ECE 368 Project 1 Report

64 Processor Simulation Program Report

Tatparya Shankar: tshankar
Harika Thatukuru: hthatuku

Introduction:

Through this project we created an event driven simulation for a multiprocessor system using a priority queue. Our simulation has two different modes. In mode 1, we take the following inputs:

- service rate
- arrival rate of priority 0 tasks
- arrival rate of priority 1 tasks
- number of tasks of each priority

In mode 2, we take in an input file.

Through rigorous calculations and simulation, we are able to produce the following outputs:

- average waiting time of priority 0 tasks
- average waiting time of priority 1 tasks
- average queue length
- average utilization of each processor
- average load balancing factor

In this report we have included the necessary calculations, results of our simulation, and plots depicting the outputs.

Calculations:

Arrival Time Calculations:

$\mu = 0.2$ (given)

10,000 arrivals in each group (given)

Equations:

$\rho 0 = \lambda 0 / (4 * \mu)$ $\lambda 0 = \rho 0 * (4 * \mu)$

$\rho 1 = \lambda 1 / (4 * \mu)$ $\lambda 1 = \rho 1 * (4 * \mu)$

Plot 1: $\rho 1 = 0.3$

$\lambda 1 = \rho 1 * (4 * \mu) = 0.3 * (4 * 0.2) = \mathbf{0.24}$

$\rho 0 = 0.1$

$\lambda 0 = \rho 0 * (4 * \mu) = 0.1 * (4 * 0.2) = \mathbf{0.08}$

$\rho 0 = 0.3$

$\lambda 0 = \rho 0 * (4 * \mu) = 0.3 * (4 * 0.2) = \mathbf{0.24}$

$\rho 0 = 0.5$

$\lambda 0 = \rho 0 * (4 * \mu) = 0.5 * (4 * 0.2) = \mathbf{0.40}$

$\rho 0 = 0.7$

$\lambda 0 = \rho 0 * (4 * \mu) = 0.7 * (4 * 0.2) = \mathbf{0.56}$

Plot 2: $\rho 0 = 0.3$

$\lambda 0 = \rho 0 * (4 * \mu) = 0.3 * (4 * 0.2) = \mathbf{0.24}$

$\rho 1 = 0.1$

$\lambda 1 = \rho 1 * (4 * \mu) = 0.1 * (4 * 0.2) = \mathbf{0.08}$

$\rho 1 = 0.3$

$\lambda 1 = \rho 1 * (4 * \mu) = 0.3 * (4 * 0.2) = \mathbf{0.24}$

$\rho 1 = 0.5$

$\lambda 1 = \rho 1 * (4 * \mu) = 0.5 * (4 * 0.2) = \mathbf{0.40}$

$\rho 1 = 0.7$

$\lambda 1 = \rho 1 * (4 * \mu) = 0.7 * (4 * 0.2) = \mathbf{0.56}$

Table 1: Table for Plot 1 ρ0 vs Simulation Outputs

Service Rate (mU)	0.2	0.2	0.2	0.2
p0	0.1	0.3	0.5	0.7
p1	0.3	0.3	0.3	0.3
Arrival Rate 0 (Λ0)	0.08	0.24	0.4	0.56
Arrival Rate 1 (Λ1)	0.24	0.24	0.24	0.24
Waiting Time 0	0.0002	0.0001	0.0001	0
Waiting Time 1	0.1112	0.0463	0.0215	0.0162
Average Queue Length	0	0	0	0
Average Utilization of Processor	0.44205	0.44447	0.30859	0.23349
Average Load Balancing Factor	0.66652	0.66665	0.66682	0.66702

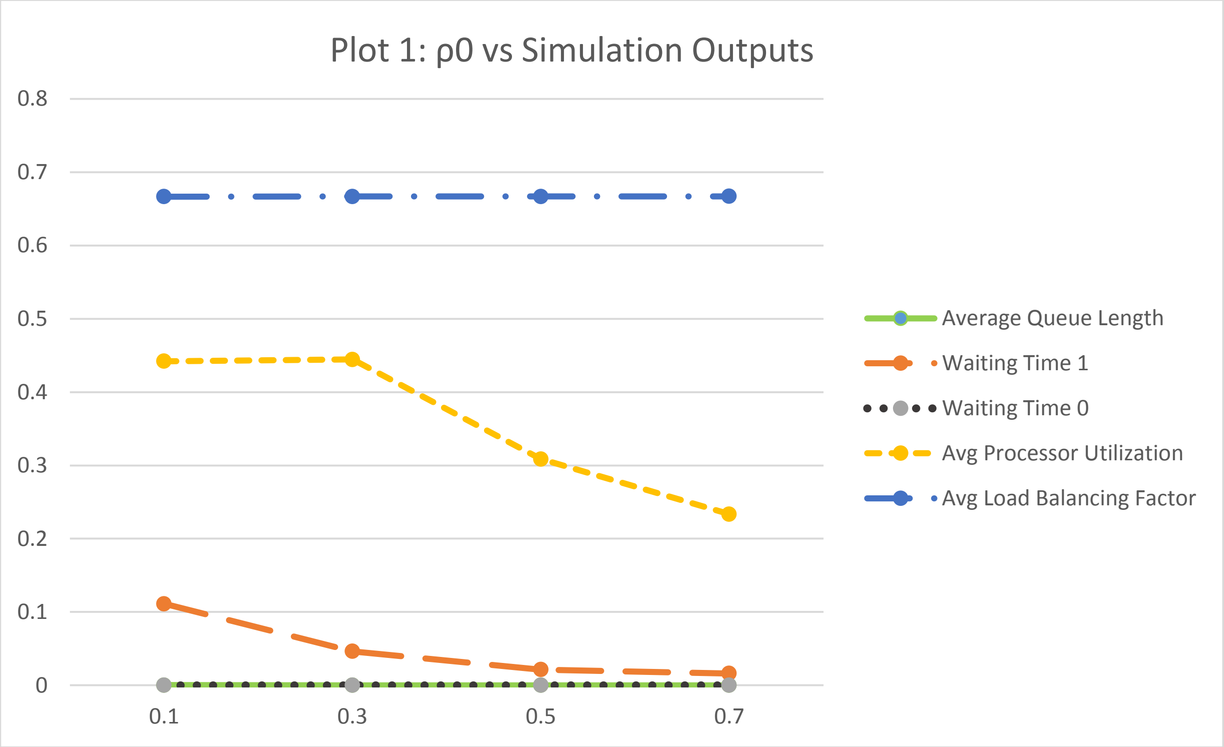
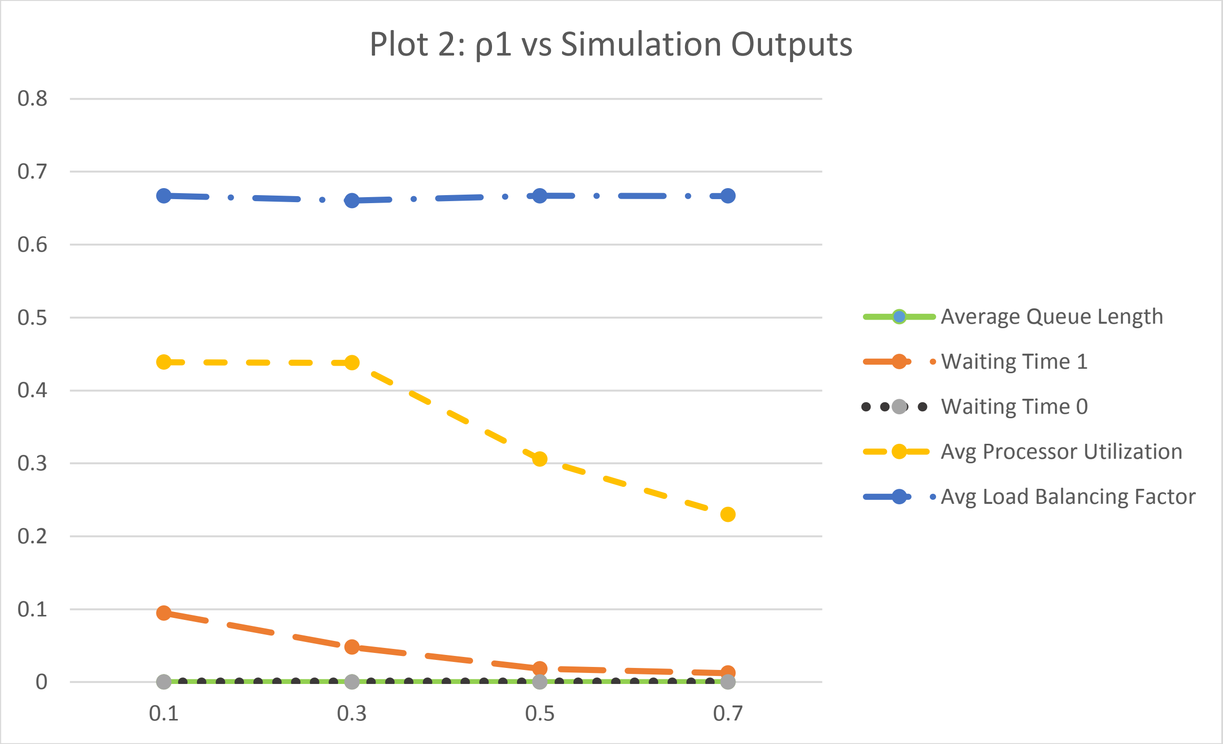


Table 2: Table for Plot 1 ρ1 vs Simulation Outputs

Service Rate (mU)	0.2	0.2	0.2	0.2
P0	0.3	0.3	0.3	0.3
P1	0.1	0.3	0.5	0.7
Arrival Rate 0 (Λ0)	0.24	0.24	0.24	0.24
Arrival Rate 1 (Λ1)	0.08	0.24	0.4	0.56
Waiting Time 0	0.0001	0	0.0001	0
Waiting Time 1	0.0947	0.048	0.0184	0.0123
Average Queue Length	0	0	0	0
Average Utilization of Processor	0.43887	0.438	0.30565	0.2298
Average Load Balancing Factor	0.6669	0.66032	0.66689	0.66676



Execution:

The following depicts how the program simulation works:

For the simulation of the 64 processor system, we maintain the 4 following queues in our program:

1. **AllTaskQueue**: Queue stores all of the tasks that need to be processed in the system
2. **Q0**: Queue stores all of the tasks of priority 0 that are in the system but have not been served yet
3. **Q1**: Queue stores all of the tasks of priority 1 that are in the system but have not been served yet
4. **ProcessQ**: The queue is used to keep track of all the subtasks being processed by the system at any given time

The following are the main functions that are handled by the system for the simulation:

Creation of Tasks Queue (FEL):

In case of Mode 1, the program using inputs provided for average arrival time 0, average arrival time 1, average service time and the number of tasks, creates tasks of priorities 1 and 0 and stores them in a common queue (AllTaskQueue).

In case of Mode 2, the program simply goes through the input file, reading every line one by one and creating corresponding task elements and stores all of them in the common queue.

Task Arrival:

A main while loop with a time variable keeps track of the current time and checks for the arrival time of elements in the common task queue; if the arrival time of an element is same as the current time, then the element is popped from the common queue and then pushed on to a priority queue (Q0 / Q1 depending on its priority.

Task Service:

The same while loop above, keeps track of the number of available processors in the system. For every iteration, the system checks if any task in either of the priority queues can be served using the number of currently available processors. If a task can be served, then it is popped off the priority queue and added to the queue of tasks being processed.

Task Departure:

The same while loop with every iteration checks if any subtask in the subtask queue has been served and is ready to depart. If any element has been served, then the element is popped off the queue and is deleted from the system to avoid memory errors.

Conclusion:

In summation, our event driven simulation for a multiprocessor system using a priority queue is able to produce accurate and consistent output as depicted in the plots above. Our outputs are appropriate for varying arrival rates of priority 0 tasks and for varying arrival rates of priority 1 tasks. Hence, we are able to simulate any event driven simulation for a multiprocessor system using a priority queue and produce accurate output results.