

User Manual for Store Manager Application

1.

Overview

This Store Manager application simulates a queuing system with multiple servers. This project explores event-driven programming, system modeling, and continuous probability distributions (Poisson, Exponential). The simulation compares predicted analytical results against a random, event-driven timeline utilizing a custom Min-Heap priority queue and a standard FIFO queue as requested in the project requirements.

This Store Manager app uses a MMc Queuing System

M — Markovian Arrivals Customers arrive randomly according to a Poisson distribution. Time between arrivals follows an exponential curve and the probability of the next arrival is not dependent on wait time of current arrival. (λ) is used to represent the average arrival rate measured in customers per hour.

M — Markovian Service Times Time to service is exponentially distributed, and is without “memory” like the arrivals. The remaining service time for a given customer is independent of how long that server has been busy. (μ) is used to represent the average service rate, measured in number of customers per hour served, per server. (Ex: 2 customers/h per server)

c — Number of Servers How many parallel service channels exist. When $c = 1$, we’re dealing with a single-server queue (M/M/1). If $c = 3$, three parallel service channels exist. Customers waiting in a single shared line get routed to the next available server.

2.

Customer Data Type (Event Node)

In this simulation, the `Customer` class serves two purposes. It acts as the **event payload** for our timeline and the **waiting customer** in our line.

A single `Customer` object is used to represent either an arrival or a departure:
* **Event Type (type):** An enum with two values indicating the object as either an ARRIVAL or a DEPARTURE.
* **Priority Queue Time (pqTime):** The time this event occurs. This acts as the key for our Min-Heap Priority Queue, meaning customer arrival time can be compared against server completion time and trigger the simulation to process the next event.
* **Simulation Times:** Tracks `arrivalTime`, `startOfServiceTime`, and `departureTime`. As the event moves through our simulation these times are calculated and recorded and are

ultimately used to calculate wait times, system utilization, and averages. *

Linked List Pointer (`nextCust`): A reference/pointer to the next node, allowing the customer to be placed in the queue when all servers (`c`) are currently busy.

3.

Variables and Measurement Outputs

- **P₀:** The probability that the entire system is empty (zero people in line, zero people being served).
- **L:** The average number of people in the entire system (in line + being served).
- **W:** The average time a customer spends in the entire system.
- **L_q:** The average number of customers waiting in the FIFO queue.
- **W_q:** The average time a customer spends waiting in the FIFO queue.
- **(Rho):** The system utilization factor (how busy the servers are on average).

4.

Input Config

The simulation isn't interactive, and reads from two static files, **test1.txt** and **test2.txt** which are located in the root directory with the executable.

Each file must contain exactly four lines of data, representing the following values in order:

1. **Lambda ()**: Rate of arrival
2. **Mu ()**: Rate of service
3. **M / c**: Number of available servers
4. **Total Events**: Number of arrivals/departures to simulate

Example test1.txt format:

2 3 2 5000

5.

Compilation and Execution

The app requires a Linux environment to compile and run. Here are the steps:

1. Open terminal and navigate to the root directory
2. Run **make** in the terminal to compile the code

3. Execute binary file ./simulation to run the application
4. Clean directory of excess files by using **make clean**
- 5.

Interpretation of Results

The program automatically processes both test1.txt and test2.txt. All output is shown directly in the console.

For each test file, the model will display:

Analytical Results: The predicted results based on theoretical math and the given input variables **Simulation Results:** The actual recorded values for the measures after having run the simulation. Because this sim is driven by random number generation and Exponential/Poisson distributions, it will **approximate** the analytical results, but will vary marginally from the analytical predictions.