Performance Analysis of Computer Networks ECE_TEL951

Electrical & Computer Engineering Department
UNIVERSITY OF PELOPONNESE

M/M/S/Q Queue Simulation

Queueing Theory

Students

Mantvydas Deltuva ece24601

Justinas Teselis ece24600

Professor

Χριστίνα (Τάνια) Πολίτη

Patras, 2024

Contents

Figures

Overview

This report shows an implementation of an **M/M/S/Q queueing model**, a classic queueing system used to study scenarios with a finite number of servers and a finite queue capacity. Calls that cannot be immediately served join the queue if there is space, otherwise, they are dropped.

The M/M/S/Q 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.
- **Q** (Finite Queue Capacity): The queue has a finite capacity; calls that exceed this capacity are dropped.

This model is widely applicable in areas such as telecommunications, server networks, and customer service systems where queuing space is limited.

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*), number of dropped calls (*num_dropped_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 (*exprnd(1 / lambda)*). This is updated every time an arrival event occurs.
- **Departure Time:** The earliest scheduled departure time from the scheduled departures (*scheduled_departure_times*) list.

The simulation compares the next arrival time (*next_arrival_time*) and the earliest departure time (*min(scheduled_departure_times)*) to decide:

- If the arrival time is sooner, an **arrival event** occurs:
    - If servers are available (*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 (*num_calls >= S*) and the queue has space (*num_queue_calls < Q*), the call joins the queue (*num_queue_calls*), and the delayed calls counter (*num_delayed_calls*) is incremented.
    - If the queue is also full (*num_queue_calls >= Q*), the call is dropped and dropped calls counter (*num_dropped_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 (*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 = 30 % arrivals / unit time
```

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

```
mu = 2 % 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 % number of servers
```

- **Queue Capacity (Q):** The maximum number of calls that can wait in the queue.

```
Q = 3 % maximum number of calls in queue
```

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

```
total_time = 100 % 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. If servers are full but queue space is available, add a call to the queue, increment the delayed calls counter (*num_delayed_calls*), and generate the next arrival time.
        3. If both servers and the queue are full, increment the dropped calls counter (*num_dropped_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.

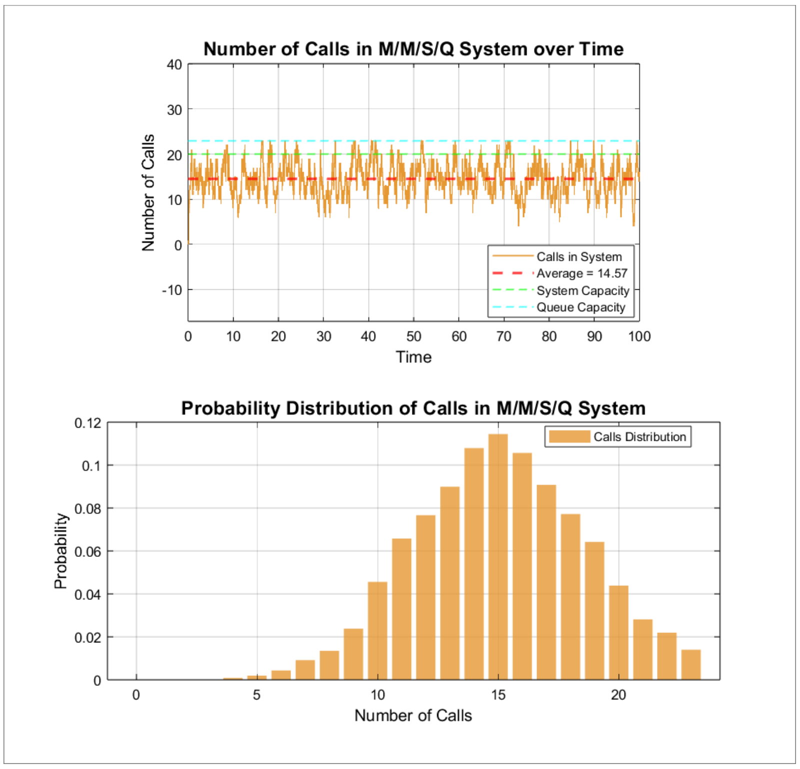
Visualizations

Number of Calls in the System Over Time and Probability Distribution of Calls in the System

This visualization consists of two graphs:

1. **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 cyan dashed line indicates the overall maximum system capacity (servers + queue). The red dashed line shows the time-weighted average number of calls, computed across the simulation duration.
2. **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.

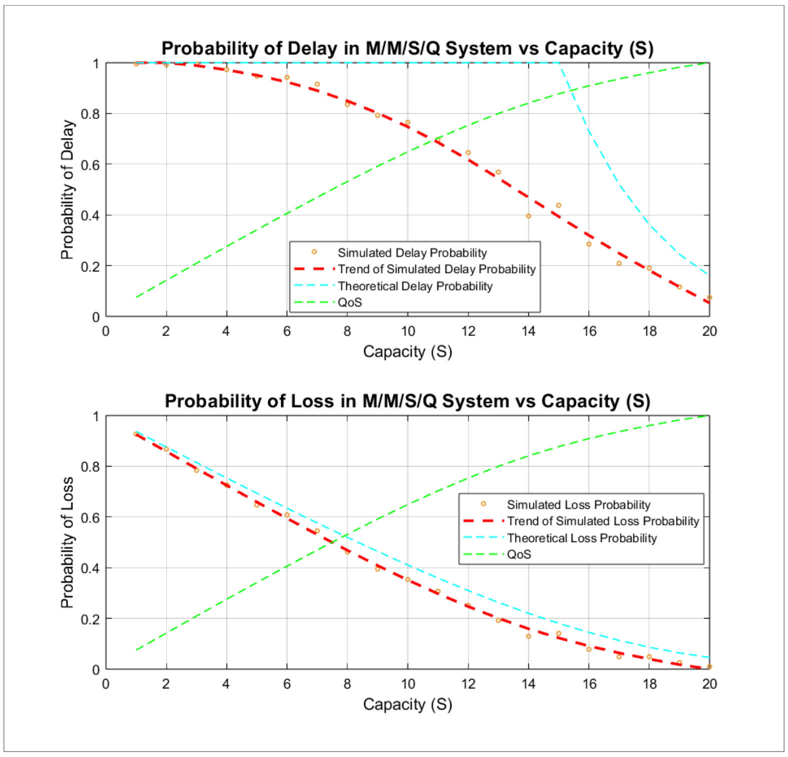


1 pic. Number of Calls in the System Over Time and Probability Distribution of Calls in the System Visualization

Probability of Delay Across Server Capacity and Probability of Loss Across Server Capacity

This plot provides two separate graphs:

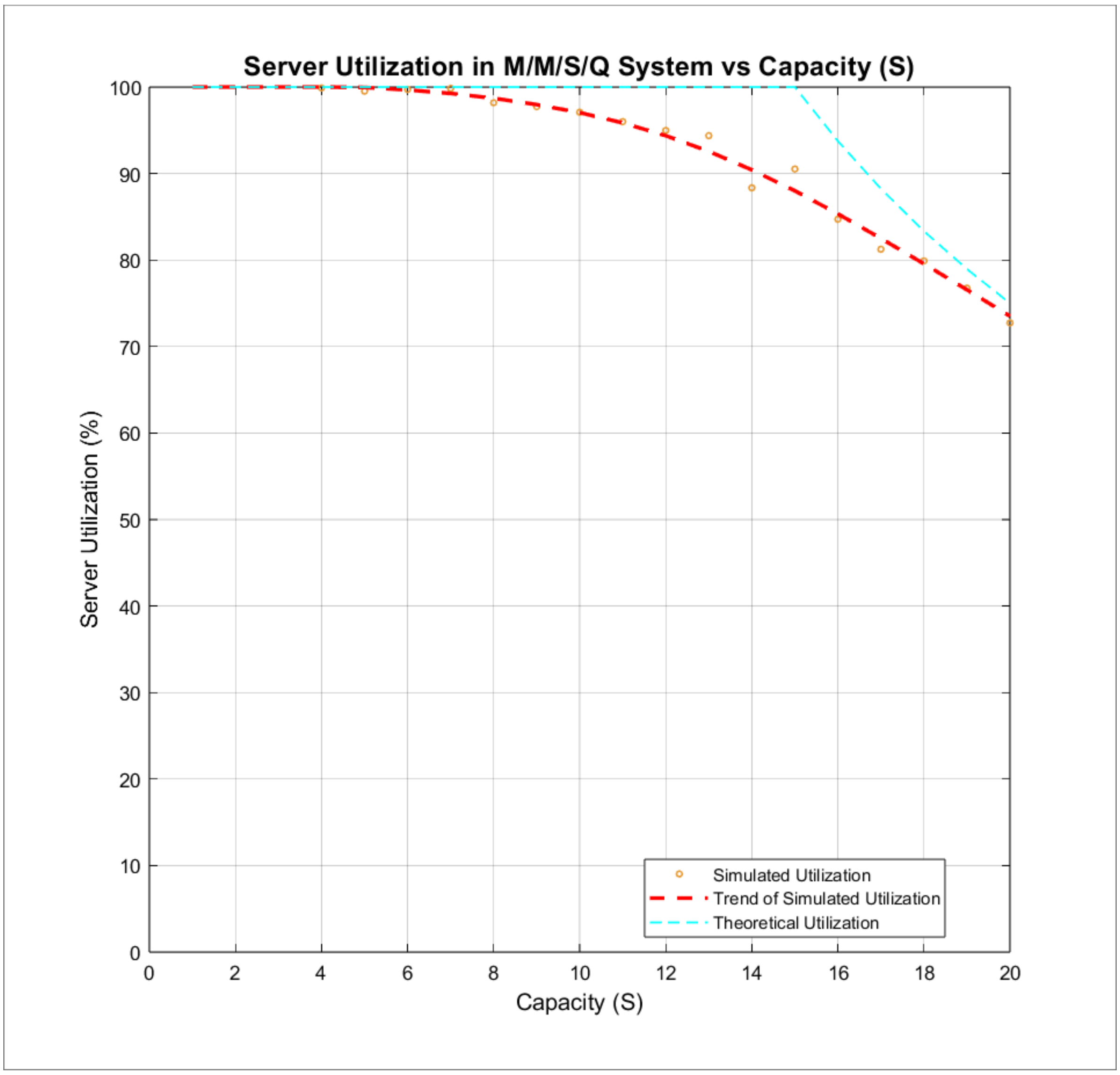
1. **Probability of Delay Across Server Capacities:** This plot shows the probability of call delay for different server capacities (S), based on fixed arrival rate (λ) and service rate (μ). The red dashed line represent the trend of simulated delay probabilities across server capacities. The green dashed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



2 pic. Probability of Delay Across Server Capacity and Probability of Loss Across Server Capacity Visualization

Server Utilization Across Server Capacity

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. Server Utilization Across Server Capacity Visualization

Source Code

The source code for the implementation of the M/M/S/Q 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/Q Queue Simulation Source Code.