



Electrical & Computer
Engineering Department
**UNIVERSITY OF
PELOPONNESE**

M/M/S/ ∞ Queue Simulation

Queueing Theory

Students

Mantvydas Deltuva ece24601

Justinas Teselis ece24600

Professor

Χριστίνα (Τάνια) Πολίτη

Patras, 2024

Contents

Overview	4
Events in the Simulation	4
Event Determination Process	5
Simulation Parameters	6
Simulation Flow	7
Visualizations	8
Number of Calls in the System Over Time	8
Probability Distribution of Calls in the System	9
Probability of Delay Across Server Capacities and Server Utilization Across Server Capacities	10
Source Code	11

Figures

1 pic. Number of Calls in the System Over Time Visualization	8
2 pic. Probability Distribution of Calls in the System Visualization	9
3 pic. Probability of Delay Across Server Capacities and Server Utilization Across Server Capacities Visualization	10

Overview

This report shows an implementation of an **M/M/S/ ∞ queueing model**, a classic queueing system used to study scenarios with a finite number of servers and an infinite capacity for queuing. Calls which cannot be immediately served join the queue instead of being lost.

The M/M/S/ ∞ model is defined by:

- **M** (Markovian Inter-Arrival Times): The time between successive arrivals follows an exponential distribution.
- **M** (Markovian Service Times): The service times are exponentially distributed.
- **S** (Finite Servers): A fixed number of servers are available to serve arriving calls.
- **∞** (Infinite Queue Capacity): Calls can wait indefinitely in the queue if all servers are busy.

This model is widely applicable in areas such as customer service centers, manufacturing systems, and computing systems, where queuing is permitted to accommodate demand exceeding server capacity.

Events in the Simulation

The simulation revolves around two types of **events**:

1. **Arrival Event:** A new call enters the system. If a server is available, the call is served. Otherwise, the call joins the queue. Each arrival event generates the next arrival time (*next_arrival_time*) by sampling from the exponential inter-arrival time distribution.
2. **Departure Event:** A call completes its service and leaves the system. If there are calls in the queue, the next one starts service.

Each event updates the system state, including the number of active calls (*num_calls*), number of calls in queue (*num_queue_calls*), number of delayed calls (*num_delayed_calls*), and the time (*current_time*).

Event Determination Process

At any point, the simulation determines which event (arrival or departure) occurs next:

- **Arrival Time:** Calculated as the current time plus a random inter-arrival time ($\text{exprnd}(1 / \lambda)$). This is updated every time an arrival event occurs.
- **Departure Time:** The earliest scheduled departure time from the scheduled departures ($\text{scheduled_departure_times}$) list.

The simulation compares the next arrival time (next_arrival_time) and the earliest departure time ($\min(\text{scheduled_departure_times})$) to decide:

- If the arrival time is sooner, an **arrival event** occurs:
 - If servers are available ($\text{num_calls} < S$), the arriving call is served, the number of active calls (num_calls) increases, and a departure is scheduled.
 - If servers are full ($\text{num_calls} \geq S$), the call joins the queue (num_queue_calls), and the delayed calls counter (num_delayed_calls) is incremented.
 - Determines the next arrival time by sampling a new inter-arrival time.
- Otherwise, a **departure event** occurs:
 - If there are calls in the queue ($\text{num_queue_calls} > 0$), one is moved from the queue to service, and a new departure is scheduled.
 - If no one is in the queue, the number of active calls (num_calls) decreases.
 - Removes the completed departure from the schedule.

Simulation Parameters

- **Arrival Rate (λ):** The average number of arrivals per unit time.

$$\lambda = 10 \% \text{ arrivals / unit time}$$

- **Service Rate (μ):** The average number of services completed per unit time.

$$\mu = 2 \% \text{ services / unit time}$$

- **Number of Servers (S):** The total number of servers. This is an array for server utilization and delay probability statistics.

$$S_range = 1:1:20 \% \text{ number of servers}$$

- **Total Time:** The duration of the simulation.

$$total_time = 100 \% \text{ units}$$

Simulation Flow

1. Start the simulation clock at zero (*current_time* = 0).
2. Generate the first arrival event and initialize the system state:
 - Sample the time until the first arrival from the exponential distribution.
 - Log the initial system state (time and number of calls).
3. Iteratively process events by determining whether the next event is an **arrival** or **departure**:
 - Compare the time of the next arrival with the earliest departure.
 - Update the system state when **Arrival Event** occurs:
 1. If space is available, serve the call, increase the number of active calls, schedule a departure and generate the next arrival time.
 2. Otherwise, add a call to the queue, increment the delayed calls counter (*num_delayed_calls*) and generate the next arrival time.
 - Update the system state when **Departure Event** occurs:
 1. If there are calls in the queue (*num_queue_calls* > 0), one is moved from the queue to service, and a new departure is scheduled.
 2. If no one is in the queue, the number of active calls (*num_calls*) decreases.
 3. Remove the completed departure from the schedule.
 - Log the system state after each event (time and number of calls).
4. Continue until simulation time is exceeded (*current_time* >= *total_time*).

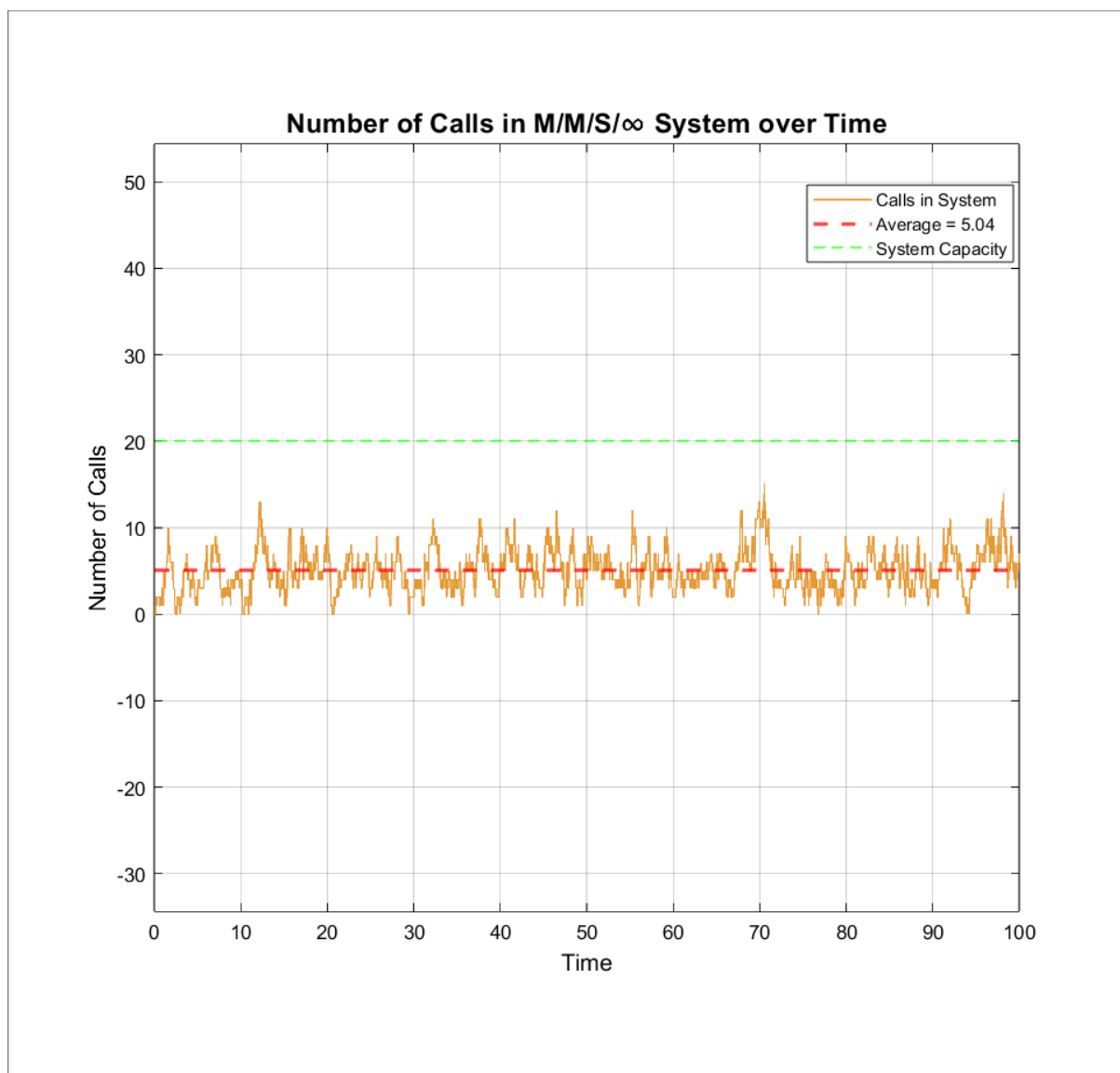
This process iterates until simulations are computed with all server capacities.

Visualizations

Number of Calls in the System Over Time

This plot tracks the evolution of the number of active calls during the simulation. The orange line represents the number of active calls (being served and delayed in queue) at each logged event. If the orange graph is above the green dashed line that indicates that the calls over it are in system queue. The red dashed line shows the time-weighted average number of calls, computed across the simulation duration. The green dashed line indicates the overall maximum system capacity.

This visualization is only created for the last server capacity (S) value. In this case for $S = 20$.

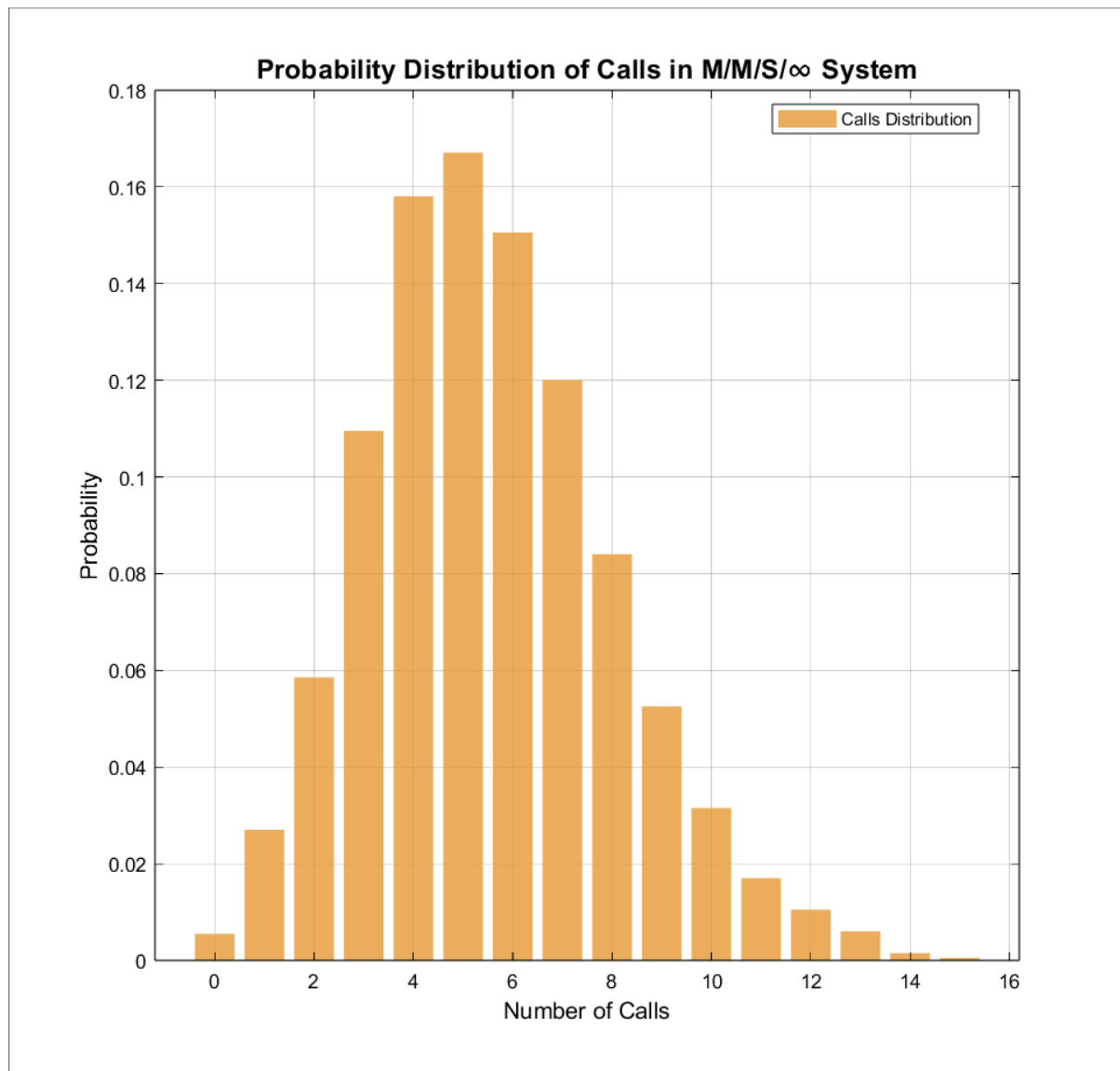


1 pic. Number of Calls in the System Over Time Visualization

Probability Distribution of Calls in the System

This histogram displays the probability distribution of the number of active calls (being served and delayed in queue) during the simulation. The distribution reflects the system's overall load.

This visualization is only created for the last server capacity (S) value. In this case for $S = 20$.

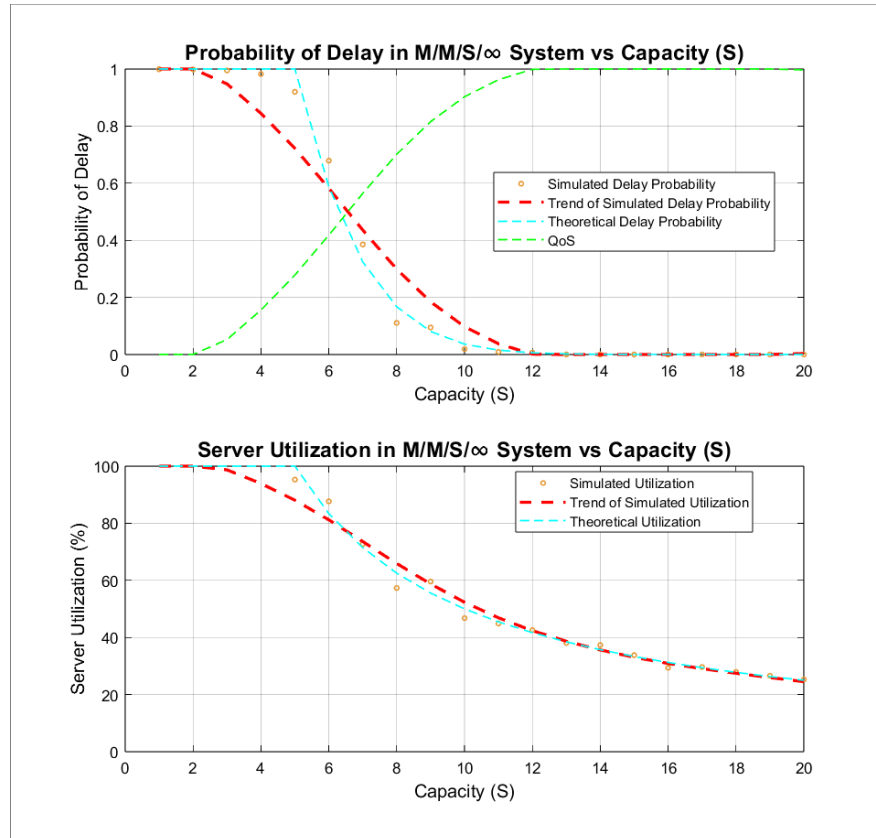


2 pic. Probability Distribution of Calls in the System Visualization

Probability of Delay Across Server Capacities and Server Utilization Across Server Capacities

This plot provides two separate graphs:

1. **Probability of Delay Across Server Capacities:** This plot shows the probability of call delay for different server capacities (S), based on fixed arrival rate (λ) and service rate (μ). The red dashed line represent the trend of simulated delay probabilities across server capacities. The green dashed line indicates simulated system overall quality of service across server capacities. The cyan dashed line represents theoretical delay probability across server capacities. Orange dots are simulated delay probabilities at each server capacity.
2. **Server Utilization Across Server Capacities:** This plot represents the server utilization for different server capacities (S), based on fixed arrival rate (λ) and service rate (μ). The red dashed line shows the trend of simulated utilization across server capacities. The cyan dashed line represents theoretical utilization across server capacities. Orange dots are simulated utilizations at each server capacity.



3 pic. Probability of Delay Across Server Capacities and Server Utilization Across Server Capacities Visualization

Source Code

The source code for the implementation of the M/M/S/ ∞ queueing model simulation is available on **GitHub**. It includes the full simulation logic, data visualization scripts, and documentation to help users understand and run the simulation with MATLAB. You can access the repository at the following link: [M/M/S/ \$\infty\$ Queue Simulation Source Code](#).