Homework 2: Load Balancing
Group: Markov

Trina Sahoo (1901254),Mohanraj Chandrasekar(1921450)
Debodeep Banerjee (1901253), Moamen Abdrabbu(1894967)

Part I

(i) Expression of β as a function of ρ, N, α and E(T):
We know, $E(X) = \beta\Gamma\left(1+\frac{1}{\alpha}\right)$ and, $\rho = \frac{E(X)}{N.E(T)}$
Following the above two equations, we get, $\beta = \frac{\rho.N.E(T)}{\Gamma\left(1+\frac{1}{\alpha}\right)}$

(ii) Mean Delay Plot
In the Mean Delay Plot, the curve represents a function ρ for $\rho = [0.8, 0.82, 0.84, 0.86, 0.88, 0.9, 0.92, 0.94, 0.96, 0.98]$,N=20 and n=100000. Pod3 is having the delay slightly higher as compared to JBT-d as both are similar in the sense that pod selects random 3 servers and choose the shortest queue to assign task whereas JBT maintain table to keep the server which has below threshold that is 3 so that it will soon fill in and it will start assigning task randomly. As both are almost randomly assigning task the latency is high when compared to the JSQ. The JSQ has the least Mean Delay curve as it is always assign a task based on shortest queue.

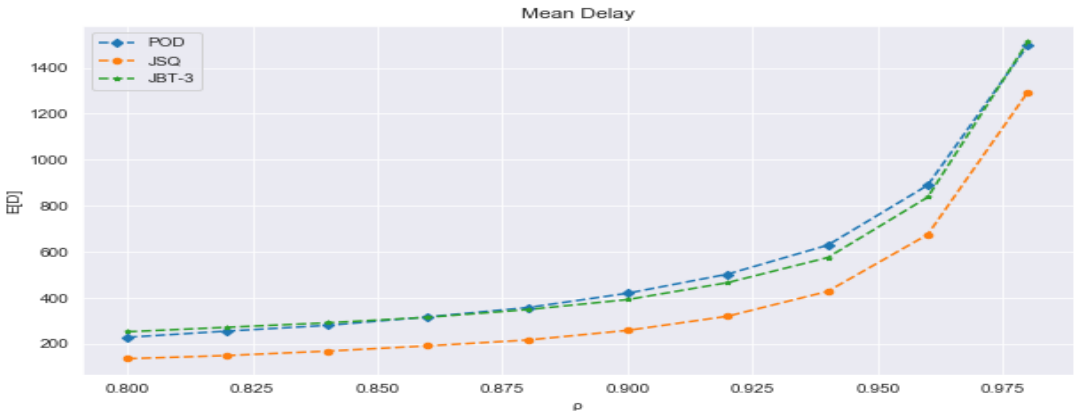

Figure 1: **Mean Delay Plot**

(iii) Message Overhead Plot
The Message Overhead Plot that is mean number of messages per task arrival sent from the load balancer to N=20 homogenous servers. As we can see that the message overhead is High in JSQ as it will always send N messages to servers and receives N messages from servers. The Pod3 will always have message overhead as 2*d that is 6 because the dispatcher will send messages only to those randomly selected 3 servers and receives back message from those servers. The JBT-d has the least trend of Message Overhead as the server only message to load balancer whenever it is below threshold. The messege overhead of the JBT-d is $\leq 1 + \frac{N+2d}{T}$.

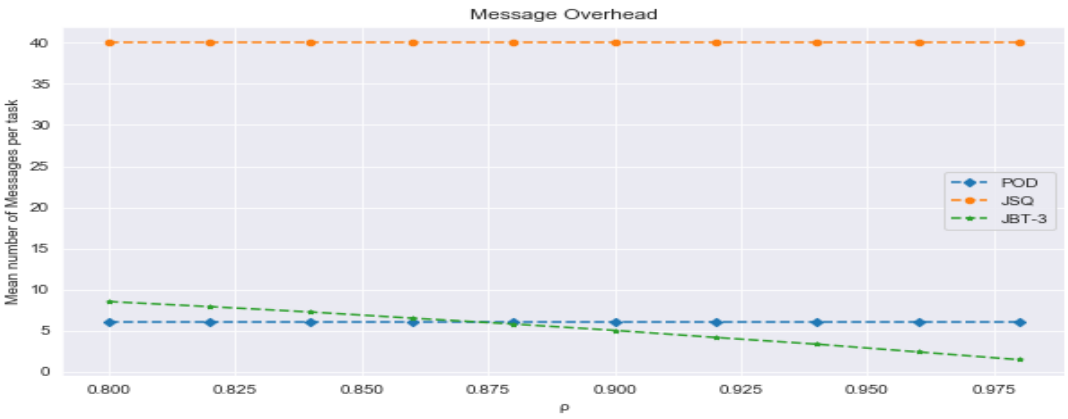

Figure 2: **Message Overhead Plot**

Part II

(i)New Algorithm:Join Shortest Waiting Time Queue (JSWTQ)

It is the slightly Modified Algorithm of JSQ. The algorithm is such that upon arrival of task at the load balancer look up the table structure of N bits. If bit is 1 then its the minimum waiting time server among all. Then the task is assigned to any one of the server that is available. In this algorithm atmost one server will be available similar to JSQ. But in Heavy Traffic the algorithm will have the Message overhead same as JSQ eventhough having table structure. Because the dispatcher has to reset the table to find out the minimum waiting time server by polling all servers asking what is the service time of all the tasks that are in the queue.

(ii)Mean Delay Plot

The given plot is the Mean delay of the tasks arrives at the Load Balancer which is plotted for different values of ρ[0.8,0.99], N=20 and n = 100000. JSQ is having slightly higher variance of delay from the new algorithm that we designed for load balancer. The main reason is that JSQ may end up choosing shortest queue which is having more processing time than the longer queue. The newly designed algorithm will always assign the task where it has the minimal waiting time thus causing it less delay when compared to traditional JSQ.

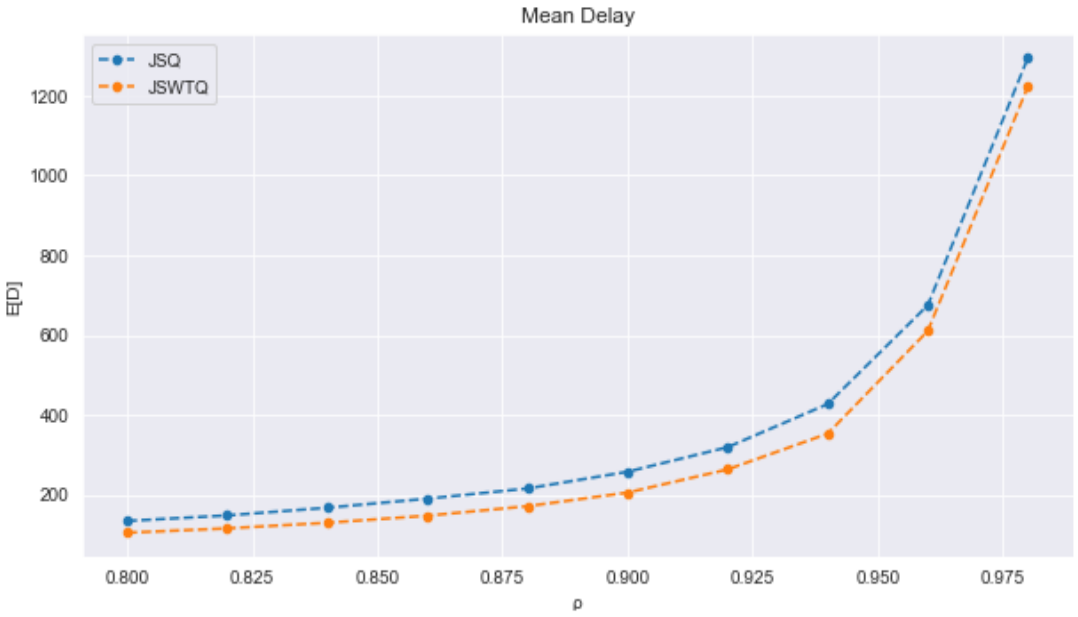

Figure 3: **Mean Delay Plot: Comparison between JSQ and JSWTQ**

(iii)Message Overhead Plot

The Plot shows Message Overhead of the JSQ and new algorithm. The JSQ has the Message overhead of 2*N(number of servers) that means it will be receiving and sending messages between N servers to keep track of shortest queue at servers Whereas the New algorithm designed similar to the JSQ and having table structure like JBT-d but still during Heavy traffic of tasks arrives at load balancer it has to reset table by conducting same poll like JSQ, making message overhead of new algorithm is also 2*N(Number of servers).

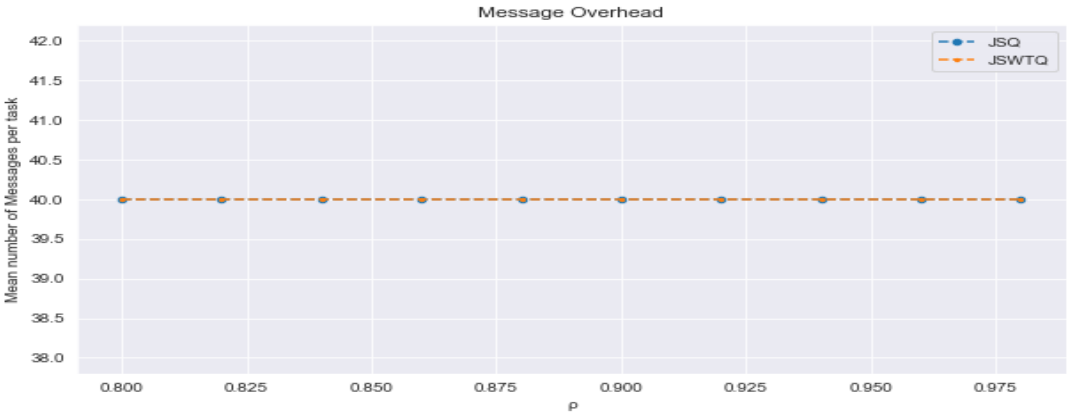

Figure 4: **Message Overhead Plot: Comparison between JSQ and JSWTQ**