Analyzing Airport Security Lines

Group 7:
Aastha Khandelwal
Sayali Manish Raje
Sajal Agarwal

We first take a look at the three parts of a queuing system:

- The arrival or inputs to the system (calling population)
- The queue or the buffer
- The service facility or server

These three components have certain characteristics that must be examined before mathematical queuing models can be developed.

Arrival Characteristics

The input source that generates arrivals or passengers for the service system has three major characteristics. It is important to consider the size of the calling population, the pattern of arrivals at the queuing system, and the behavior of the arrivals.

Size of the Calling Population:
Population sizes are considered to be either unlimited (essentially infinite) or limited (finite).
Pattern of arrivals at the System:
The number of arrivals per unit of time can be estimated by a probability distribution known as the Poisson distribution. For any given arrival rate, a discrete Poisson distribution can be established by using the formula

$P(n,t) = e^{-\lambda t} \frac{(\lambda t)^n}{n!}$ for n = 0,1,2,3,4…

Waiting Line Characteristics

The waiting line itself is the second component of a queuing system. The length of a line can be either limited or unlimited. A second waiting line characteristic deals with queue discipline. This refers to the rule by which passengers in the line are to receive service. Most systems use a queue discipline known as the first-in, first-out rule (FIFO).

Service Facility Characteristics

The third part of any queuing system is the service facility. It is important to examine two basic properties:

- The configuration of the service system
- The pattern of service times

Now we will observe how different models and optimizations can enhance the efficiency of the airport security screening process. Every model has the following features:

- Arrivals are described by a Poisson probability distribution.
- Service times also vary from one passenger to the next and are independent of one another, but their average rate is known.
- Service times occur according to the exponential probability distribution.
- The average service rate is greater than the average arrival rate (stability condition).

Formulae for single server system

- Average Waiting Time = $\frac{\lambda}{\mu(\mu-\lambda)}$
- Average Queue Length = $\frac{\lambda^2}{\mu(\mu-\lambda)}$
- System Utilization = $\frac{\lambda}{\mu}$

Single Server with infinite buffer (M/M/1)

Simulation readings

S.No	λ	μ	Theoretical Average Waiting Time	Empirical Average Waiting Time	Theoretical Average Queue Length	Empirical Average Queue Length	Theoretical System Utilization	Empirical System Utilization
1	3	6	0.1667	0.1683	0.5000	0.5436	0.5000	0.5110
2	5	15	0.0333	0.0262	0.1667	0.1361	0.3333	0.3521
3	10	50	0.0050	0.0052	0.0500	0.0533	0.2000	0.1939

Single Server with finite buffer (M/M/1/K)

Simulation readings

S.No	λ	μ	K	Theoretical Average Waiting Time	Empirical Average Waiting Time	Theoretical Average Queue Length	Empirical Average Queue Length	Theoretical System Utilization	Empirical System Utilization
1	3	6	7	0.1667	0.1323	0.5000	0.4235	0.5000	0.5274
2	3	15	5	0.0166	0.0169	0.0500	0.0485	0.2000	0.1960
3	8	33	10	0.0097	0.0062	0.0775	0.0482	0.2424	0.2155

Multi-Server with infinite buffer (M/M/m)

Simulation readings

S.No	λ	μ	m	Empirical Average Waiting Time	Empirical Average Queue Length	Empirical System Utilization
1	3	6	5	0.0418	0.1195	0.9912
2	13	60	7	0.0030	0.0387	0.0293
3	8	15	3	0.0195	0.1553	0.1741

Multi-Server with finite buffer (M/M/m/K)

Simulation readings

S.No	λ	μ	m	K	Empirical Average Waiting Time	Empirical Average Queue Length	Empirical System Utilization
1	3	6	5	5	0.0396	0.1075	0.0873
2	9	20	5	10	0.0138	0.1330	0.0942
3	5	7	5	2	0.0546	0.2708	0.1392

Code Explanation

The provided C++ code simulates a queuing system with multiple configurations, including single-server and multiple-server setups with both infinite and finite buffers. The simulation is based on the principles of queueing theory and uses threads to model the arrival and processing of passengers in a service center.

Key Components:

1. Passenger Class:
    a. Represents a passenger with attributes such as inter-arrival time, global arrival time, and processing time.
    b. The inter-arrival and processing times are generated using exponential distributions.
2. Global Variables:
    a. arrival_rate and service_rate define the rates for passenger arrival and service.
    b. global_time tracks the simulation time.
    c. Various other global variables such as total_waiting_time, start_proc_time, passenger_cnt, etc., store simulation metrics.
3. Queues and Buffers:
    a. The simulation uses a queue (security_line) to model the waiting line for processing.
    b. For multiple servers, there are multiple queues (security_lines), each associated with a server.
4. Threads:
    a. Threads are used to simulate passenger arrivals and processing concurrently.
    b. Mutexes are employed to ensure thread safety, especially when accessing shared resources like queues and global variables.
5. Simulation Functions:
    a. Functions like simulateArrivals_singleServer_infiniteBuffer, simulateProcessing_singleServer_infiniteBuffer, etc., represent different configurations of the simulation.
6. Main Function:
    a. Presents a menu for the user to choose the simulation configuration.

7. Simulation Flow:
    a. The program starts by presenting a menu to the user.
    b. Based on the user's choice, it runs the corresponding simulation function.
    c. The simulation functions use threads to simulate passenger arrivals and processing.
    d. Mutexes ensure thread safety when accessing shared resources, such as while enqueueing and dequeuing in the security line queue
    e. After the simulation, it prints theoretical and empirical metrics, including average waiting time, average queue length, and service utilization.

Observations and Analysis

- **Higher arrival rates** in single server systems lead to **higher system utilization**. In case of a finite buffer, higher arrival rate will lead to more packet loss. This highlights the need for incorporating multiple servers to handle passengers more efficiently.

- Introduction of a multi-server system showed a significant **decrease in waiting time** and **decrease in average queue length**. This is due to the fact that more security scanners (servers) are available for passengers. So, passengers can be parallelly processed thus reducing the average amount of time spent in the queue.

- Multi-server systems are beneficial for higher arrival rate in case of finite buffers. This will lead to **reduction in packet drop** since multiple servers are available.

- Multi-server queues **can handle a larger volume of customers** or tasks simultaneously, leading to a higher system capacity. This can be especially **beneficial in situations with high arrival rates**, **reducing the risk of congestion** and long waiting times.

- Multi server systems provide **better resource utilization**. In single-server queues, when the server is idle, no work is being done. In multi-server queues, **idle servers can be quickly assigned to serve incoming customers**, reducing resource wastage**.**

Conclusion:

The code successfully models queuing systems with various configurations. It demonstrates key concepts of queuing theory, including arrival rates, service rates, waiting times, and queue lengths. The use of threads and mutexes ensures a concurrent and thread-safe simulation.