (0.1, 0.2, 0.3 ...1.0) and calculated the appropriate load for my server given its average service rate (1203 clients/sec) For each of these ten loads we then ran generator.py for 30 seconds generating the loads number of clients for the server to handle. The output was saved in a file with the load number in the name of the file. We repeated this process for the lighttpd server. We then modified the plot.py script from Lab 3 to parse all of the files associated with a given server and plot a graph of response time over utilization. On the same graph we plotted what the theoretical response time should look like. Excluding outliers my graphs partially followed the theoretical graph. As the utilization approached 1 however, the expected response time was much higher than the actual response time. There didn't seem to be any really noticeable differences between the response times of my server and those of the lightserver

4. Conclusion

From this lab we learned that servers will crash if they are given requests at a faster rate then they can respond to them. As the average rate of arrival approaches the average rate of service, theoretically the response time should exponentially rise to an unmanageable level. A possible reason for why our servers did not slow down as utilization approached 1 could be that our methods of calculating our average response time were not accurate. The average rate of service might have been much higher than predicted. To explore what the real rate of service might be, we could continuously increase our load until the server response times began to resemble an exponential curve up. We would then have a rough estimate that the average service rate is close to the current load.