Effect of changing service Distribution on Performance:

Changing Service Distribution: Previously, we used parameters (generated from profiling, generated from actual testing; henceforth called **Paramter1**) to calculate the average latency for different configuration. But, now we try to randomize the basic service time for any configuration using different distribution centered about **Parameter1**, and try to compare results.

The following results is generated when I used the Erlang of the **Parameter1** to calculate the basic service time, and proceeded as previous. That is the only change made. This is simulated for inter-arrival rate of 70,where each time frame consists of 5 requests, learning rate =0.5.

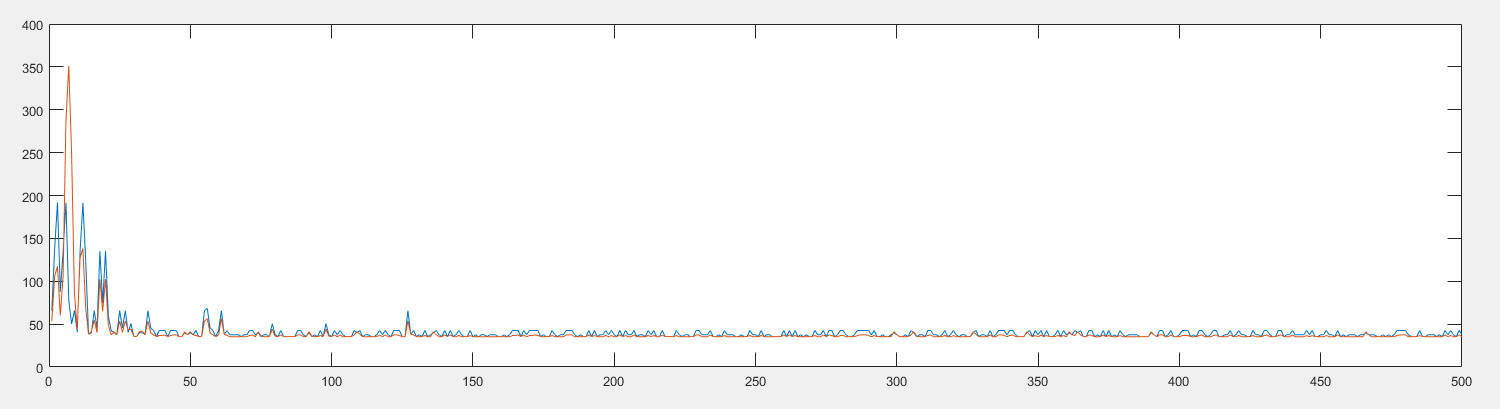


Fig 1. Erlang Change

We can see that RL algorithm is better than SC in this case. But, under further investigation, I found out that there is no convergence in this setting. The configurations selected do result in less average time as compared to the beginning of the simulation as can be seen in the figure. I have supplied the same basic service time for SC as well. For SC, for the current inter-arrival rate it was executed in a fixed configuration, R8T1, where all 8 threads are used to process a single request.

In essence, for RL implementation, the program switches between different configuration and tries to find the best one. But, since we have used Erlang distribution, which produces random numbers (it doesn’t always give smallest value at a some fixed configuration), it didn’t converge to any one configuration, but is changing between configurations that produces small average latency. For SC, we have a fixed configuration and the average latency is influenced by the Erlang distribution. The random numbers produced by Erlang distribution is same for corresponding requests in both implementation.

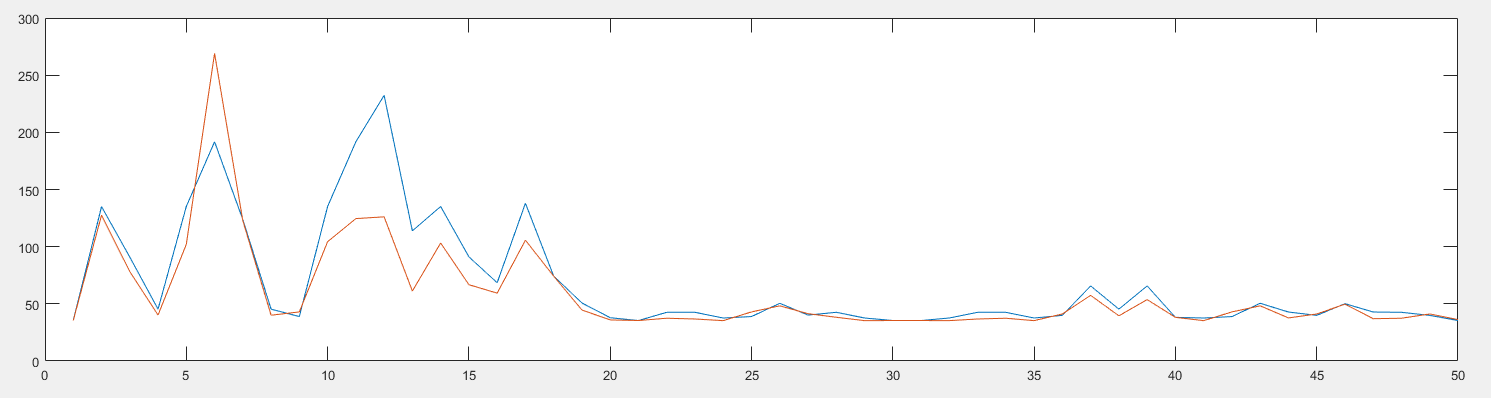


Fig 2. Erlang (50 repetition)

This figure is the result when there are only 50 time frames for inter-arrival rate of 70.

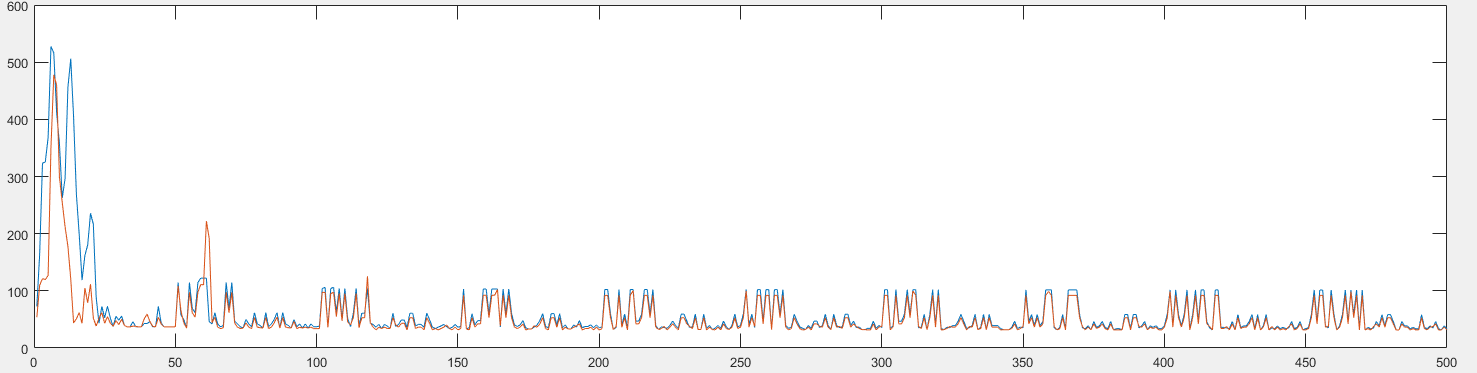


Fig 3. Erlang Distributed Service Time

Above figure is the result of processing 10 different inter-arrival rates repeated for 50 time frames all at once. We can see here as well the RL is better than SC .

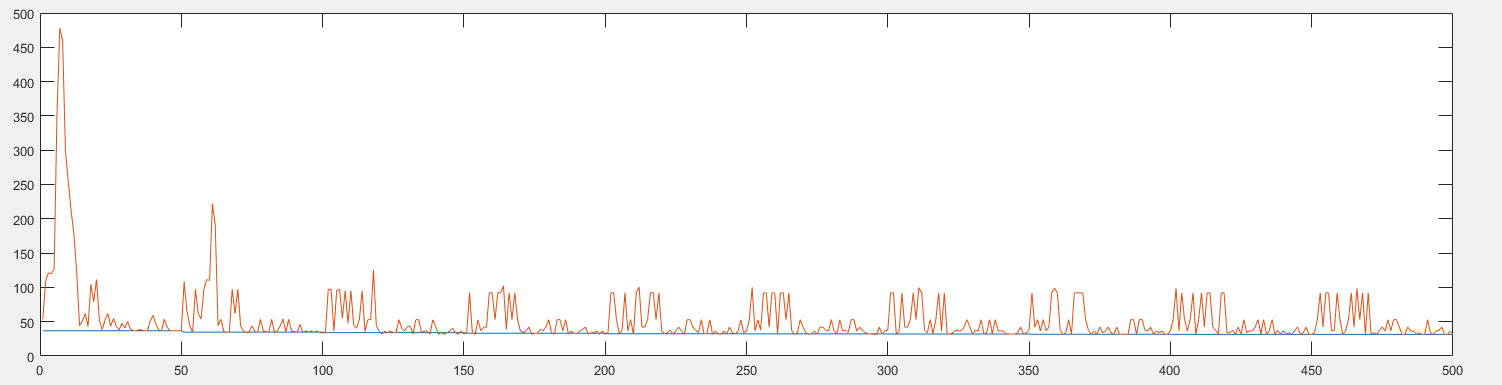


Fig 4. Erlang Distributed Service Time

Above figure is the result of using different basic service time for RL and SC. We can see most of the time RL is worst than SC, but there are times where RL is better than SC.

Lets see what happens when we run the same for longer time frames.

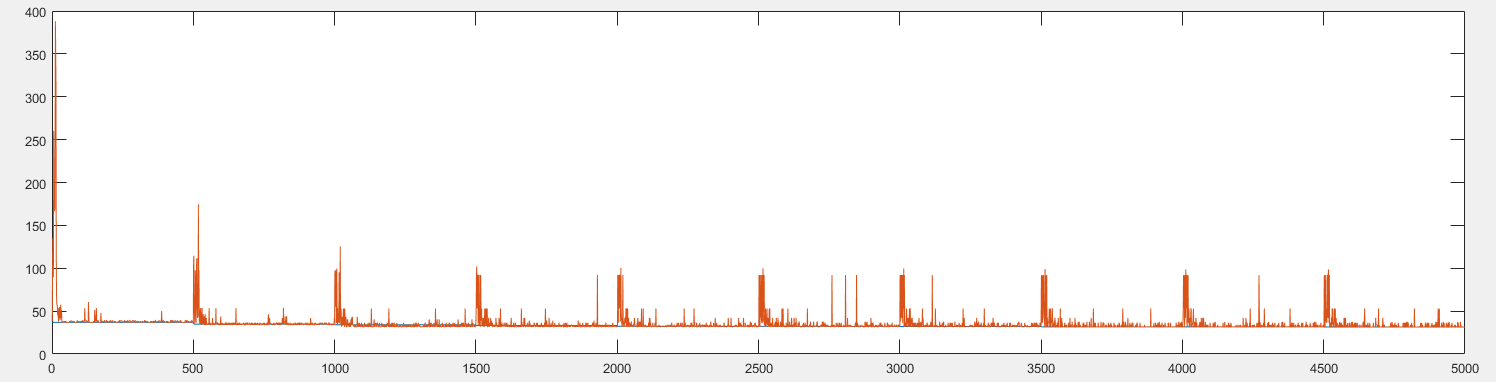


Fig 5.i Erlang Distributed Service Time

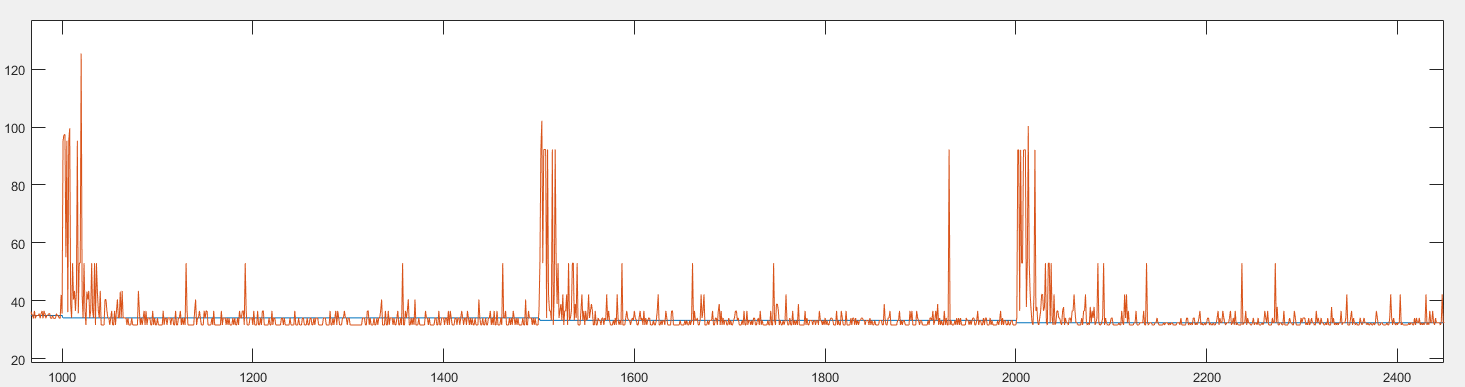


Fig 5.ii Zoomed version of 5.i

We can see that there are so many places where RL is better than SC. SC always operated at a fix configuration, and is a straight line because the random numbers generated at each time frame is same. Another observation is that RL is better than SC only upto inter-arrival rates of 300. It approximates the performance of SC after that, but never beats it above 300.

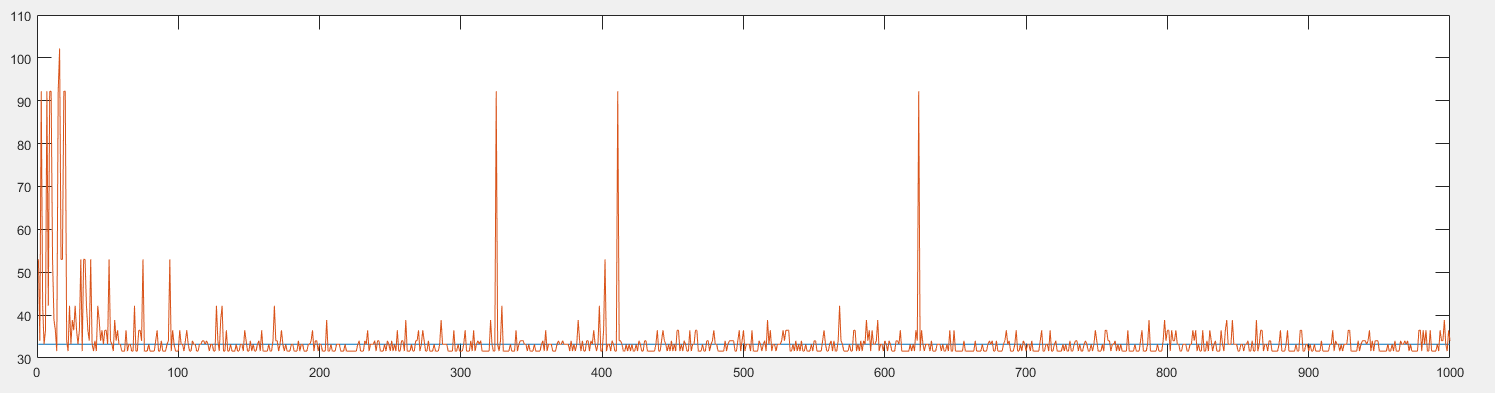


Fig 5.iii Erlang at inter-arrival rate of 200

This figure shows the performance of RL and SC for inter-arrival rate of 200. Clearly, Rl performs better than SC most of the times.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Let us repeat these experiments for other service distributions.**

Let us try the uniformly distributed version of the parameters that we have.

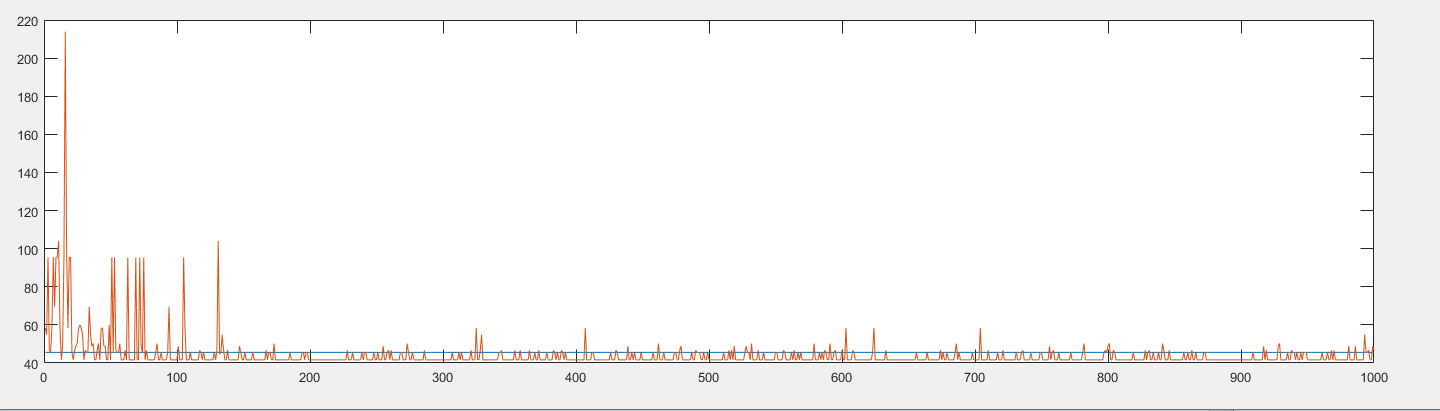


Fig. 6.i Uniformly Distributed **Paramter1**

This is the case when each time frame has same average basic service time for SC. We can see that RL performs better than SC in most of the cases.

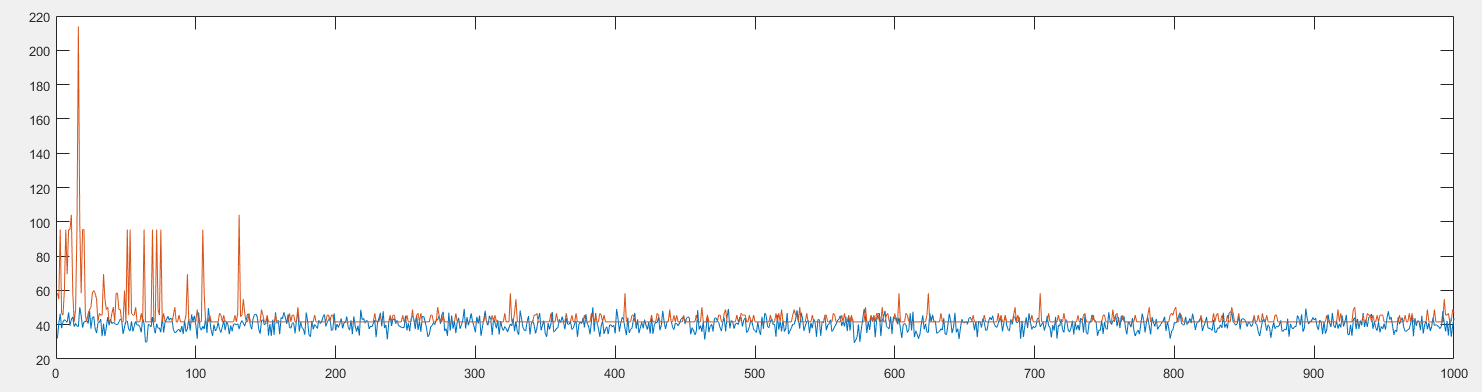


Fig 6.ii Uniformly distributed

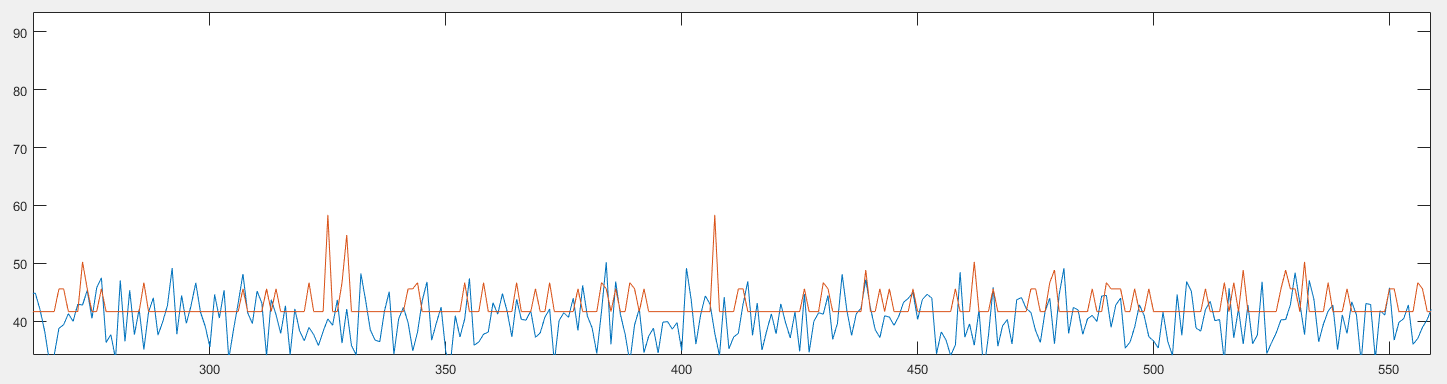


Fig 6.iii Uniformly Distributed (zoomed version of Fig 6.ii)

Fig 6.iii shows the performance comparison when the basic service distribution is uniformly distributed, and each time frame has different random basic service time. Figure 6.i had same average basic service time for each time frame. In this case, the RL algorithm converges to suboptimal configuration. WE can see that most of the times RL performs badly compared to SC.

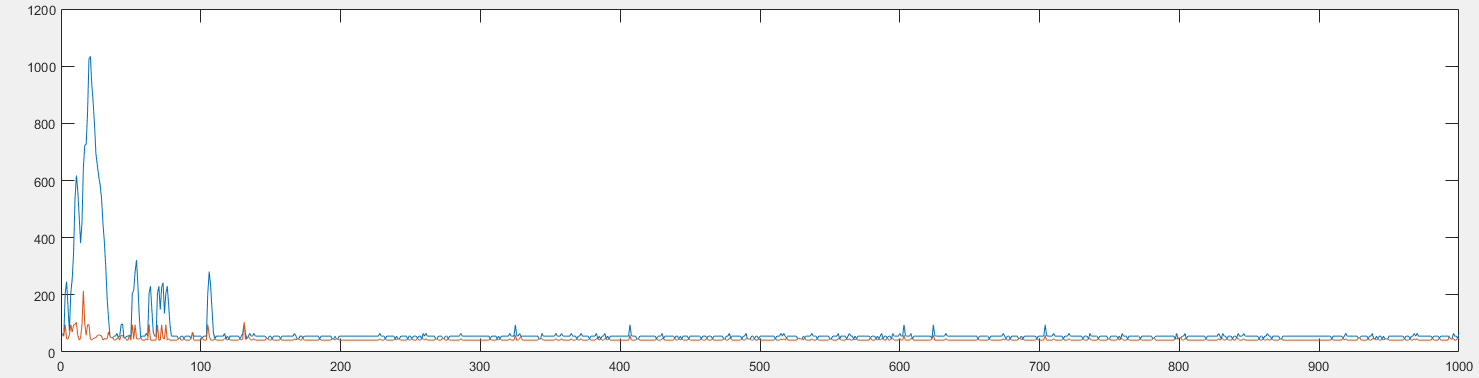


Fig. 7 I Uniformly Distributed Service Time ( both RL and SC were passed same basic service time)

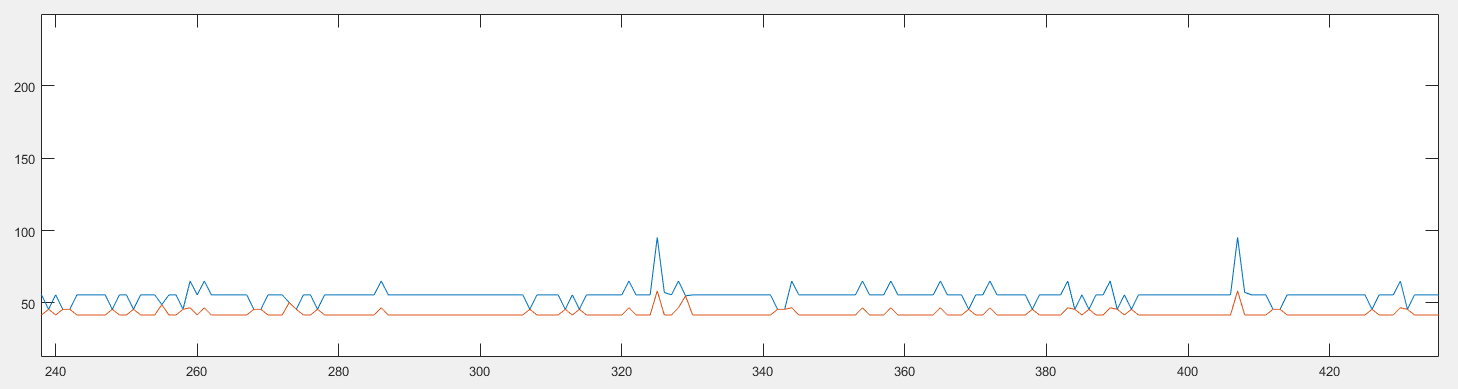


Fig 7 ii Zoomed version of Fig 7 i

Fig 7 shows the performance of RL and SC when requests are processed by generating same basic service time for different requests.

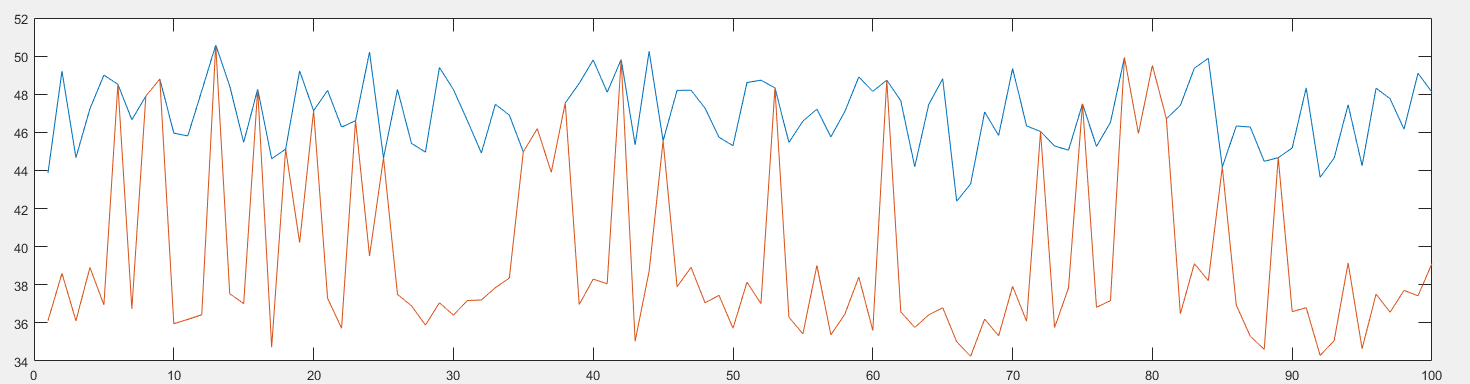


Fig 8

Figure 8 shows the performance comparision for RL and SC when requests are processed by supplying same random basic service time to both RL and SC.