Analysis of Steady-State Behavior in Server Queues using Markov Chains and Eigenvalues in the M/M/1 Model

Nicholas Wise Saragih Sumbayak - 13524037
Department of Informatics Engineering
School of Electrical Engineering and Informatics
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
nicholasaragih@gmail.com — 13524037@std.stei.itb.ac.id

***Abstract*—Queueing systems are fundamental to many computing environments where shared resources must handle randomly arriving workloads. One of the most widely used analytical models for such systems is the M/M/1 queue, which represents a single-server system with stochastic arrivals and service times. This paper analyzes the steady-state behavior of the M/M/1 queue using concepts from linear algebra and Markov chain theory.**

The system is modeled as a continuous-time Markov chain, where each state represents the number of jobs in the system and transitions are governed by arrival and service rates. The steady-state distribution is obtained by solving a system of linear balance equations derived from the generator matrix, highlighting the role of eigenvalues and null spaces in determining long-term system behavior. The stability condition for the queue is also examined and shown to depend on the relationship between the arrival and service rates.

To validate the analytical results, a discrete-event simulation of the M/M/1 queue is implemented. The simulation estimates steady-state probabilities and performance metrics, which are then compared with theoretical predictions. The results demonstrate strong agreement between the analytical model and simulation for stable systems, confirming the correctness of the continuous-time formulation.

***Index Terms*—M/M/1 queue, queueing theory, continuous-time Markov chain, steady-state analysis, generator matrix, eigenvalues, discrete-event simulation**

I. Introduction

Queueing behavior arises naturally in nearly every computing environment where resources are shared among multiple tasks. Whenever incoming work arrives faster than it can be immediately processed, the excess work must wait, forming a queue. This phenomenon appears in a broad range of systems, including CPU scheduling, packet forwarding in routers, job dispatching in cloud infrastructures, disk I/O scheduling, and networked application servers. The performance of these systems is heavily influenced by their queueing characteristics, making analytical models essential for understanding and improving real-world performance [1].

Modern computing workloads are highly variable and unpredictable. Task arrivals do not occur at fixed intervals, and service times fluctuate due to user behavior, network delays, resource contention, and software-level scheduling. These uncertainties make deterministic approaches nonoptimal. However, probabilistic theory provides methods to determine whether a server will remain stable under a particular load, estimate average waiting times, and understand how performance degrades as traffic increases.

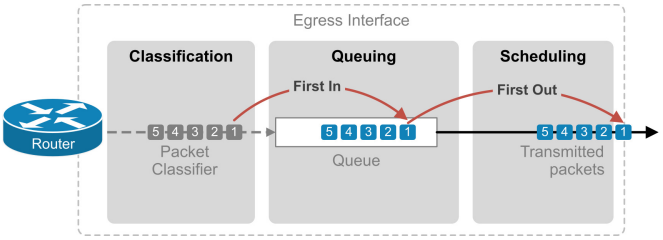

Fig. 1. A network router queue with the FIFO algorithm [2].

To formally analyze such behavior, queueing systems are commonly modeled as stochastic processes, with one of the simplest and most fundamental being the M/M/1 queue. In this model, arrivals follow a Poisson process with service times that follow an exponential distribution, all handled by a single server. Despite its simplicity, the model can capture trends and predict behaviors such as stability, queue buildup, and performance spikes [3].

The M/M/1 queue can be naturally represented as a Markov process, where each state corresponds to the number of jobs in the system. Due to the continuous nature of arrival and service events, the M/M/1 model is more accurately described as a continuous-time Markov chain (CTMC). In this framework, the system dynamics are governed by transition rates rather than discrete-time probabilities.

While steady-state behavior in discrete-time Markov chains is commonly characterized by eigenvectors associated with eigenvalue 1 of a stochastic transition matrix, continuous-time systems require a different formulation. In particular, the steady-state distribution of a CTMC is obtained by solving a system of linear balance equations derived from the generator matrix of the process. This distinction plays a critical role in correctly analyzing the long-term behavior of queueing systems.

II. Theoretical Foundation

A. Matrices

A matrix is a rectangular array of numbers, symbols, or expressions, arranged in rows and columns. The individual items in a matrix are called its elements or entries. The size of a matrix is described in terms of the number of rows and columns it contains. Generally, a general $m \times n$ matrix may be denoted as:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

with A denoting the matrix, m the number of rows, n the number of columns, and a_{ij} the element in row i and column j. A matrix with size $n \times n$ is called a square matrix of order n and the elements a_{ii} (where the row and column indices are equal) form the main diagonal of matrix A.

1) Row and Column Vectors: A matrix with only one row or one column is called a row matrix (or a row vector) or column matrix (or a column vector), respectively. A general $m \times 1$ column matrix and a $1 \times n$ row matrix b may be denoted as:

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} \text{ and } \mathbf{b} = \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix}$$

2) Matrix Addition and Subtraction: If A and B are matrices of the same size, their sum (A + B) and difference (A - B) are obtained by adding or subtracting their corresponding entries. Matrices of different sizes cannot be added or subtracted. In matrix notation, if $A = [a_{ij}]$ and $B = [b_{ij}]$ have the same size, then $(A \pm B)_{ij} = a_{ij} \pm b_{ij}$.

3) Scalar Multiplication: If A is any matrix and k is any scalar, then the scalar multiple kA is the matrix obtained by multiplying every entry of A by k. In matrix notation, if $A = [a_{ij}]$, then $(kA)_{ij} = ka_{ij}$.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \rightarrow 2A = \begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \end{bmatrix}$$

4) Matrix Multiplication: If A is an $m \times n$ matrix and B is an $n \times p$ matrix, then the product AB is defined to be the $m \times p$ matrix C whose entries are given by:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

In simpler terms, to find entry c_{ij} of the product matrix C, multiply the corresponding entries of the i^{th} row of matrix A with the j^{th} column of matrix B and add the results. Given the example below:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 0 \end{bmatrix}, B = \begin{bmatrix} 2 & 1 \\ 4 & 3 \\ 5 & 0 \end{bmatrix}$$

$$C = AB = \begin{bmatrix} 25 & 7 \\ 22 & 15 \end{bmatrix}$$

Since A is a 2×3 matrix and B is a 3×2 matrix, the resulting product AB is a 2×2 matrix. For example, to determine entry c_{11} of the product matrix AB, we multiply the corresponding entries of the first row of matrix A with the first column of matrix B and add the results.

5) Transpose of a Matrix: Given a matrix A of size $m \times n$, the transpose of A, denoted by A^T, is the $n \times m$ matrix obtained by interchanging the rows and columns of A. In matrix notation, if $B = A^T$, then the entries of B are defined as $b_{ij} = a_{ji}$, $1 \leq i \leq n$, $1 \leq j \leq m$.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \rightarrow A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

B. Eigenvalues and Eigenvectors

If A is an $n \times n$ matrix, then a nonzero vector **v** in R^n is called an eigenvector of A if there exists a scalar λ such that:

$$A\mathbf{v} = \lambda\mathbf{v}$$

The scalar λ is called the eigenvalue of A corresponding to the eigenvector **v**. In other words, multiplying the matrix A by the vector **v** results in a new vector that is a scalar multiple of the original vector **v**.

Given a matrix A with size $n \times n$, the eigenvalues and eigenvectors are found by solving the following characteristic equation:

$$\begin{gathered} Ax = \lambda x \\ IAx = \lambda Ix \\ Ax = \lambda Ix \\ (A - \lambda I)x = 0 \end{gathered}$$

Since $x = 0$ is the only trivial solution, for $(A - \lambda I)x = 0$ to have non-trivial solutions, the matrix $(A - \lambda I)$ must be singular, and therefore $det(A - \lambda I)$ must be zero. The polynomial given by $det(A - \lambda I) = 0$ is called the characteristic equation of A, and the solutions to such equation are the eigenvalues of A, otherwise denoted as the characteristic roots [5].

Geometrically, an eigenvector of a matrix A represents a direction in R^n that is preserved under the linear transformation defined by A. While most vectors are both rotated and scaled by a linear transformation, eigenvectors are only scaled by the corresponding eigenvalue. If $|\lambda| > 1$, the transformation stretches vectors in the direction of the eigenvector, if $|\lambda| < 1$, it contracts them, and if $\lambda < 0$, it reverses their direction.

C. Markov Chains

A Markov Chain is a mathematical system used to model systems that transition between different states over time. The defining characteristic of a Markov Chain is that the probability of transitioning to the next state depends only on the current state, and not on the sequence of states that preceded it. This property is known as the Markov property.

A Markov Chain is commonly represented using a matrix formed by transition probabilities between states. This matrix

is called the transition matrix, and the state vectors at successive time intervals are defined by

$$x(n+1) = Px(n),$$

where $x(n)$ is a probability vector describing the state distribution at time n, and P_{ij} is the probability that the system will be in state i at time $n+1$ given that it was in state j at time n [5].

The transition matrix P is a stochastic matrix, meaning that all of its entries are nonnegative and that the sum of each column is equal to one. These properties ensure that multiplying a probability vector by P produces another valid probability vector.

As an example, consider a system with three states. A possible transition matrix for this system is

$$P = \begin{bmatrix} 0.6 & 0.2 & 0.1 \\ 0.3 & 0.5 & 0.2 \\ 0.1 & 0.3 & 0.7 \end{bmatrix}.$$

Each column of this matrix sums to one, and each entry represents the probability of transitioning from one state to another in a single time step.

From a linear algebra perspective, the long-term behavior of a Markov Chain is determined by the eigenvalues and eigenvectors of its transition matrix. In particular, a steady-state distribution is a probability vector x that remains unchanged under the transition matrix, satisfying

$$Px = x.$$

This corresponds to an eigenvector associated with the eigenvalue $\lambda = 1$.

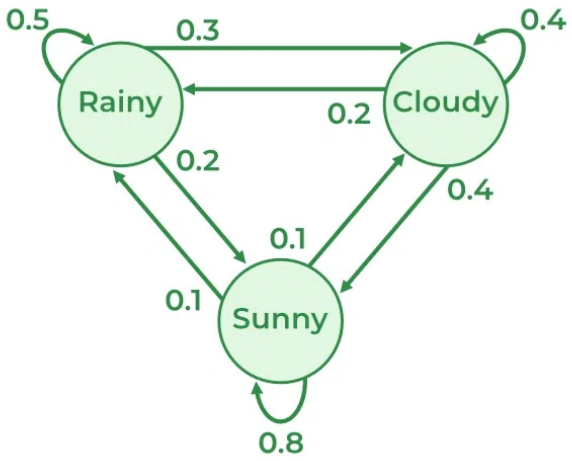

Fig. 2. Example of a Markov Chain representing weather states [6].

III. Methodology

A. The M/M/1 Queue Model

The $M/M/1$ queue is one of the simplest stochastic models used to represent server-based systems. The notation $M/M/1$ indicates that arrivals follow a (Markovian) Poisson process (M), service times are exponentially distributed (M), and the system consists of a single server (1).

Let λ denote the average arrival rate and μ the average service rate. At any time, the state of the system is defined as the number of jobs currently present in the system, including the job being served. Thus, the system is considered stable if $\lambda < \mu$ and if otherwise, the queue will grow indefinitely long over time.

Expanding over such, the utilization of the queue's buffer can be written as $\rho = \frac{\lambda}{\mu}$ and for it to be stable, it must hold that $\rho < 1$. The steady-state probability of having n jobs in the system is given by:

$$P_n = (1-\rho)\rho^n, \quad n = 0, 1, 2, \ldots$$

where P_n represents the probability of having n jobs in the system at steady state [3].

Fig. 3. An M/M/1 queueing node [7].

B. Markov Chain Representation

The M/M/1 queue is modeled as a continuous-time Markov chain (CTMC) with a countably infinite state space, where each state $n \geq 0$ represents the number of jobs currently present in the system. Transitions between states occur due to two independent stochastic events: arrivals and service completions.

An arrival causes the system to transition from state n to $n+1$ at rate λ, while a service completion causes a transition from state n to $n-1$ at rate μ for $n > 0$. This structure defines a birth–death process, a special class of CTMCs commonly used to model queueing systems [3].

The dynamics of a CTMC are characterized by an infinitesimal generator matrix Q, whose off-diagonal entries represent transition rates between states and whose diagonal entries ensure that each row sums to zero. For the M/M/1 queue, the generator matrix has the form:

$$Q = \begin{bmatrix} -\lambda & \lambda & 0 & 0 & \cdots \\ \mu & -(\lambda+\mu) & \lambda & 0 & \cdots \\ 0 & \mu & -(\lambda+\mu) & \lambda & \cdots \\ \vdots & & & & \end{bmatrix}.$$

For computational purposes, the infinite state space is truncated to a finite set $\{0, 1, \ldots, N\}$. This approximation allows matrix-based analysis while preserving the essential structure of the queueing process.

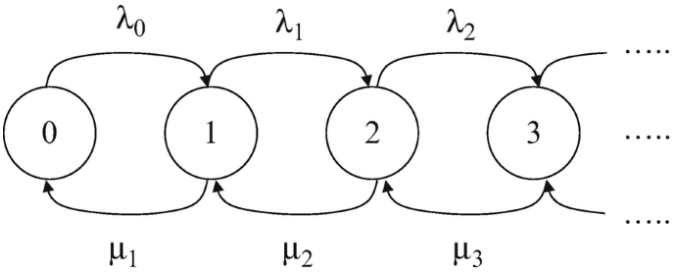

Fig. 4. A Markov Chain representation of a queue [8].

C. Steady-State Analysis via Eigenvalues and Linear Systems

The primary objective of the Markov Chain representation is to determine the long-term behavior of the queue. In steady-state operation, the probability distribution of queue lengths no longer changes over time, indicating that the system has reached statistical equilibrium.

For discrete-time Markov chains, a steady-state distribution is commonly defined as a probability vector x satisfying

$$Px = x,$$

corresponding to an eigenvector of the transition matrix associated with the eigenvalue $\lambda = 1$. Intuitively, this condition indicates that repeated applications of the transition matrix leave the distribution unchanged, so the system neither grows nor decays over time.

However, the M/M/1 queue is inherently a continuous-time Markov chain. In this setting, steady-state behavior is governed not by a stochastic transition matrix, but by the infinitesimal generator matrix Q. The steady-state distribution π is therefore defined as the solution to the global balance equations

$$\pi Q = 0, \quad \sum_{n=0}^{\infty} \pi_n = 1.$$

From a linear algebra perspective, this formulation corresponds to finding a normalized vector in the left nullspace of the generator matrix. The condition $\pi Q = 0$ ensures that, for each state, the total rate of probability flow into the state equals the total rate of probability flow out, resulting in a time-invariant distribution.

For the M/M/1 queue, these balance equations admit a valid solution only when the arrival rate satisfies $\lambda < \mu$. In this stable regime, the steady- state distribution is given by

$$\pi_n = (1 - \rho)\rho^n, \quad \rho = \frac{\lambda}{\mu}.$$

When $\lambda \geq \mu$, no normalizable solution exists, reflecting the fact that the queue grows without bound and the system fails to reach equilibrium [9].

Although eigenvalue analysis provides useful intuition for understanding stability in discrete-time systems, direct application of eigenvector methods to discretized approximations of continuous-time queues may yield inaccurate results. This distinction highlights the importance of selecting an analytical framework consistent with the underlying stochastic process.

D. C++ Queue Simulation

To complement the analytical steady-state analysis, a discrete-event simulation of the M/M/1 queue was implemented in C++. The purpose of this simulation is to estimate the steady-state distribution of queue lengths and to validate the results obtained from the continuous-time Markov chain model.

The simulation models the system as a continuous-time process driven by two random events: arrivals and service completions. Inter-arrival times are generated from an exponential distribution with rate λ, and service times follow an exponential distribution with rate μ. Events are processed in chronological order, and the system state is updated whenever an event occurs, allowing the queue to evolve naturally over continuous time.

During the simulation, the total amount of time spent in each state is recorded. After running the simulation for a sufficiently long duration T, the steady-state probability of being in state n is estimated by

$$\hat{P}_n = \frac{\text{total time spent in state } n}{T}.$$

For stable systems where $\lambda < \mu$, these estimates converge to the true steady-state distribution.

The simulator also computes performance measures such as the average number of jobs in the system, which are directly compared with theoretical results from queueing theory to evaluate accuracy.

The simulation was implemented in C++ due to its efficiency, precise control over random number generation, and native support from the Standard Template Library (STL) for queues. These properties make C++ well suited for simulations that require accurate timing and reproducible results. All outputs are exported in CSV format for further analysis.

Unlike eigenvector-based methods applied to discrete-time transition matrices, the discrete-event simulation directly follows the continuous-time behavior of the M/M/1 queue, making it a reliable numerical benchmark for validating the analytical model.

IV. Implementation

In simulating the M/M/1 queue, results may differ based on the system the program is running on. The following results were obtained on a system running Windows 11 with an Intel Core i7-13620H processor and 32 GB of DDR5 RAM.

A. QueueSimulator Class

```
class QueueSimulator {
private:
    double lambda, mu, rho;
    double currentTime;
    int currentQueueLength;

    std::priority_queue<Event,
        std::vector<Event>,
        std::greater<Event>> eventQueue;

    void scheduleArrival();
    void handleArrival();
```

```
    void handleDeparture();

public:
    QueueSimulator(double arrivalRate, double
        serviceRate);
    void simulate(double duration);
};
```

Listing 1. Core structure of the M/M/1 queue simulator

The simulation of the M/M/1 queue is encapsulated within the `QueueSimulator` class. The class maintains the current state of the system, including the arrival rate (λ), service rate (μ), utilization (ρ), current time, and current queue length. Arrival and departure events are managed using a time-ordered priority queue.

B. ExperimentRunner Class

```
struct ExperimentResult {
    double lambda;
    double rho;
    double avgQueueLengthSim;
    double avgQueueLengthTheory;
};

class ExperimentRunner {
private:
    double mu;
    double simTime;

public:
    ExperimentRunner(double serviceRate, double
        simTime);
    void runVaryingLambdaExperiments();
};
```

Listing 2. Structure of the ExperimentRunner class

To observe the system's behavior under varying load conditions, the `ExperimentRunner` class is designed to execute multiple simulation runs with varying arrival rates (λ) as shown in Listing 2.

```
class MarkovChainAnalysis {
private:
    int numStates;
    double lambda, mu;
    std::vector<std::vector<double>> generatorMatrix
        ;

    void buildGeneratorMatrix();

public:
    MarkovChainAnalysis(int states,
                        double arrivalRate,
                        double serviceRate);

    std::vector<double> computeSteadyState();
};
```

Listing 3. Structure of CMTC Analyzer

To solve the steady-state distribution analytically, the `MarkovChainAnalysis` class constructs the generator matrix for a truncated M/M/1 queue and computes the steady-state distribution by solving the corresponding linear system as shown in Listing 3.

Algorithm 1 Simulation Driver for M/M/1 Queue

```
Read input parameters λ, μ, T
Compute utilization ρ = λ/μ
if ρ < 1 then
    Run discrete-event simulation for duration T
    Compute steady-state statistics
    Compare simulation results with theory
else
    Run finite-horizon simulation
    Observe queue growth behavior
end if
Perform experiments for multiple λ values
Export results for analysis
```

C. Simulation

With the defined classes, the simulation driver orchestrates the overall process. The main flow of the simulation is outlined in Algorithm 1.

The experimental driver runs three types of simulations to infer the behavior of the M/M/1 queue under different parameters, which in this case refers to varying the arrival rate λ while keeping the service rate μ constant. Those experiments are:

- A discrete-event simulation of the M/M/1 queue based on the input parameters.
- Evaluation and comparison of the steady-state statistics obtained from the simulation against the theoretical results derived from queueing theory.
- Comparative experiments across different traffic intensities.

V. Results and Discussion

This section presents the results obtained from the discrete-event simulation of an M/M/1 queue and compares them with analytical predictions derived from continuous-time Markov chain (CTMC) theory. The analysis focuses on three key aspects: validation of steady-state probabilities for stable systems, the impact of utilization on average queue length, and simulation behavior near the stability boundary.

A. Steady-State Validation for Stable Systems

The simulation was first conducted with parameters $\lambda = 0.8$ and $\mu = 1.0$, corresponding to a utilization factor of $\rho = 0.8$. The system was simulated for $T = 100{,}000$ time units to ensure convergence to steady-state behavior.

TABLE I
Steady-State Probabilities: Simulation vs Theory ($\rho = 0.8$)

State n	Simulated	Theoretical	Rel. Error
0	0.1964	0.2000	1.82%
1	0.1605	0.1600	0.34%
2	0.1301	0.1280	1.60%
3	0.1021	0.1024	0.29%
4	0.0830	0.0819	1.32%

Table I compares the simulated steady-state probabilities with the theoretical distribution $P_n = (1-\rho)\rho^n$. All relative errors remain below 2%, indicating strong agreement between simulation and theory.

The average queue length further validates this result. The simulation yields $L_{\text{sim}} = 4.0355$, while the theoretical value is $L_{\text{theory}} = \rho/(1-\rho) = 4.0$. The relative error of 0.89% demonstrates that the simulation accurately captures steady-state performance for a stable M/M/1 system.

B. Impact of Utilization on System Performance

To evaluate the effect of increasing utilization, the simulation was repeated for multiple values of ρ, covering both stable ($\rho < 1$) and unstable ($\rho \geq 1$) regimes.

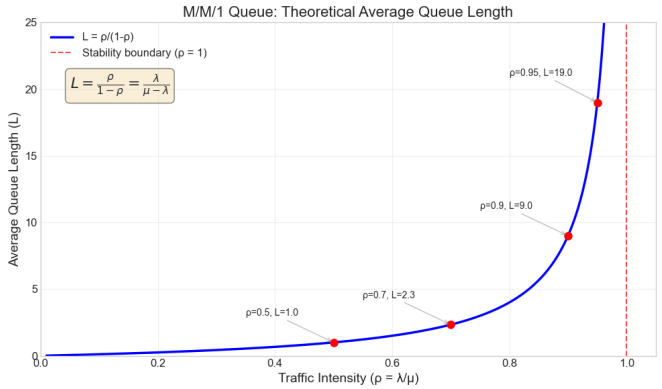

Fig. 5. System performance comparison across different utilization levels

As shown in Figure 5, simulated results closely match theoretical predictions for all stable configurations. The average queue length increases rapidly as ρ approaches 1, consistent with the hyperbolic relationship $L = \rho/(1-\rho)$. This highlights the severe performance degradation that occurs near system saturation.

For $\rho \geq 1$, the system becomes unstable and no steady-state distribution exists. The observed average queue lengths in these cases depend on the finite simulation horizon rather than a theoretical equilibrium, confirming unbounded queue growth when the arrival rate equals or exceeds the service rate.

C. Accuracy Degradation Near the Stability Boundary

To examine simulation accuracy near the stability boundary, additional experiments were conducted with $\mu = 1.0$ and varying λ values approaching 1.0.

Table II demonstrates that simulation accuracy deteriorates significantly as $\rho \to 1^-$. While errors remain below 2% for $\rho \leq 0.9$, they increase sharply at higher utilization levels. This occurs because the theoretical average queue length diverges as $\rho \to 1$, requiring increasingly long simulation horizons for accurate estimation. At $\rho = 0.99$, the theoretical value of $L = 99$ is not fully observed within the finite simulation time, resulting in substantial underestimation.

TABLE II
SIMULATION ACCURACY FOR VARIOUS ARRIVAL RATES ($\mu = 1.0$)

λ	ρ	**Sim. L**	**Theory L**	**Error**
0.30	0.30	0.4290	0.4286	0.10%
0.50	0.50	0.9825	1.0000	1.75%
0.70	0.70	2.3252	2.3333	0.35%
0.80	0.80	4.0355	4.0000	0.89%
0.90	0.90	8.8898	9.0000	1.22%
0.95	0.95	15.6963	19.0000	17.39%
0.99	0.99	34.9768	99.0000	64.67%

D. Discussion

Taken together, the results demonstrate strong consistency between CTMC-based analytical models and discrete-event simulation for the M/M/1 queue. The close agreement observed in stable regimes confirms the correctness of the generator-matrix formulation and validates the steady-state solution obtained from the balance equations.

The minimal margin of error between simulation and theory for stable systems validates the steady-state distribution $P_n = (1-\rho)\rho^n$ and the average queue length formula $L = \rho/(1-\rho)$ as seen from Table I and Figure 5.

The sharp transition between stable and unstable behavior emphasizes the critical importance of maintaining $\rho < 1$. Operating near saturation leads to disproportionate increases in queue length and delay. In practical systems, this necessitates careful capacity planning and load management to avoid performance collapse.

Finally, these results highlight an important conceptual distinction emphasized throughout this work: although discrete-time Markov chains rely on eigenvectors associated with eigenvalue 1, steady-state analysis for continuous-time systems arises from the null space of the generator matrix. The successful alignment between theory and simulation confirms that adopting a CTMC-consistent framework is essential for accurately modeling real-world queueing systems.

VI. CONCLUSION

This paper analyzed the steady-state behavior of the M/M/1 queue using continuous-time Markov chain (CTMC) theory and linear algebra concepts. By formulating the queue as a birth–death process and deriving its generator matrix, the steady-state distribution was obtained through the balance equations and interpreted as a normalized vector in the left null space of the generator matrix. The analytical results were validated through a discrete-event simulation implemented in `C++`, which demonstrated strong agreement with theoretical predictions for stable systems.

Simulation results confirmed that when the utilization factor satisfies $\rho < 1$, both the steady-state probabilities and average queue length closely match the closed-form expressions derived from queueing theory. As utilization approaches unity, performance degrades rapidly and simulation accuracy deteriorates due to the divergence of the theoretical mean queue length. When $\rho \geq 1$, the absence of a steady-state distribution was clearly observed through unbounded queue

growth, reinforcing the fundamental stability condition of the M/M/1 model.

Despite its analytical tractability and simplicity, the M/M/1 queue represents an abstraction of real-world systems that captures straightforward behavior changes based on queue utilization and load. The assumption of a single server limits its ability to fully capture the behavior of modern computing infrastructures, which frequently employ multiple parallel servers, load balancing, and heterogeneous service rates. Models such as M/M/c queues and networks of queues provide more accurate representations of contemporary systems but require more complex analytical and numerical techniques.

APPENDIX

GitHub Repository: https://github.com/nicholaswisee/makalah-algeo

Youtube Video: https://youtube.com/shorts/XaIVOJ5UMQQ

ACKNOWLEDGMENT

First and foremost, the author would like to express his profound gratitude to God Almighty for His blessings, strength, and guidance throughout the completion of this research paper. The author also wishes to convey his deepest appreciation to his dear family for their unwavering love, endless prayers, and constant encouragement, which serve as the primary motivation in his academic journey.

Furthermore, the author extends his gratitude to Dr. Rinaldi Munir and the teaching staff of the IF2123 Linear and Geometrical Algebra course at Institut Teknologi Bandung for their invaluable lessons and knowledge bestowed upon the author. The author also wishes to thank their colleagues in the Informatics Engineering department for their support and collaboration throughout the semester.

Lastly, the utmost gratitude is extended to fellow Istana Dago Townhouse Unit 8 frequent-stayers for their companionship and support during the course of the semester, including the conception and writing of this paper. Their presence has made the author's journey throughout the semester much more bearable.

Soli Deo Gloria!

REFERENCES

[1] M. Harchol-Balter, *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge, U.K.: Cambridge Univ. Press, 2013.

[2] "Queuing and Scheduling," NetworkAcademy.IO, CCNA Network Services. [Online]. Available: https://www.networkacademy.io/ccna/network-services/queuing-and-scheduling. [Accessed: Dec. 20, 2025].

[3] L. Kleinrock, *Queueing Systems, Volume 1: Theory*. New York, NY, USA: Wiley, 1975.

[4] Munir, Rinaldi. "Nilai Eigen dan Vektor Eigen (Bagian 1)", School of Electrical Engineering and Informatics (STEI) ITB, 2025. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-19-Nilai-Eigen-dan-Vektor-Eigen-Bagian1-2025.pdf

[5] H. Anton and C. Rorres, *Elementary Linear Algebra: Applications Version*, 11th ed. Hoboken, NJ, USA: Wiley, 2013.

[6] "Markov Chains in NLP," GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/nlp/markov-chains-in-nlp/. [Accessed: Dec. 15, 2025].

[7] "M/M/1 queue," Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/M/M/1_queue. [Accessed: Dec. 16, 2025].

[8] G. Giambene, "Markov chains and queuing theory," in *Queuing Theory and Telecommunications*, Textbooks in Telecommunication Engineering, Cham, Switzerland: Springer, 2021, doi: 10.1007/978-3-030-75973-5_4.

[9] M. Harchol-Balter, *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge, U.K.: Cambridge Univ. Press, 2013.

STATEMENT

In this statement, I declare that this paper I have written is my own work, not a duplication or translation of someone else's paper, and is not plagiarized.

Bandung, 24 December 2025

Nicholas Wise Saragih Sumbayak
13524037