# 摘要

本研究探討一個以區塊鏈為基礎的排隊系統，該系統在不同的使用者優先權與無耐性組合下，建構區塊鏈交易過程的數學模型，並引入開/關（ON/OFF）機制，以模擬區塊生成與共識階段的隨機可用性。系統由兩個串接的佇列構成：顧客佇列，客戶在此等待被打包進區塊；區塊佇列，已打包的客戶在此等待共識處理。系統採用部分批次服務機制進行區塊生成，其中允許不大於一最大數量的客戶可打包成一個區塊。在 OFF 狀態期間，區塊生成與共識作業會暫停，然而顧客到達仍持續進行，且若顧客佇列尚有容量，則新到達的客戶仍可進入排隊。

為深入描繪系統的行為與結構特性，本研究設計四種情境模型：（1）不具有無耐性的單一類型客戶、（2）不具有無耐性的雙類型客戶、（3）具有無耐性的單一類型客戶、（4）具有無耐性的雙類型客戶。針對每一情境，我們建構對應的多維馬可夫鏈來描述系統狀態，推導平衡方程並透過疊代方式求得穩態機率分布，進而計算多項效能指標，包括成功送達率、阻塞機率與平均等待時間。此外，本研究也探討不同參數對系統效能的影響。

為驗證解析結果，我們以 C++ 語言實現離散事件模擬，以忠實重現各情境中的事件邏輯與服務規則。模擬結果不僅驗證了解析結果的準確性，也揭示多項值得注意的系統行為，包括非搶佔優先權排程中公平性與效率的權衡、無耐性機制帶來的效能提升，以及部分批次機制與 ON/OFF 動態對整體系統表現的影響。

**關鍵字：區塊鏈、非搶佔優先權、無耐性、區塊生成、共識、部分批量服務、開/關機制**

# Abstract

This research investigates a blockchain-based queuing system that models the transaction process under varying combinations of customer priority and impatience, while incorporating ON/OFF mechanism to reflect the stochastic availability of block generation and the consensus phases. The system consists of two sequential queues: a customer queue, where customers wait to be grouped into a block, and a block queue, where grouped customers await consensus. A partial batch service mechanism is employed for block generation, allowing up to a maximum number of customers to be batched into a block. During OFF periods, block generation and consensus operations are suspended, although customer arrivals continue and are admitted into the customer queue if queue capacity is available.

To capture the behavioral and structural characteristics of the system, four distinct scenarios are modeled: (1) single-class customers without impatience, (2) two-class customers without impatience, (3) single-class customers with impatience, and (4) two-class customers with impatience. For each scenario, a multi-dimensional Markov chain is constructed to describe the system state. Balance equations are derived and solved iteratively to obtain the steady-state distribution, from which key performance metrics, including throughput, blocking probability, and average waiting times, are calculated. The effects of various parameters on the performance metrics are explored.

To validate the analytical results, a discrete-event simulation is implemented in C++, faithfully replicating the event logic and service rules of each scenario. The simulation results confirm the accuracy of analytical results and highlight several notable system behaviors, including the trade-off between fairness and efficiency in non-preemptive priority settings, the performance benefits introduced by impatience mechanisms, and the system-level effects of partial batch size and ON/OFF dynamics.

**Keywords: blockchain, non-preemptive priority, impatience, block generation, consensus, partial batch service, ON/OFF mechanism**

# Contents

# List of Tables



# List of Figures

# 1. Introduction

Blockchain technology, first proposed by Nakamoto in 2008, has rapidly evolved into a foundational infrastructure for digital trust and decentralized systems. As a form of distributed ledger technology (DLT), blockchain is characterized by decentralization, transparency, and immutability, making it particularly suited for secure data storage and peer-to-peer value transfer. Each block contains a hash of the previous block, a timestamp, and a collection of transactions, forming an append-only chain that is nearly impossible to tamper with. Originally designed to support cryptocurrencies like Bitcoin, blockchain has since been adopted in numerous domains including finance, healthcare, media, logistics, and energy.

One of the most pressing technical challenges in today's blockchain landscape is interoperability, that is the ability for independently operated blockchain networks to communicate and exchange assets or information. This challenge has spurred the development of cross-chain bridges, which act not as physical connections but rather as a collection of protocols and mechanisms that allow heterogeneous blockchains to interoperate. These bridges enable asset transfers, data exchange, and coordinated smart contract execution between public chains, consortium chains, and private chains, which are otherwise isolated due to architectural and consensus differences. Similar concerns have also emerged in layered edge-cloud systems, where performance evaluation frameworks are often built on queueing-theoretic modeling.

To address the performance analysis of blockchain systems, a number of prior works have employed queueing theory to model system dynamics under realistic assumptions. For instance, researchers have simulated edge-cloud offloading networks using M/G/1 and M/G/m models to assess task delays and system throughput in blockchain-based layered environments [1]. Another study applied M/M/n/L queues to model transaction processing and block generation in Bitcoin, demonstrating how queue length and block production rates impact performance [2].

In more structured systems such as Hyperledger Fabric, a queueing network model was proposed to divide the consensus process into execution, ordering, and validation stages, enabling analysis of latency across phases [3]. Other researchers combined queueing models with multidimensional Markov chains to analyze PBFT-based consensus systems with repairable voting nodes, quantifying system reliability and throughput under dynamic conditions [4]. Performance bottlenecks in Fabric's architecture were also identified via benchmarking, helping guide practical

optimizations [5].

Beyond consensus mechanics, theoretical models have captured the growth dynamics and reward allocation strategies in multi-mining pool environments such as Ethereum. One study introduced a tree-based blockchain structure and renewal reward theory to model stale and uncle blocks [6]. Priority-based transaction handling has also been modeled using non-preemptive limited-priority queues, illustrating the performance tradeoffs between high- and low-priority transaction classes [7]. To tackle intractable steady-state distributions in complex systems, another approach applied the maximum entropy principle to estimate probabilities based on observable statistics, providing flexible approximations without strong distributional assumptions [8].

In response to the lack of simple yet effective models for analyzing cross-chain systems, this thesis draws on examples from [9] and [10] to develop a queueing model for cross-chain transaction flows. The model abstracts the system into two sequential queues: the customer queue, where transactions wait to be selected for block formation, and the block queue, where blocks undergo consensus. To better capture realistic customer behavior, the model considers multiple user classes with non-preemptive limited priority and partial batch service, as well as system states that alternate between ON and OFF. User impatience is also incorporated to reflect transaction abandonment in highly congested environments. These dynamics are analyzed across four scenarios: (1) single-class customers without impatience, (2) two-class customers without impatience, (3) single-class customers with impatience, and (4) two-class customers with impatience.

To evaluate the system's steady-state behavior under complex configurations, this thesis adopts a numerical iteration method based on the balance equations of the underlying Markov chain. A simulation is also performed for validation. This approach enables the computation of key performance metrics.

The remainder of this thesis is organized as follows. Chapter 2 introduces the system model, detailing the cross-chain process structure and the queuing assumptions used in this study. Chapter 3 presents the analytical model, which formalizes the system behavior under various parameter settings and derives key performance metrics. Chapter 4 describes the simulation model, providing implementation details and simulation strategies used to validate the analytical results. Chapter 5 reports the numerical results and performance evaluation across the proposed scenarios. Finally, Chapter 6 concludes the thesis and outlines potential directions for future work.

# 2. System Model

We aim to study four blockchain scenarios, each involving two queues of finite capacity: the customer queue and the block queue. The maximum capacity of the customer queue is denoted by $N$, while the block queue, which represents the stage where users participate the consensus after being grouped into a block, has a capacity of $b$. Customers in the system first wait in the customer queue for the block generation process. Once this process is complete, a group of customers is moved to the block queue to undergo the consensus process.

The blocking generation process is based on a partial batch service mechanism. When the block queue becomes idle, if there are more than $b$ customers in the customer queue, the first $b$ customers are selected and moved to the block queue. If there are $b$ or fewer customers waiting, all of them are transferred instead. After the consensus process finishes, regardless of whether the result is successful, all customers in the block queue leave the system. The performance of the system depends on the balance between the effective service rate ($\mu_{eff}$) and the effective arrival rate ($\lambda_{eff}$).

Additionally, the system may switch between ON and OFF periods. During the OFF period, caused by events such as hacking attacks or connection failures due to environmental factors, both the block generation and consensus processes are suspended. Once the system returns to the ON period, these processes resume as usual.

## 2.1. Scenario 1: Single-Class Customers without Impatience

In the first scenario, we assume that there is only a single class of customers in the system, and the queueing discipline for the customer queue is First-Come-First-Served (FCFS). It is noted that if the block queue is empty, at most $N$ customers can wait in the customer queue for the block generation process. On the other hand, if the block queue is not empty, the maximum number of customers allowed in the customer queue is reduced to $N - b$.

## 2.2. Scenario 2: Two-Class Customers without Impatience

In the second scenario, we assume that there are two classes of customers in the system: high-priority customers and low-priority customers. Customers with the same priority are served according to the First-Come-First-Served (FCFS) discipline. Note that for high-priority customers are always placed ahead of low-priority customers in the customer queue. Customers with different priorities are served according to the non-

preemptive priority discipline. Specifically, high-priority customers are always placed ahead of low-priority customers in the customer queue and the consensus process of the low-priority customers cannot be interrupted. Note that for high-priority customers, the maximum capacity of the customer queue is $N$ when the block queue is idle, and $N - b$ when it is not idle. On the other hand, for low-priority customers, the maximum capacity of the customer queue is always $N - b$, regardless of whether the block queue is idle or not.

## 2.3. Scenario 3: Single-Class Customers with Impatience

The third scenario considers a single class of customers with impatience. Customers still follow the First-Come-First-Served (FCFS) discipline, but they may leave the system while waiting in the customer queue if their waiting time exceeds their patience threshold. Once a customer enters the block queue, impatience is no longer considered. It is noted that if the block queue is empty, at most $N$ customers can wait in the customer queue for the block generation process. On the other hand, if the block queue is not empty, the maximum number of customers allowed in the customer queue is reduced to $N - b$.

## 2.4. Scenario 4: Two-Class Customers with Impatience

In the fourth scenario, we again consider two classes of customers with impatience —high-priority and low-priority. Customers of the same priority follow the First-Come-First-Served (FCFS) discipline, and customers of different priorities follow the non-preemptive discipline, i.e., high-priority customers are given precedence over low-priority customers in the queue and the consensus process of the low-priority customers cannot be interrupted. Customers from both priority classes may leave the queue if they wait too long. Each priority class may have its own impatience rate. Impatience is no longer relevant once customers enter the block queue. Note that for high-priority customers, the maximum capacity of the customer queue is $N$ when the block queue is idle, and $N - b$ when it is not idle. On the other hand, for low-priority customers, the maximum capacity of the customer queue is always $N - b$, regardless of whether the block queue is idle or not.

# 3. Analytical Model

In this chapter, we are going to present four different scenarios for modeling blockchain-based systems: (1) single-class customers without impatience, (2) two-class customers without impatience, (3) single-class customers with impatience, and (4) two-class customers with impatience. Each of these scenarios is built upon a queuing-based abstraction of the blockchain process and aims to capture distinct behavioral features related to customer priority and abandonment. In all cases, as shown in Figure 3-1, the system is composed of two queues with limited capacity: the customer queue, which temporarily holds customers before block generation, and the block queue, which represents the stage where customers participate in the consensus protocol after being grouped into a block.

Assume that the arrivals of customers follow a Poisson process, where the arrival rate is denoted by λ. In the multi-class scenarios, we further distinguish between high-priority and low-priority customers, whose respective arrival rates are $\lambda_H$ and $\lambda_L$, so that the total arrival rate satisfies $\lambda_{total} = \lambda_H + \lambda_L$. After arriving at the customer queue, customers wait for the block generation process, which occurs at a rate of $\mu_1$ (or $\mu_{1_H}$and $\mu_{1_L}$in the two-class case). Once a block is formed, a group of customers is transferred to the block queue, where the consensus process is carried out at a service rate denoted by $\mu_2$ (or $\mu_{2_H}$ and $\mu_{2_L}$depending on customer class).

In scenarios that involve impatience, we assume that customers may abandon the system while waiting in the customer queue if their waiting time exceeds a certain threshold. The impatience threshold is modeled as an exponential random variable with a rate $\gamma$ for single-class customers, and rates $\gamma_H$ and $\gamma_L$ for high-priority and low-priority customers, respectively. Once a customer enters the block queue, impatience is no longer considered. In addition, we consider the operational reliability of the system by incorporating the possibility of the system state alternating between ON and OFF periods. During ON periods, both block generation and consensus operations are allowed to proceed, while during OFF periods, these operations are suspended. The durations of both ON and OFF periods are exponentially distributed. The transition rates between the two states are given by $\alpha$ (ON to OFF) and $\beta$ (OFF to ON) respectively.

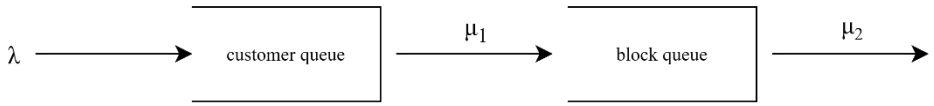


Figure 3-1: System diagram

We assume the queueing discipline is First-Come-First-Served (FCFS) for customers of the same class. In the two-class scenarios, customers are additionally scheduled under a non-preemptive priority rule, in which high-priority customers are placed ahead of low-priority ones in the customer queue, but once a customer enters the block queue, their service cannot be interrupted. These settings allow us to examine the interplay between system structure, service prioritization, impatience-driven abandonment, and queue dynamics in a blockchain-inspired environment. The parameters used in different scenarios are shown in Table 3.1

| Description | Single-class | Two-class |
|---|---|---|
| Arrival rate | $\lambda$ | $\lambda_H$ |
| | | $\lambda_L$ |
| Block generation rate | $\mu_1$ | $\mu_{1_H}$ |
| | | $\mu_{1_L}$ |
| Consensus rate | $\mu_2$ | $\mu_{2_H}$ |
| | | $\mu_{2_L}$ |
| Impatient rate | $\gamma$ | $\gamma_H$ |
| | | $\gamma_L$ |
| Transition rate (ON to OFF) | $\alpha$ | $\alpha$ |
| Transition rate (OFF to ON) | $\beta$ | $\beta$ |

Table 3-1 The parameters used in different scenarios

## 3.1. Scenario 1: Single-Class Customer without Impatience

In this scenario, we consider a single-class customer system without impatience, where arrivals follow a Poisson process and customers are served based on the First-Come-First-Served (FCFS) discipline. The service process is divided into block generation and consensus phases, and the system switches between ON and OFF states, affecting service availability.

Assume that the arrivals of customers follow a Poisson process, where the arrival rate is denoted by λ. After arriving at the customer queue, customers wait for the block

generation process, which occurs at a rate of $\mu_1$. Each block is generated according to the partial batch policy, i.e., each block can contain 1 to $b$ customers. Once a block is formed, a group of customers is transferred to the block queue, where the consensus process is carried out at a service rate denoted by $\mu_2$.

In addition, we consider the operational reliability of the system by incorporating the possibility of the system state alternating between ON and OFF periods. During ON periods, both block generation and consensus operations are allowed to proceed, while during OFF periods, these operations are suspended. The durations of both ON and OFF periods are exponentially distributed. The transition rates from ON to OFF and from OFF to ON are given by $\alpha$ and $\beta$, respectively.

The $\lambda_{eff}$ and $\mu_{eff}$ of this scenario is:

$$\lambda_{eff} = \lambda(1 - P_b) \tag{3-1}$$

$$\mu_{eff} = b\left(\frac{1}{\mu_1} + \frac{1}{\mu_2}\right)^{-1} \frac{\beta}{\alpha + \beta} \tag{3-2}$$

### 3.1.1. State Balance Equations

The system under consideration is described as a three-dimensional Markov chain with state denoted by $(i, x, k)$, where $i$ denotes the number of customers in the customer queue, $x$ denotes the number of customers in the block queue, and $k$ denotes the system state. When $x = 0$, the maximum number of customers in the customer queue is $N$. When $1 \le x \le b$, meaning that the block queue is occupied, the maximum number of customers allowed in the customer queue is reduced to $N - b$. The system state $k = 1$ indicates that the system is in the ON state, where customers can enter the customer queue and both block generation and consensus operations can proceed. On the other hand, when $k = 0$, the system is in the OFF state, during which only customer arrivals to the queue are permitted, while block generation and consensus are suspended. The state space can be denoted as follows:

$$S = \left\{(i, x, k) \middle| \begin{Bmatrix} 0 \le k \le 1 \\ if\ x = 0 : 0 \le i \le N \\ if\ 1 < x \le b : 0 \le i \le N - b \end{Bmatrix}\right\} \tag{3-3}$$

Hence, the total number of feasible states is given by:

$$|S| = 2(N + 1) + 2b(N - b + 1) \tag{3-4}$$

For example, if $N = 50$ and $b = 15$, the number of feasible states is 1182. The steady state probability of state $(i, x, k)$ is denoted as $\pi_{i,x,k}$. In this scenario, the feasible states can be categorized into 16 distinct cases, as described below.

(a) System off, $k = 0$

Case 1: $\mathrm{i} = 0, 0 \leq \mathrm{x} \leq \mathrm{b}$

$$(\beta + \lambda)\pi_{0,x,0} = \alpha\pi_{0,x,1}$$

Case 2: $1 \leq i \leq N - b - 1, 0 \leq x \leq b$

$$(\beta + \lambda)\pi_{i,x,0} = \alpha\pi_{i,x,1} + \lambda\pi_{i-1,x,0}$$

Case 3: $i = N - b, x = 0$

$$(\beta + \lambda)\pi_{N-b,0,0} = \alpha\pi_{N-b,0,1} + \lambda\pi_{N-b-1,0,0}$$

Case 4: $i = N - b, 1 \leq x \leq b$

$$\beta\pi_{N-b,x,0} = \alpha\pi_{N-b,x,1} + \lambda\pi_{N-b-1,x,0}$$

Case 5: $N - b + 1 \leq i \leq N - 1, x = 0$

$$(\beta + \lambda)\pi_{i,0,0} = \alpha\pi_{i,0,1} + \lambda\pi_{i-1,0,0}$$

Case 6: $i = N, x = 0$

$$\beta\pi_{N,0,0} = \alpha\pi_{N,0,1} + \lambda\pi_{N-1,0,0}$$

(b) $System\ on, k = 1$

Case 7: $i = 0, x = 0$

$$(\alpha + \lambda)\pi_{0,0,1} = \beta\pi_{0,0,0} + \sum_{t=1}^{b} \mu_2\pi_{0,t,1}$$

Case 8: $i = 0, 1 \leq x \leq b$

$$(\alpha + \lambda + \mu_2)\pi_{0,x,1} = \beta\pi_{0,x,0} + \mu_1\pi_{x,0,1}$$

Case 9: $1 \leq i \leq N - b - 1, x = 0$

$$(\alpha+\lambda+\mu_1)\pi_{i,0,1}=\beta\pi_{i,0,0}+\lambda\pi_{i-1,0,1}+\sum_{t=1}^{b}\mu_2\pi_{i,t,1}$$

Case 10: $1\le i\le N-b-1, 1\le x\le b-1$

$$(\alpha+\lambda+\mu_2)\pi_{i,x,1}=\beta\pi_{i,x,0}+\lambda\pi_{i-1,x,1}$$

Case 11: $1\le i\le N-b-1, x=b$

$$(\alpha+\lambda+\mu_2)\pi_{i,b,1}=\beta\pi_{i,b,0}+\lambda\pi_{i-1,b,1}+\mu_1\pi_{i+b,0,1}$$

Case 12: $i=N-b, x=0$

$$(\alpha+\lambda+\mu_1)\pi_{N-b,0,1}=\beta\pi_{N-b,0,0}+\lambda\pi_{N-b-1,0,1}+\sum_{t=1}^{b}\mu_2\pi_{N-b,t,1}$$

Case 13: $i=N-b, 1\le x\le b-1$

$$(\alpha+\mu_2)\pi_{N-b,x,1}=\beta\pi_{N-b,x,0}+\lambda\pi_{N-b-1,x,1}$$

Case 14: $i=N-b, x=b$

$$(\alpha+\mu_2)\pi_{N-b,b,1}=\beta\pi_{N-b,b,0}+\lambda\pi_{N-b-1,b,1}+\mu_1\pi_{N,0,1}$$

Case 15: $N-b+1\le i\le N-1, x=0$

$$(\alpha+\lambda+\mu_1)\pi_{i,0,1}=\beta\pi_{i,0,0}+\lambda\pi_{i-1,0,1}$$

Case 16: $i=N, x=0$

$$(\alpha+\mu_1)\pi_{N,0,1}=\beta\pi_{N,0,0}+\lambda\pi_{N-1,0,1}$$

Given the large number of equations presented above, it is impractical to illustrate all the corresponding state transition diagrams. Therefore, we focus on a relatively complex case, specifically Case 11, as a representative example, shown in Figure 3-2.

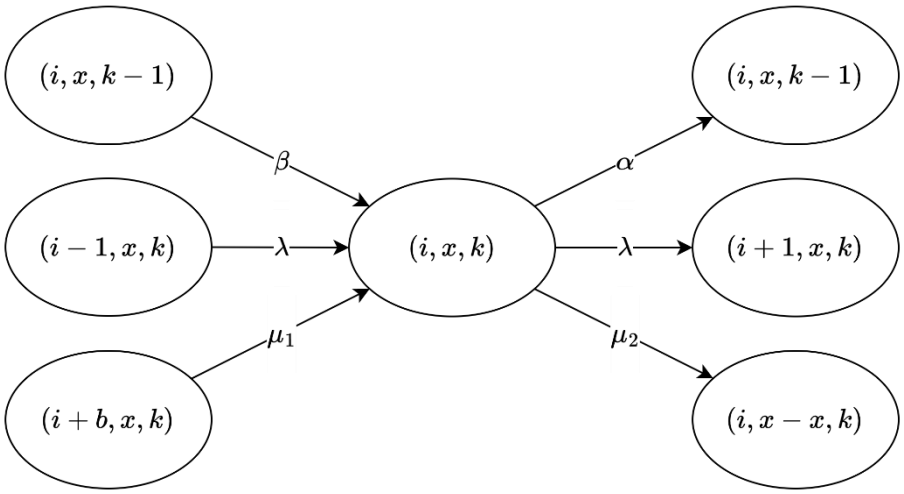


Figure 3-2: The state transition diagram of Case 11: $1 \leq i \leq N-b-1, x = b, k = 1$

### 3.1.2. Iterative Algorithm

We use the iterative algorithm provided below, and perform calculations on the state balance equations until they converge, allowing us to determine the steady-state distribution of the system.

#### Iterative algorithm:

**Step 1**: Initialize $\pi(i,x,k)^{old} = \frac{1}{|s|}$ for all $(i,x,k) \in S$, where $|S|$ is the total number of feasible states.

**Step 2**: Substitute $\pi(i,x,k)^{old}$ into the balance equations from Case 1 to Case 16 to find $\pi(i,x,k)^{new}$, $\forall i,x,k$.

**Step 3**: Normalize $\pi(i,x,k)^{new}$, $\forall i,x,k$.

**Step 4**: If $\sqrt{\sum\sum\sum_{(i,x,k)\in S}|\pi(i,x,k)^{old} - \pi(i,x,k)^{new}|^2} < \varepsilon$, then stop the iteration. Otherwise, set $\pi(i,x,k)^{old} = \pi(i,x,k)^{new}$, $\forall i,x,k$, and return to **Step 2**.

In our analysis, the convergence threshold $\varepsilon$ is set to $10^{-8}$, and the algorithm typically converges after about 75 iterations.

### 3.1.3. Performance Measure

After obtaining the steady-state probabilities $\pi_{i,x,k}$ through the iterative algorithm, we proceed to compute several performance metrics to evaluate the effectiveness of the system.

First of all, the average number of customers in the whole system, denoted by $L$, is given by:

$$L = \sum_{k=0}^{1}\sum_{i=1}^{N} i\pi_{i,0,k} + \sum_{k=0}^{1}\sum_{i=0}^{N-b}\sum_{x=1}^{b}(i+x)\pi_{i,x,k} \tag{3-5}$$

Second, the average number of customers in customer queue, denoted by $L_c$, is given by:

$$L_c = \sum_{k=0}^{1}\sum_{i=1}^{N} i\pi_{i,0,k} + \sum_{k=0}^{1}\sum_{i=0}^{N-b}\sum_{x=1}^{b} i\pi_{i,x,k} \tag{3-6}$$

Third, the average number of customers in block queue, denoted by $L_b$, is given by:

$$L_b = \sum_{k=0}^{1}\sum_{i=0}^{N-b}\sum_{x=1}^{b} x\pi_{i,x,k} \tag{3-7}$$

Fourth, the blocking probability of the system, denoted by $P_b$, is given by:

$$P_b = \sum_{k=0}^{1} \pi_{N,0,k} + \sum_{k=0}^{1}\sum_{x=1}^{b} \pi_{N-b,x,k} \tag{3-8}$$

Fifth, the throughput of the system, denoted by $T_h$, is given by:

$$T_h = \sum_{i=0}^{N-b}\sum_{x=1}^{b} \mu_2 x\pi_{N-b,x,k} \tag{3-9}$$

Sixth, the average waiting time in the system, denoted by $W$, is given by:

$$W = \frac{L}{\lambda(1-P_b)} \tag{3-10}$$

Seventh, the average waiting time in the customer queue, denoted by $W_c$, is given by:

$$W_c = \frac{L_c}{\lambda(1 - P_b)} \tag{3-11}$$

Eighth, the average waiting time in the block queue, denoted by $W_b$, is given by:

$$W_b = \frac{L_b}{\lambda(1 - P_b)} \tag{3-12}$$

Finally, the average number of blocks participating in the consensus process per unit of time, denoted by $B_n$, is given below.

$$B_n = \sum_{k=0}^{1} \sum_{i=0}^{N-b} \sum_{x=1}^{b} \pi_{i,x,k} \tag{3-13}$$

## 3.2. Scenario 2: Two-Class Customer without Impatience

In this scenario, we consider a two-class customer system without impatience, where the arrivals of high-priority and low-priority customers follow the independent Poisson processes, with arrival rates denoted by $\lambda_H$ and $\lambda_L$, respectively. Customers are served based on the non-preemptive priority discipline, where high-priority customers are placed ahead of low-priority ones in the queue, but ongoing service cannot be interrupted.

The service process is divided into block generation and consensus phases. After arriving at the customer queue, customers wait for the block generation process, which occurs at a rate of $\mu_{1_H}$ and $\mu_{1_L}$ for high-priority and low-priority customers, respectively. Each block is generated according to the partial batch policy, i.e., each block can contain 1 to $b$ customers of the same policy class. Once a block is formed, it is transferred to the block queue, where the consensus process is carried out at the service rate denoted by $\mu_{2_H}$ and $\mu_{2_L}$ for high-priority and low-priority customers, respectively.

In addition, we consider the operational reliability of the system by incorporating the possibility of the system state alternating between ON and OFF periods. During ON periods, both block generation and consensus operations are allowed to proceed, while during OFF periods, these operations are suspended. The durations of both ON and OFF periods are exponentially distributed. The transition rates from ON to OFF and from OFF to ON are given by $\alpha$ and $\beta$, respectively.

The $\lambda_{eff}$ and $\mu_{eff}$ of this scenario is:

$$\lambda_{eff} = \lambda(1 - P_b) \tag{3-14}$$

$$\mu_{eff} = b\left(\frac{1}{\mu_1} + \frac{1}{\mu_2}\right)^{-1} \frac{\beta}{\alpha + \beta} \tag{3-15}$$

### 3.2.1. State Balance Equations

The system under consideration is described as a five-dimensional Markov chain denoted by $(i, j, x, y, k)$, where $i$ and $j$ represent the number of high-priority and low-priority customers in the customer queue, respectively. $x$ and $y$ represent the number of high-priority and low-priority customers in the block queue, respectively. And $k$ denotes the system state. When the block queue is empty (i.e., $x = 0$ and $y = 0$), the maximum number of customers allowed in the customer queue is $N$, implying that $i + j \leq N$. However, when the block queue is occupied (i.e., $x > 0$ or $y > 0$),

the maximum number of customers in the customer queue is reduced to $N-b$, and thus $i+j \leq N-b$.

Customers are scheduled according to the non-preemptive priority discipline, where high-priority customers are always placed ahead of low-priority ones in the queue, but service already in progress cannot be interrupted. When a block is generated, it must contain customer(s) of only one priority class, and is transferred into the block queue as a batch for processing without preemption.

The system state $k=1$ indicates that the system is in the ON state, where customers can enter the customer queue and both block generation and consensus operations can proceed. On the other hand, when $k=0$, the system is in the OFF state, during which only customer arrivals to the queue are permitted, while block generation and consensus are suspended. The state space can be denoted as follows:

$$S=\left\{(i,j,x,y,k)\middle|\begin{Bmatrix}0\leq i\leq N, 0\leq j\leq N-b\\ if\ x=0\ and\ y=0: i+j\leq N\\ if\ 0<x\leq b\ or\ 0<y\leq b: i+j\leq N-b\end{Bmatrix}\right. \tag{3-16}$$

Hence, the total number of feasible states is given by:

$$|S|=2\left[(N+1)(N-b+1)-\frac{(N-b)(N-b+1)}{2}\right]+\frac{2b(N-b+1)(N-b+2)}{2} \tag{3-17}$$

For example, if $N=40$ and $b=15$, the number of feasible states is 22524. The steady state probability of state $(i,j,x,y,k)$ is denoted as $\pi_{i,j,x,y,k}$. In this scenario, the feasible states can be categorized into 99 distinct cases, as described below.

(a) $x=0, y=0, k=0$

Case 1: $i=0, j=0$

$$(\beta+\lambda_H+\lambda_L)\pi_{0,0,0,0,0}=\alpha\pi_{0,0,0,0,1}$$

Case 2: $i=0, 1\leq j\leq N-b-1$

$$(\beta+\lambda_H+\lambda_L)\pi_{0,j,0,0,0}=\alpha\pi_{0,j,0,0,1}+\lambda_L\pi_{0,j-1,0,0,0}$$

Case 3: $i=0, j=N-b$

$$(\beta + \lambda_H)\pi_{0,N-b,0,0,0} = \alpha\pi_{0,N-b,0,0,1} + \lambda_L\pi_{0,N-b-1,0,0,0}$$

Case 4: $1 \le i \le b-1, j = 0$

$$(\beta + \lambda_H + \lambda_L)\pi_{i,0,0,0,0} = \alpha\pi_{i,0,0,0,1} + \lambda_H\pi_{i-1,0,0,0,0}$$

Case 5: $1 \le i \le b-1, 1 \le j \le N-b-1$

$$(\beta + \lambda_H + \lambda_L)\pi_{i,j,0,0,0} = \alpha\pi_{i,j,0,0,1} + \lambda_H\pi_{i-1,j,0,0,0} + \lambda_L\pi_{i,j-1,0,0,0}$$

Case 6: $1 \le i \le b-1, j = N-b$

$$(\beta + \lambda_H)\pi_{i,N-b,0,0,0} = \alpha\pi_{i,N-b,0,0,1} + \lambda_H\pi_{i-1,N-b,0,0,0} + \lambda_L\pi_{i,N-b-1,0,0,0}$$

Case 7: $i = b, j = 0$

$$(\beta + \lambda_H + \lambda_L)\pi_{b,0,0,0,0} = \alpha\pi_{b,0,0,0,1} + \lambda_H\pi_{b-1,0,0,0,0}$$

Case 8: $i = b, 1 \le j \le N-b-1$

$$(\beta + \lambda_H + \lambda_L)\pi_{b,j,0,0,0} = \alpha\pi_{b,j,0,0,1} + \lambda_H\pi_{b-1,j,0,0,0} + \lambda_L\pi_{b,j-1,0,0,0}$$

Case 9: $i = b, j = N-b$

$$\beta\pi_{b,N-b,0,0,0} = \alpha\pi_{b,N-b,0,0,1} + \lambda_H\pi_{b-1,N-b,0,0,0} + \lambda_L\pi_{b,N-b-1,0,0,0}$$

Case 10: $b+1 \le i \le N-b, j = 0$

$$(\beta + \lambda_H + \lambda_L)\pi_{i,0,0,0,0} = \alpha\pi_{i,0,0,0,1} + \lambda_H\pi_{i-1,0,0,0,0}$$

Case 11: $b+1 \le i \le N-b, 1 \le j \le N-1-i$

$$(\beta + \lambda_H + \lambda_L)\pi_{i,j,0,0,0} = \alpha\pi_{i,j,0,0,1} + \lambda_H\pi_{i-1,j,0,0,0} + \lambda_L\pi_{i,j-1,0,0,0}$$

Case 12: $b+1 \le i \le N-b, j = N-i$

$$\beta\pi_{i,N-i,0,0,0} = \alpha\pi_{i,N-i,0,0,1} + \lambda_H\pi_{i-1,N-i,0,0,0} + \lambda_L\pi_{i,N-i-1,0,0,0}$$

Case 13: $N-b+1 \le i \le N-2, j = 0$

$$(\beta + \lambda_H + \lambda_L)\pi_{i,0,0,0,0} = \alpha\pi_{i,0,0,0,1} + \lambda_H\pi_{i-1,0,0,0,0}$$

Case 14: $N-b+1 \le i \le N-2, 1 \le j \le N-1-i$

$$(\beta + \lambda_H + \lambda_L)\pi_{i,j,0,0,0} = \alpha\pi_{i,j,0,0,1} + \lambda_H\pi_{i-1,j,0,0,0} + \lambda_L\pi_{i,j-1,0,0,0}$$

Case 15: $N - b + 1 \leq i \leq N - 2, j = N - i$

$$\beta\pi_{i,N-i,0,0,0} = \alpha\pi_{i,N-i,0,0,1} + \lambda_H\pi_{i-1,N-i,0,0,0} + \lambda_L\pi_{i,N-i-1,0,0,0}$$

Case 16: $i = N - 1, j = 0$

$$(\beta + \lambda_H + \lambda_L)\pi_{N-1,0,0,0,0} = \alpha\pi_{N-1,0,0,0,1} + \lambda_H\pi_{N-2,0,0,0,0}$$

Case 17: $i = N - 1, j = 1$

$$\beta\pi_{N-1,1,0,0,0} = \alpha\pi_{N-1,1,0,0,1} + \lambda_H\pi_{N-2,1,0,0,0} + \lambda_L\pi_{N-1,0,0,0,0}$$

Case 18: $i = N, j = 0$

$$\beta\pi_{N,0,0,0,0} = \alpha\pi_{N,0,0,0,1} + \lambda_H\pi_{N-1,0,0,0,0}$$

(b) $1 \leq x \leq b, y = 0, k = 0$

Case 19: $i = 0, j = 0$

$$(\beta + \lambda_H + \lambda_L)\pi_{0,0,x,0,0} = \alpha\pi_{0,0,x,0,1}$$

Case 20: $i = 0, 1 \leq j \leq N - b - 1$

$$(\beta + \lambda_H + \lambda_L)\pi_{0,j,x,0,0} = \alpha\pi_{0,j,x,0,1} + \lambda_L\pi_{0,j-1,x,0,0}$$

Case 21: $i = 0, j = N - b$

$$\beta\pi_{0,N-b,x,0,0} = \alpha\pi_{0,N-b,x,0,1} + \lambda_L\pi_{0,N-b-1,x,0,0}$$

Case 22: $1 \leq i \leq N - b - 2, j = 0$

$$(\beta + \lambda_H + \lambda_L)\pi_{i,0,x,0,0} = \alpha\pi_{i,0,x,0,1} + \lambda_H\pi_{i-1,0,x,0,0}$$

Case 23: $1 \leq i \leq N - b - 2, 1 \leq j \leq N - b - 1 - i$

$$(\beta + \lambda_H + \lambda_L)\pi_{i,j,x,0,0} = \alpha\pi_{i,j,x,0,1} + \lambda_H\pi_{i-1,j,x,0,0} + \lambda_L\pi_{i,j-1,x,0,0}$$

Case 24: $1 \leq i \leq N - b - 2, j = N - b - i$

$$\beta\pi_{i,N-b-i,x,0,0} = \alpha\pi_{i,N-b-i,x,0,1} + \lambda_H\pi_{i-1,N-b-i,x,0,0} + \lambda_L\pi_{i,N-b-i-1,x,0,0}$$

Case 25: $i = N - b - 1, j = 0$

$$(\beta + \lambda_H + \lambda_L)\pi_{N-b-1,0,x,0,0} = \alpha\pi_{N-b-1,0,x,0,1} + \lambda_H\pi_{N-b-2,0,x,0,0}$$

Case 26: $i = N - b - 1, j = 1$

$$\beta\pi_{N-b-1,1,x,0,0} = \alpha\pi_{N-b-1,1,x,0,1} + \lambda_H\pi_{N-b-2,1,x,0,0} + \lambda_L\pi_{N-b-1,0,x,0,0}$$

Case 27: $i = N - b, j = 0$

$$\beta\pi_{N-b,0,x,0,0} = \alpha\pi_{N-b,0,x,0,1} + \lambda_H\pi_{N-b-1,0,x,0,0}$$

(c) $x = 0,1 \le y \le b, k = 0$

Case 28: $i = 0, j = 0$

$$(\beta + \lambda_H + \lambda_L)\pi_{0,0,0,y,0} = \alpha\pi_{0,0,0,y,1}$$

Case 29: $i = 0,1 \le j \le N - b - 1$

$$(\beta + \lambda_H + \lambda_L)\pi_{0,j,0,y,0} = \alpha\pi_{0,j,0,y,1} + \lambda_L\pi_{0,j-1,0,y,0}$$

Case 30: $i = 0, j = N - b$

$$\beta\pi_{0,N-b,0,y,0} = \alpha\pi_{0,N-b,0,y,1} + \lambda_L\pi_{0,N-b-1,0,y,0}$$

Case 31: $1 \le i \le N - b - 2, j = 0$

$$(\beta + \lambda_H + \lambda_L)\pi_{i,0,0,y,0} = \alpha\pi_{i,0,0,y,1} + \lambda_H\pi_{i-1,0,0,y,0}$$

Case 32: $1 \le i \le N - b - 2, 1 \le j \le N - b - 1 - i$

$$(\beta + \lambda_H + \lambda_L)\pi_{i,j,0,y,0} = \alpha\pi_{i,j,0,y,1} + \lambda_H\pi_{i-1,j,0,y,0} + \lambda_L\pi_{i,j-1,0,y,0}$$

Case 33: $1 \le i \le N - b - 2, j = N - b - i$

$$\beta\pi_{i,N-b-i,0,y,0} = \alpha\pi_{i,N-b-i,0,y,1} + \lambda_H\pi_{i-1,N-b-i,0,y,0} + \lambda_L\pi_{i,N-b-i-1,0,y,0}$$

Case 34: $i = N - b - 1, j = 0$

$$(\beta + \lambda_H + \lambda_L)\pi_{N-b-1,0,0,y,0} = \alpha\pi_{N-b-1,0,0,y,1} + \lambda_H\pi_{N-b-2,0,0,y,0}$$

Case 35: $i = N - b - 1, j = 1$

$$\beta\pi_{N-b-1,1,0,y,0} = \alpha\pi_{N-b-1,1,0,y,1} + \lambda_H\pi_{N-b-2,1,0,y,0} + \lambda_L\pi_{N-b-1,0,0,y,0}$$

Case 36: $i = N - b, j = 0$

$$\beta\pi_{N-b,0,0,y,0} = \alpha\pi_{N-b,0,0,y,1} + \lambda_H\pi_{N-b-1,0,0,y,0}$$

(d) $x = 0, y = 0, k = 1$

Case 37: $i = 0, j = 0$

$$(\alpha + \lambda_H + \lambda_L)\pi_{0,0,0,0,1} = \beta\pi_{0,0,0,0,0} + \sum_{t=1}^{b} \mu_{2_H}\pi_{0,0,t,0,1} + \sum_{t=1}^{b} \mu_{2_L}\pi_{0,0,0,t,1}$$

Case 38: $i = 0, 1 \le j \le N - b - 1$

$$\left(\alpha + \lambda_H + \lambda_L + \mu_{1_L}\right)\pi_{0,j,0,0,1}$$
$$= \beta\pi_{0,j,0,0,0} + \lambda_L\pi_{0,j-1,0,0,1} + \sum_{t=1}^{b} \mu_{2_H}\pi_{0,j,t,0,1} + \sum_{t=1}^{b} \mu_{2_L}\pi_{0,j,0,t,1}$$

Case 39: $i = 0, j = N - b$

$$\left(\alpha + \lambda_H + \mu_{1_L}\right)\pi_{0,N-b,0,0,1}$$
$$= \beta\pi_{0,N-b,0,0,0} + \lambda_L\pi_{0,N-b-1,0,0,1} + \sum_{t=1}^{b} \mu_{2_H}\pi_{0,N-b,t,0,1}$$
$$+ \sum_{t=1}^{b} \mu_{2_L}\pi_{0,N-b,0,t,1}$$

Case 40: $1 \le i \le b - 1, j = 0$

$$\left(\alpha + \lambda_H + \lambda_L + \mu_{1_H}\right)\pi_{i,0,0,0,1}$$
$$= \beta\pi_{i,0,0,0,0} + \lambda_H\pi_{i-1,0,0,0,1} + \sum_{t=1}^{b} \mu_{2_H}\pi_{i,0,t,0,1} + \sum_{t=1}^{b} \mu_{2_L}\pi_{i,0,0,t,1}$$

Case 41: $1 \le i \le b - 1, 1 \le j \le N - b - i$

$$(\alpha + \lambda_H + \lambda_L + \mu_{1_H})\pi_{i,j,0,0,1}$$

$$= \beta\pi_{i,j,0,0,0} + \lambda_H\pi_{i-1,j,0,0,1} + \lambda_L\pi_{i,j-1,0,0,1} + \sum_{t=1}^{b}\mu_{2_H}\pi_{i,j,t,0,1}$$

$$+ \sum_{t=1}^{b}\mu_{2_L}\pi_{i,j,0,t,1}$$

Case 42: $1 \le i \le b-1, N-b+1-i \le j \le N-b-1$

$$(\alpha + \lambda_H + \lambda_L + \mu_{1_H})\pi_{i,j,0,0,1} = \beta\pi_{i,j,0,0,0} + \lambda_H\pi_{i-1,j,0,0,1} + \lambda_L\pi_{i,j-1,0,0,1}$$

Case 43: $1 \le i \le b-1, j = N-b$

$$(\alpha + \lambda_H + \mu_{1_H})\pi_{i,N-b,0,0,1} = \beta\pi_{i,N-b,0,0,0} + \lambda_H\pi_{i-1,N-b,0,0,1} + \lambda_L\pi_{i,N-b-1,0,0,1}$$

Case 44: $i = b, j = 0$

$$(\alpha + \lambda_H + \lambda_L + \mu_{1_H})\pi_{2,0,0,0,1}$$

$$= \beta\pi_{2,0,0,0,0} + \lambda_H\pi_{2-1,0,0,0,1} + \sum_{t=1}^{b}\mu_{2_H}\pi_{2,0,t,0,1} + \sum_{t=1}^{b}\mu_{2_L}\pi_{2,0,0,t,1}$$

Case 45: $i = b, 1 \le j \le N-b-i$

$$(\alpha + \lambda_H + \lambda_L + \mu_{1_H})\pi_{2,j,0,0,1}$$

$$= \beta\pi_{2,j,0,0,0} + \lambda_H\pi_{2-1,j,0,0,1} + \lambda_L\pi_{2,j-1,0,0,1} + \sum_{t=1}^{b}\mu_{2_H}\pi_{2,j,t,0,1}$$

$$+ \sum_{t=1}^{b}\mu_{2_L}\pi_{2,j,0,t,1}$$

Case 46: $i = b, N-b+1-i \le j \le N-b-1$

$$(\alpha + \lambda_H + \lambda_L + \mu_{1_H})\pi_{2,j,0,0,1} = \beta\pi_{2,j,0,0,0} + \lambda_H\pi_{2-1,j,0,0,1} + \lambda_L\pi_{2,j-1,0,0,1}$$

Case 47: $i = b, j = N-b$

$$(\alpha + \mu_{1_H})\pi_{2,N-b,0,0,1} = \beta\pi_{2,N-b,0,0,0} + \lambda_H\pi_{2-1,N-b,0,0,1} + \lambda_L\pi_{2,N-b-1,0,0,1}$$

Case 48: $b+1 \le i \le N-b-1, j=0$

$$(\alpha+\lambda_H+\lambda_L+\mu_{1_H})\pi_{i,0,0,0,1} = \beta\pi_{i,0,0,0,0}+\lambda_H\pi_{i-1,0,0,0,1}+\sum_{t=1}^{b}\mu_{2_H}\pi_{i,0,t,0,1}+\sum_{t=1}^{b}\mu_{2_L}\pi_{i,0,0,t,1}$$

Case 49: $b+1 \le i \le N-b-1, 1 \le j \le N-b-i$

$$(\alpha+\lambda_H+\lambda_L+\mu_{1_H})\pi_{i,j,0,0,1} = \beta\pi_{i,j,0,0,0}+\lambda_H\pi_{i-1,j,0,0,1}+\lambda_L\pi_{i,j-1,0,0,1}+\sum_{t=1}^{b}\mu_{2_H}\pi_{i,j,t,0,1}+\sum_{t=1}^{b}\mu_{2_L}\pi_{i,j,0,t,1}$$

Case 50: $b+1 \le i \le N-b-1, N-b+1-i \le j \le N-1-i$

$$(\alpha+\lambda_H+\lambda_L+\mu_{1_H})\pi_{i,j,0,0,1} = \beta\pi_{i,j,0,0,0}+\lambda_H\pi_{i-1,j,0,0,1}+\lambda_L\pi_{i,j-1,0,0,1}$$

Case 51: $b+1 \le i \le N-b-1, j=N-i$

$$(\alpha+\mu_{1_H})\pi_{i,N-i,0,0,1} = \beta\pi_{i,N-i,0,0,0}+\lambda_H\pi_{i-1,N-i,0,0,1}+\lambda_L\pi_{i,N-i-1,0,0,1}$$

Case 52: $i=N-b, j=0$

$$(\alpha+\lambda_H+\lambda_L+\mu_{1_H})\pi_{N-b,0,0,0,1} = \beta\pi_{N-b,0,0,0,0}+\lambda_H\pi_{N-b-1,0,0,0,1}+\sum_{t=1}^{b}\mu_{2_H}\pi_{N-b,0,t,0,1}+\sum_{t=1}^{b}\mu_{2_L}\pi_{N-b,0,0,t,1}$$

Case 53: $i=N-b, 1 \le j \le b-1$

$$(\alpha+\lambda_H+\lambda_L+\mu_{1_H})\pi_{N-b,j,0,0,1} = \beta\pi_{N-b,j,0,0,0}+\lambda_H\pi_{N-b-1,j,0,0,1}+\lambda_L\pi_{N-b,j-1,0,0,1}$$

Case 54: $i=N-b, j=b$

$$(\alpha + \mu_{1_H})\pi_{N-b,b,0,0,1} = \beta\pi_{N-b,b,0,0,0} + \lambda_H\pi_{N-b-1,b,0,0,1} + \lambda_L\pi_{N-b,b-1,0,0,1}$$

Case 55: $N - b + 1 \leq i \leq N - 2, j = 0$

$$(\alpha + \lambda_H + \lambda_L + \mu_{1_H})\pi_{i,0,0,0,1} = \beta\pi_{i,0,0,0,0} + \lambda_H\pi_{i-1,0,0,0,1}$$

Case 56: $N - b + 1 \leq i \leq N - 2, 1 \leq j \leq N - 1 - i$

$$(\alpha + \lambda_H + \lambda_L + \mu_{1_H})\pi_{i,j,0,0,1} = \beta\pi_{i,j,0,0,0} + \lambda_H\pi_{i-1,j,0,0,1} + \lambda_L\pi_{i,j-1,0,0,1}$$

Case 57: $N - b + 1 \leq i \leq N - 2, j = N - i$

$$(\alpha + \mu_{1_H})\pi_{i,N-i,0,0,1} = \beta\pi_{i,N-i,0,0,0} + \lambda_H\pi_{i-1,N-i,0,0,1} + \lambda_L\pi_{i,N-i-1,0,0,1}$$

Case 58: $i = N - 1, j = 0$

$$(\alpha + \lambda_H + \lambda_L + \mu_{1_H})\pi_{N-1,0,0,0,1} = \beta\pi_{N-1,0,0,0,0} + \lambda_H\pi_{N-2,0,0,0,1}$$

Case 59: $i = N - 1, j = 1$

$$(\alpha + \mu_{1_H})\pi_{N-1,1,0,0,1} = \beta\pi_{N-1,1,0,0,0} + \lambda_H\pi_{N-2,1,0,0,1} + \lambda_L\pi_{N-1,0,0,0,1}$$

Case 60: $i = N, j = 0$

$$(\alpha + \mu_{1_H})\pi_{N,0,0,0,1} = \beta\pi_{N,0,0,0,0} + \lambda_H\pi_{N-1,0,0,0,1}$$

(e) $1 \leq x < b, y = 0, k = 1$

Case 61: $i = 0, j = 0$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_H})\pi_{0,0,x,0,1} = \beta\pi_{0,0,x,0,0} + \mu_{1_H}\pi_{x,0,0,0,1}$$

Case 62: $i = 0, 1 \leq j \leq N - b - 1$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_H})\pi_{0,j,x,0,1} = \beta\pi_{0,j,x,0,0} + \lambda_L\pi_{0,j-1,x,0,1} + \mu_{1_H}\pi_{x,j,0,0,1}$$

Case 63: $i = 0, j = N - b$

$$(\alpha + \mu_{2_H})\pi_{0,N-b,x,0,1} = \beta\pi_{0,N-b,x,0,0} + \lambda_L\pi_{0,N-b-1,x,0,1} + \mu_{1_H}\pi_{x,N-b,0,0,1}$$

Case 64: $1 \leq i \leq N - b - 2, j = 0$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_H})\pi_{i,0,x,0,1} = \beta\pi_{i,0,x,0,0} + \lambda_H\pi_{i-1,0,x,0,1}$$

Case 65: $1 \le i \le N-b-2, 1 \le j \le N-b-1-i$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_H})\pi_{i,j,x,0,1} = \beta\pi_{i,j,x,0,0} + \lambda_H\pi_{i-1,j,x,0,1} + \lambda_L\pi_{i,j-1,x,0,1}$$

Case 66: $1 \le i \le N-b-2, j = N-b-i$

$$(\alpha + \mu_{2_H})\pi_{i,N-b-i,x,0,1} = \beta\pi_{i,N-b-i,x,0,0} + \lambda_H\pi_{i-1,N-b-i,x,0,1} + \lambda_L\pi_{i,N-b-i-1,x,0,1}$$

Case 67: $i = N-b-1, j = 0$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_H})\pi_{N-b-1,0,x,0,1} = \beta\pi_{N-b-1,0,x,0,0} + \lambda_H\pi_{N-b-2,0,x,0,1}$$

Case 68: $i = N-b-1, j = 1$

$$(\alpha + \mu_{2_H})\pi_{N-b-1,1,x,0,1} = \beta\pi_{N-b-1,1,x,0,0} + \lambda_H\pi_{N-b-2,1,x,0,1} + \lambda_L\pi_{N-b-1,0,x,0,1}$$

Case 69: $i = N-b, j = 0$

$$(\alpha + \mu_{2_H})\pi_{N-b,0,x,0,1} = \beta\pi_{N-b,0,x,0,0} + \lambda_H\pi_{N-b-1,0,x,0,1}$$

(f) $x = b, y = 0, k = 1$

Case 70: $i = 0, j = 0$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_H})\pi_{0,0,b,0,1} = \beta\pi_{0,0,b,0,0} + \mu_{1_H}\pi_{2,0,0,0,1}$$

Case 71: $i = 0, 1 \le j \le N-b-1$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_H})\pi_{0,j,b,0,1} = \beta\pi_{0,j,b,0,0} + \lambda_L\pi_{0,j-1,b,0,1} + \mu_{1_H}\pi_{2,j,0,0,1}$$

Case 72: $i = 0, j = N-b$

$$(\alpha + \mu_{2_H})\pi_{0,N-b,b,0,1} = \beta\pi_{0,N-b,b,0,0} + \lambda_L\pi_{0,N-b-1,b,0,1} + \mu_{1_H}\pi_{2,N-b,0,0,1}$$

Case 73: $1 \le i \le N-b-2, j = 0$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_H})\pi_{i,0,b,0,1} = \beta\pi_{i,0,b,0,0} + \lambda_H\pi_{i-1,0,b,0,1} + \mu_{1_H}\pi_{i+b,0,0,0,1}$$

Case 74: $1 \le i \le N-b-2, 1 \le j \le N-b-1-i$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_H})\pi_{i,j,b,0,1} = \beta\pi_{i,j,b,0,0} + \lambda_H\pi_{i-1,j,b,0,1} + \lambda_L\pi_{i,j-1,b,0,1} + \mu_{1_H}\pi_{i+b,j,0,0,1}$$

Case 75: $1 \le i \le N-b-2, j = N-b-i$

$$(\alpha + \mu_{2_H})\pi_{i,N-b-i,b,0,1} = \beta\pi_{i,N-b-i,b,0,0} + \lambda_H\pi_{i-1,N-b-i,b,0,1} + \lambda_L\pi_{i,N-b-i-1,b,0,1} + \mu_{1_H}\pi_{i+b,N-b-i,0,0,1}$$

Case 76: $i = N-b-1, j = 0$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_H})\pi_{N-b-1,0,b,0,1} = \beta\pi_{N-b-1,0,b,0,0} + \lambda_H\pi_{N-b-2,0,b,0,1} + \mu_{1_H}\pi_{N-1,0,0,0,1}$$

Case 77: $i = N-b-1, j = 1$

$$(\alpha + \mu_{2_H})\pi_{N-b-1,1,b,0,1} = \beta\pi_{N-b-1,1,b,0,0} + \lambda_H\pi_{N-b-2,1,b,0,1} + \lambda_L\pi_{N-b-1,0,b,0,1} + \mu_{1_H}\pi_{N-1,1,0,0,1}$$

Case 78: $i = N-b, j = 0$

$$(\alpha + \mu_{2_H})\pi_{N-b,0,b,0,1} = \beta\pi_{N-b,0,b,0,0} + \lambda_H\pi_{N-b-1,0,b,0,1} + \mu_{1_H}\pi_{N,0,0,0,1}$$

(g) $x = 0, 1 \le y \le b, k = 1$

Case 79: $i = 0, j = 0$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_L})\pi_{0,0,0,y,1} = \beta\pi_{0,0,0,y,0} + \mu_{1_L}\pi_{0,y,0,0,1}$$

Case 80: $1 \le i \le N-b-1, j = 0$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_L})\pi_{i,0,0,y,1} = \beta\pi_{i,0,0,y,0} + \lambda_H\pi_{i-1,0,0,y,1}$$

Case 81: $i = N-b, j = 0$

$$(\alpha + \mu_{2_L})\pi_{N-b,0,0,y,1} = \beta\pi_{N-b,0,0,y,0} + \lambda_H\pi_{N-b-1,0,0,y,1}$$

Case 82: $i = 0, 1 \le j \le N-b-2$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_L})\pi_{0,j,0,y,1} = \beta\pi_{0,j,0,y,0} + \lambda_L\pi_{0,j-1,0,y,1}$$

Case 83: $1 \le i \le N-b-1-j, 1 \le j \le N-b-2$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_L})\pi_{i,j,0,y,1} = \beta\pi_{i,j,0,y,0} + \lambda_H\pi_{i-1,j,0,y,1} + \lambda_L\pi_{i,j-1,0,y,1}$$

Case 84: $i = N - b - j, 1 \leq j \leq N - b - 2$

$$(\alpha + \mu_{2_L})\pi_{N-b-j,j,0,y,1} = \beta\pi_{N-b-j,j,0,y,0} + \lambda_H\pi_{N-b-j-1,j,0,y,1} + \lambda_L\pi_{N-b-j,j-1,0,y,1}$$

Case 85: $i = 0, j = N - b - 1$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_L})\pi_{0,N-b-1,0,y,1} = \beta\pi_{0,N-b-1,0,y,0} + \lambda_L\pi_{0,N-b-2,0,y,1}$$

Case 86: $i = 1, j = N - b - 1$

$$(\alpha + \mu_{2_L})\pi_{1,N-b-1,0,y,1} = \beta\pi_{1,N-b-1,0,y,0} + \lambda_H\pi_{0,N-b-1,0,y,1} + \lambda_L\pi_{1,N-b-2,0,y,1}$$

Case 87: $i = 0, j = N - b$

$$(\alpha + \mu_{2_L})\pi_{0,N-b,0,y,1} = \beta\pi_{0,N-b,0,y,0} + \lambda_L\pi_{0,N-b-1,0,y,1}$$

(h) $x = 0, y = b, k = 1$

Case 88: $i = 0, j = 0$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_L})\pi_{0,0,0,b,1} = \beta\pi_{0,0,0,b,0} + \mu_{1_L}\pi_{0,b,0,0,1}$$

Case 89: $1 \leq i \leq N - b - 1, j = 0$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_L})\pi_{i,0,0,b,1} = \beta\pi_{i,0,0,b,0} + \lambda_H\pi_{i-1,0,0,b,1}$$

Case 90: $i = N - b, j = 0$

$$(\alpha + \mu_{2_L})\pi_{N-b,0,0,b,1} = \beta\pi_{N-b,0,0,b,0} + \lambda_H\pi_{N-b-1,0,0,b,1}$$

Case 91: $i = 0, 1 \leq j \leq N - 2b$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_L})\pi_{0,j,0,b,1} = \beta\pi_{0,j,0,b,0} + \lambda_L\pi_{0,j-1,0,b,1} + \mu_{1_L}\pi_{0,j+b,0,0,1}$$

Case 92: $1 \leq i \leq N - b - 1 - j, 1 \leq j \leq N - 2b$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_L})\pi_{i,j,0,b,1} = \beta\pi_{i,j,0,b,0} + \lambda_H\pi_{i-1,j,0,b,1} + \lambda_L\pi_{i,j-1,0,b,1}$$

Case 93: $i = N - b - j, 1 \leq j \leq N - 2b$

$$(\alpha + \mu_{2_L})\pi_{N-b-j,j,0,b,1} = \beta\pi_{N-b-j,j,0,b,0} + \lambda_H\pi_{N-b-j-1,j,0,b,1} + \lambda_L\pi_{N-b-j,j-1,0,b,1}$$

Case 94: $i = 0, N - 2b + 1 \leq j \leq N - b - 2$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_L})\pi_{0,j,0,b,1} = \beta\pi_{0,j,0,b,0} + \lambda_L\pi_{0,j-1,0,b,1}$$

Case 95: $1 \le i \le N-b-1-j, N-2b+1 \le j \le N-b-2$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_L})\pi_{i,j,0,b,1} = \beta\pi_{i,j,0,b,0} + \lambda_H\pi_{i-1,j,0,b,1} + \lambda_L\pi_{i,j-1,0,b,1}$$

Case 96: $i = N-b-j, N-2b+1 \le j \le N-b-2$

$$(\alpha + \mu_{2_L})\pi_{N-b-j,j,0,b,1} = \beta\pi_{N-b-j,j,0,b,0} + \lambda_H\pi_{N-b-j-1,j,0,b,1} + \lambda_L\pi_{N-b-j,j-1,0,b,1}$$

Case 97: $i = 0, j = N-b-1$

$$(\alpha + \lambda_H + \lambda_L + \mu_{2_L})\pi_{0,N-b-1,0,b,1} = \beta\pi_{0,N-b-1,0,b,0} + \lambda_L\pi_{0,N-b-2,0,b,1}$$

Case 98: $i = 1, j = N-b-1$

$$(\alpha + \mu_{2_L})\pi_{1,N-b-1,0,b,1} = \beta\pi_{1,N-b-1,0,b,0} + \lambda_H\pi_{0,N-b-1,0,b,1} + \lambda_L\pi_{1,N-b-2,0,b,1}$$

Case 99: $i = 0, j = N-b$

$$(\alpha + \mu_{2_L})\pi_{0,N-b,0,b,1} = \beta\pi_{0,N-b,0,b,0} + \lambda_L\pi_{0,N-b-1,0,b,1}$$

Given the large number of equations presented above, it is impractical to illustrate all the corresponding state transition diagrams. Therefore, we focus on a relatively complex case, specifically Case 45, as a representative example, shown in Figure 3-3.

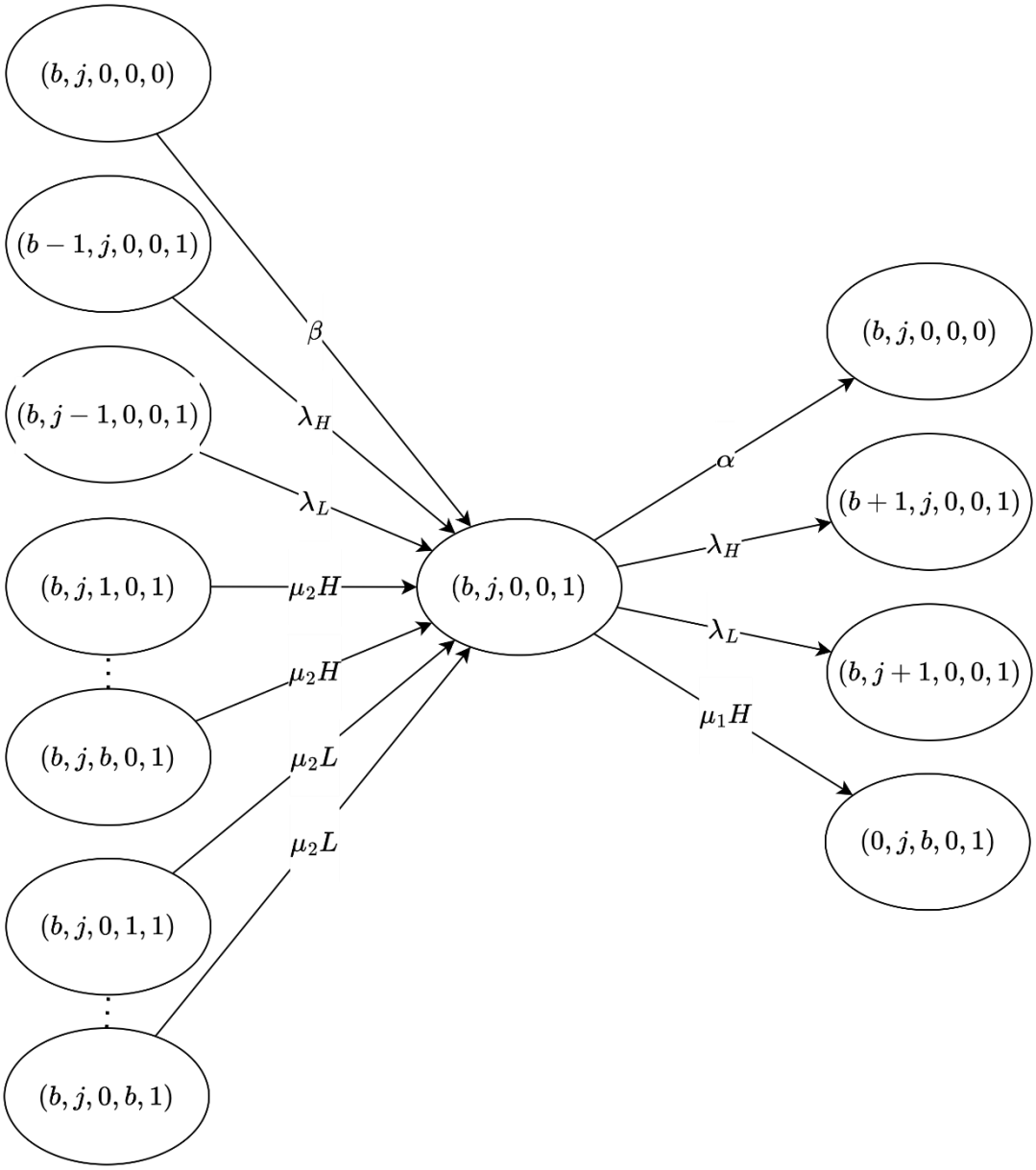


Figure 3-3: The state transition diagram of Case 45: $i = b, 1 \leq j \leq N - 2b, x = 0, y = 0, k = 1$

### 3.2.2. Iterative Algorithm

We use the iterative algorithm provided below, and perform calculations on the state balance equations until they converge, allowing us to determine the steady-state distribution of the system.

## Iterative algorithm:

**Step 1**: Initialize $\pi(i, j, x, y, k)^{old} = \frac{1}{|s|}$ for all $(i, x, k) \in S$, where $|S|$ is the total number of feasible states.

**Step 2**: Substitute $\pi(i,j,x,y,k)^{old}$ into the balance equations from Case 1 to Case 16 to find $\pi(i,j,x,y,k)^{new}$, $\forall i,x,k$.

**Step 3**: Normalize $\pi(i,x,k)^{new}$, $\forall i,x,k$.

**Step 4**: If $\sqrt{\sum\sum\sum_{(i,x,k)\in S}|\pi(i,j,x,y,k)^{old}-\pi(i,j,x,y,k)^{new}|^2} < \varepsilon$, then stop the iteration. Otherwise, set $\pi(i,j,x,y,k)^{old} = \pi(i,j,x,y,k)^{new}$, $\forall i,x,k$, and return to **Step 2**.

In our analysis, the convergence threshold $\varepsilon$ is set to $10^{-8}$, and the algorithm typically converges after about 80 iterations.

### 3.2.3. Performance Measure

After obtaining the steady-state probabilities $\pi_{i,j,x,y,k}$ through the iterative algorithm, we proceed to compute several performance metrics to evaluate the effectiveness of the system.

First of all, the average number of high-priority and low-priority customers in the whole system, denoted by $L_H$ and $L_L$, respectively, is given by:

$$L_H = \sum_{k=0}^{1}\sum_{j=0}^{N}\sum_{i=0}^{N-j} i\pi_{i,j,0,0,k} + \sum_{k=0}^{1}\sum_{x=1}^{b}\sum_{j=0}^{N-b}\sum_{i=0}^{N-b-j}(i+x)\pi_{i,j,x,0,k} + \sum_{k=0}^{1}\sum_{y=1}^{b}\sum_{j=0}^{N-b}\sum_{i=0}^{N-b-j} i\pi_{i,j,0,y,k} \tag{3-18}$$

$$L_L = \sum_{k=0}^{1}\sum_{j=0}^{N}\sum_{i=0}^{N-j} j\pi_{i,j,0,0,k} + \sum_{k=0}^{1}\sum_{x=1}^{b}\sum_{j=0}^{N-b}\sum_{i=0}^{N-b-j} j\pi_{i,j,x,0,k} + \sum_{k=0}^{1}\sum_{y=1}^{b}\sum_{j=0}^{N-b}\sum_{i=0}^{N-b-j}(j+y)\pi_{i,j,0,y,k} \tag{3-19}$$

The average number of customers in the whole system, denoted by $L$, is given by:

$$L = L_H + L_L \tag{3-20}$$

Second, the average number of high-priority and low-priority customers in

customer queue, denoted by $L_{c_H}$ and $L_{c_L}$, respectively, is given by:

$$L_{c_H} = \sum_{k=0}^{1}\sum_{j=0}^{N}\sum_{i=0}^{N-j} i\pi_{i,j,0,0,k} + \sum_{k=0}^{1}\sum_{x=1}^{b}\sum_{j=0}^{N-b}\sum_{i=0}^{N-b-j} i\pi_{i,j,x,0,k} + \sum_{k=0}^{1}\sum_{y=1}^{b}\sum_{j=0}^{N-b}\sum_{i=0}^{N-b-j} i\pi_{i,j,0,y,k} \tag{3-21}$$

$$L_{c_L} = \sum_{k=0}^{1}\sum_{j=0}^{N}\sum_{i=0}^{N-j} j\pi_{i,j,0,0,k} + \sum_{k=0}^{1}\sum_{x=1}^{b}\sum_{j=0}^{N-b}\sum_{i=0}^{N-b-j} j\pi_{i,j,x,0,k} + \sum_{k=0}^{1}\sum_{y=1}^{b}\sum_{j=0}^{N-b}\sum_{i=0}^{N-b-j} j\pi_{i,j,0,y,k} \tag{3-22}$$

The average number of customers in customer queue, denoted by $L_c$, is given by:

$$L_c = L_{c_H} + L_{c_L} \tag{3-23}$$

Third, the average number of high-priority and low-priority customers in block queue, denoted by $L_{b_H}$ and $L_{b_L}$, respectively, is given by:

$$L_{b_H} = \sum_{k=0}^{1}\sum_{x=1}^{b}\sum_{j=0}^{N-b}\sum_{i=0}^{N-b-j} x\pi_{i,j,x,0,k} \tag{3-24}$$

$$L_{b_L} = \sum_{k=0}^{1}\sum_{y=1}^{b}\sum_{j=0}^{N-b}\sum_{i=0}^{N-b-j} y\pi_{i,j,0,y,k} \tag{3-25}$$

The average number of customers in block queue, denoted by $L_b$, is given by:

$$L_b = L_{b_H} + L_{b_L} \tag{3-26}$$

Fourth, the blocking probability of high-priority and low-priority customers in the system, denoted by $P_{b_H}$ and $P_{b_L}$, respectively, is given by:

$$P_{b_H} = \sum_{k=0}^{1}\sum_{j=0}^{N-b}\pi_{N-j,j,0,0,k} + \sum_{k=0}^{1}\sum_{x=1}^{b}\sum_{j=0}^{N-b}\pi_{N-b-j,j,x,0,k} + \sum_{k=0}^{1}\sum_{y=1}^{b}\sum_{j=0}^{N-b}\pi_{N-b-j,j,0,y,k} \tag{3-27}$$

$$P_{b_L} = \sum_{k=0}^{1}\sum_{i=0}^{b}\pi_{i,N-b,0,0,k} + \sum_{k=0}^{1}\sum_{j=0}^{N-b-1}\pi_{N-j,j,0,0,k} + \sum_{k=0}^{1}\sum_{x=1}^{b}\sum_{j=0}^{N-b}\pi_{N-b-j,j,x,0,k} + \sum_{k=0}^{1}\sum_{y=1}^{b}\sum_{j=0}^{N-b}\pi_{N-b-j,j,0,y,k} \tag{3-28}$$

The blocking probability of the system, denoted by $P_b$, is given by:

$$P_b = \frac{\lambda_H P_{b_H} + \lambda_L P_{b_L}}{\lambda_H + \lambda_L} \tag{3-29}$$

Fifth, the throughput of high-priority and low-priority customers in the system, denoted by $T_{h_H}$ and $T_{h_L}$, respectively, is given by:

$$T_{h_H} = \sum_{x=1}^{b}\sum_{i=1}^{N-b}\sum_{j=0}^{N-b}\mu_{2_H} x\pi_{i,j,x,0,1} \tag{3-30}$$

$$T_{h_L} = \sum_{y=1}^{b}\sum_{i=1}^{N-b}\sum_{j=0}^{N-b}\mu_{2_L} y\pi_{i,j,x,0,1} \tag{3-31}$$

The throughput of the system, denoted by $T_h$, is given by:

$$T_h = T_{h_H} + T_{h_L} \tag{3-32}$$

Sixth, the average waiting time of the high-priority and low-priority customers in the system, denoted by $W_H$ and $W_L$, respectively, is given by:

$$W_H = \frac{L_H}{\lambda_H\left(1 - P_{b_H}\right)} \tag{3-33}$$

$$W_L = \frac{L_L}{\lambda_L\left(1 - P_{b_L}\right)} \tag{3-34}$$

The average waiting time in the system, denoted by $W$, is given by:

$$W = \frac{L}{(\lambda_H + \lambda_L)(1 - P_b)} \tag{3-35}$$

Seventh, the average waiting time of the high-priority and low-priority customers in the customer queue, denoted by $W_{c_H}$ and $W_{c_L}$, respectively, is given by:

$$W_{c_H} = \frac{L_{c_H}}{\lambda_H(1 - P_{b_H})} \tag{3-36}$$

$$W_{c_L} = \frac{L_{c_L}}{\lambda_L(1 - P_{b_L})} \tag{3-37}$$

The average waiting time in the customer queue, denoted by $W_c$, is given by:

$$W_c = \frac{L_c}{(\lambda_H + \lambda_L)(1 - P_b)} \tag{3-38}$$

Eighth, the average waiting time of the high-priority and low-priority customers in the block queue, denoted by $W_{b_H}$ and $W_{b_L}$, respectively, is given by:

$$W_{b_H} = \frac{L_{b_H}}{\lambda_H(1 - P_{b_H})} \tag{3-39}$$

$$W_{b_L} = \frac{L_{b_L}}{\lambda_L(1 - P_{b_L})} \tag{3-40}$$

The average waiting time in the block queue, denoted by $W_b$, is given by:

$$W_b = \frac{L_b}{(\lambda_H + \lambda_L)(1 - P_b)} \tag{3-41}$$

Finally, the average number of high-priority and low-priority blocks participating in the consensus process, denoted by $B_{n_H}$ and $B_{n_L}$, is given by:

$$B_{n_H} = \sum_{k=0}^{1} \sum_{x=0}^{b} \sum_{j=1}^{N-b} \sum_{i=0}^{N-j} \pi_{i,j,x,0,k} \tag{3-42}$$

$$B_{n_L} = \sum_{k=0}^{1} \sum_{y=0}^{b} \sum_{j=1}^{N-b} \sum_{i=0}^{N-j} \pi_{i,j,0,y,k} \tag{3-43}$$

The average number of blocks participating in the consensus process per unit of time, denoted by $B_n$, is given by:

$$B_n = B_{n_H} + B_{n_L} \tag{3-44}$$

## 3.3. Scenario 3: Single-Class Customer with Impatience

In this scenario, we consider a single-class customer system with impatience, where arrivals follow a Poisson process and customers are served based on the First-Come-First-Served (FCFS) discipline. The service process is divided into block generation and consensus phases, and the system switches between ON and OFF states, affecting service availability.

Assume that the arrivals of customers follow a Poisson process, where the arrival rate is denoted by λ. After arriving at the customer queue, customers wait for the block generation process, which occurs at a rate of $\mu_1$. Each block is generated according to the partial batch policy, i.e., each block can contain 1 to $b$ customers. Once a block is formed, a group of customers is transferred to the block queue, where the consensus process is carried out at a service rate denoted by $\mu_2$.

To account for customer impatience, we assume that customers in the customer queue may leave the system if they wait too long. The patience time is assumed to follow an exponential distribution, with impatience rates denoted by $\gamma$. Customers in the block queue are assumed to be committed and will not abandon once service has started.

In addition, we consider the operational reliability of the system by incorporating the possibility of the system state alternating between ON and OFF periods. During ON periods, both block generation and consensus operations are allowed to proceed, while during OFF periods, these operations are suspended. The durations of both ON and OFF periods are exponentially distributed. The transition rates from ON to OFF and from OFF to ON are given by $\alpha$ and $\beta$, respectively.

The $\lambda_{eff}$ and $\mu_{eff}$ of this scenario is shown as below, which $P_e = P_b + (1 - P_b)P_{im}$:

$$\lambda_{eff} = \lambda(1 - P_e) \tag{3-45}$$

$$\mu_{eff} = b\left(\frac{1}{\mu_1} + \frac{1}{\mu_2}\right)^{-1} \frac{\beta}{\alpha + \beta} \tag{3-46}$$

### 3.3.1. State Balance Equations

The system under consideration is described as a three-dimensional Markov chain with state denoted by $(i, x, k)$, where $i$ denotes the number of customers in the customer queue, $x$ denotes the number of customers in the block queue, and $k$

denotes the system state. When $x = 0$, the maximum number of customers in the customer queue is $N$. When $1 \le x \le b$, meaning that the block queue is occupied, the maximum number of customers allowed in the customer queue is reduced to $N - b$.

In this scenario, customers may abandon the customer queue if their waiting time exceeds a certain threshold. The patience time is assumed to follow an exponential distribution with rate $\gamma$, and abandonment occurs only while the customer is waiting in the customer queue.

The system state $k = 1$ indicates that the system is in the ON state, where customers can enter the customer queue and both block generation and consensus operations can proceed. On the other hand, when $k = 0$, the system is in the OFF state, during which only customer arrivals to the queue are permitted, while block generation and consensus are suspended. The state space can be denoted as follows:

$$S = \left\{(i,x,k) \middle| \begin{Bmatrix} 0 \le k \le 1 \\ if\ x = 0: 0 \le i \le N \\ if\ 1 < x \le b: 0 \le i \le N - b \end{Bmatrix}\right\} \tag{3-47}$$

Hence, the total number of feasible states is given by:

$$|S| = 2(N+1) + 2b(N-b+1) \tag{3-48}$$

For example, if $N = 50$ and $b = 15$, the number of feasible states is 1182. The steady state probability of state $(i,x,k)$ is denoted as $\pi_{i,x,k}$. In this scenario, the feasible states can be categorized into 16 distinct cases, as described below.

(a) System off, $k = 0$

Case 1: $i = 0, 0 \le x \le b$

$$(\beta + \lambda)\pi_{0,x,0} = \alpha\pi_{0,x,1} + (i+1)\gamma\pi_{1,x,0}$$

Case 2: $1 \le i \le N - b - 1, 0 \le x \le b$

$$(\beta + \lambda + i\gamma)\pi_{i,x,0} = \alpha\pi_{i,x,1} + \lambda\pi_{i-1,x,0} + (i+1)\gamma\pi_{i+1,x,0}$$

Case 3: $i = N - b, x = 0$

$$[\beta + \lambda + (N-b)\gamma]\pi_{N-b,0,0} = \alpha\pi_{N-b,0,1} + \lambda\pi_{N-b-1,0,0} + (N-b+1)\gamma\pi_{N-b+1,0,0}$$

Case 4: $i = N - b, 1 \le x \le b$

$$[\beta + (N-b)\gamma]\pi_{N-b,x,0} = \alpha\pi_{N-b,x,1} + \lambda\pi_{N-b-1,x,0}$$

Case 5: $N-b+1 \le i \le N-1, x = 0$

$$(\beta + \lambda + i\gamma)\pi_{i,0,0} = \alpha\pi_{i,0,1} + \lambda\pi_{i-1,0,0} + (i+1)\gamma\pi_{i+1,0,0}$$

Case 6: $i = N, x = 0$

$$[(\beta + N\gamma)]\pi_{N,0,0} = \alpha\pi_{N,0,1} + \lambda\pi_{N-1,0,0}$$

(b) System on, $k = 1$

Case 7: $i = 0, x = 0$

$$(\alpha + \lambda)\pi_{0,0,1} = \beta\pi_{0,0,0} + \sum_{t=1}^{b} \mu_2\pi_{0,t,1} + (i+1)\pi_{i+1,x,0} + \gamma\pi_{1,0,1}$$

Case 8: $i = 0, 1 \le x \le b$

$$(\alpha + \lambda + \mu_2)\pi_{0,x,1} = \beta\pi_{0,x,0} + \mu_1\pi_{x,0,1} + \gamma\pi_{1,x,1}$$

Case 9: $1 \le i \le N-b-1, x = 0$

$$(\alpha + \lambda + \mu_1 + i\gamma)\pi_{i,0,1} = \beta\pi_{i,0,0} + \lambda\pi_{i-1,0,1} + \sum_{t=1}^{b} \mu_2\pi_{i,t,1} + (i+1)\gamma\pi_{i,0,1}$$

Case 10: $1 \le i \le N-b-1, 1 \le x \le b-1$

$$(\alpha + \lambda + \mu_2 + i\gamma)\pi_{i,x,1} = \beta\pi_{i,x,0} + \lambda\pi_{i-1,x,1} + (i+1)\gamma\pi_{i,x,1}$$

Case 11: $1 \le i \le N-b-1, x = b$

$$(\alpha + \lambda + \mu_2 + i\gamma)\pi_{i,b,1} = \beta\pi_{i,b,0} + \lambda\pi_{i-1,b,1} + \mu_1\pi_{i+b,0,1} + (i+1)\gamma\pi_{i,b,1}$$

Case 12: $i = N-b, x = 0$

$$[\alpha + \lambda + \mu_1 + (N-b)\gamma]\pi_{N-b,0,1}$$
$$= \beta\pi_{N-b,0,0} + \lambda\pi_{N-b-1,0,1} + \sum_{t=1}^{b} \mu_2\pi_{N-b,t,1}$$
$$+ (N-b+1)\gamma\pi_{N-b+1,0,1}$$

Case 13: $i = N-b, 1 \le x \le b-1$

$$[\alpha + \mu_2 + (N-b)\gamma]\pi_{N-b,x,1} = \beta\pi_{N-b,x,0} + \lambda\pi_{N-b-1,x,1}$$

Case 14: $i = N-b, x = b$

$$[\alpha + \mu_2 + (N-b)\gamma]\pi_{N-b,b,1} = \beta\pi_{N-b,b,0} + \lambda\pi_{N-b-1,b,1} + \mu_1\pi_{N,0,1}$$

Case 15: $N-b+1 \le i \le N-1, x = 0$

$$(\alpha + \lambda + \mu_1 + i\gamma)\pi_{i,0,1} = \beta\pi_{i,0,0} + \lambda\pi_{i-1,0,1} + (i+1)\gamma\pi_{i,0,1}$$

Case 16: $i = N, x = 0$

$$(\alpha + \mu_1 + N\gamma)\pi_{N,0,1} = \beta\pi_{N,0,0} + \lambda\pi_{N-1,0,1}$$

Given the large number of equations presented above, it is impractical to illustrate all the corresponding state transition diagrams. Therefore, we focus on a relatively complex case, specifically Case 12, as a representative example, shown in Figure 3-4.

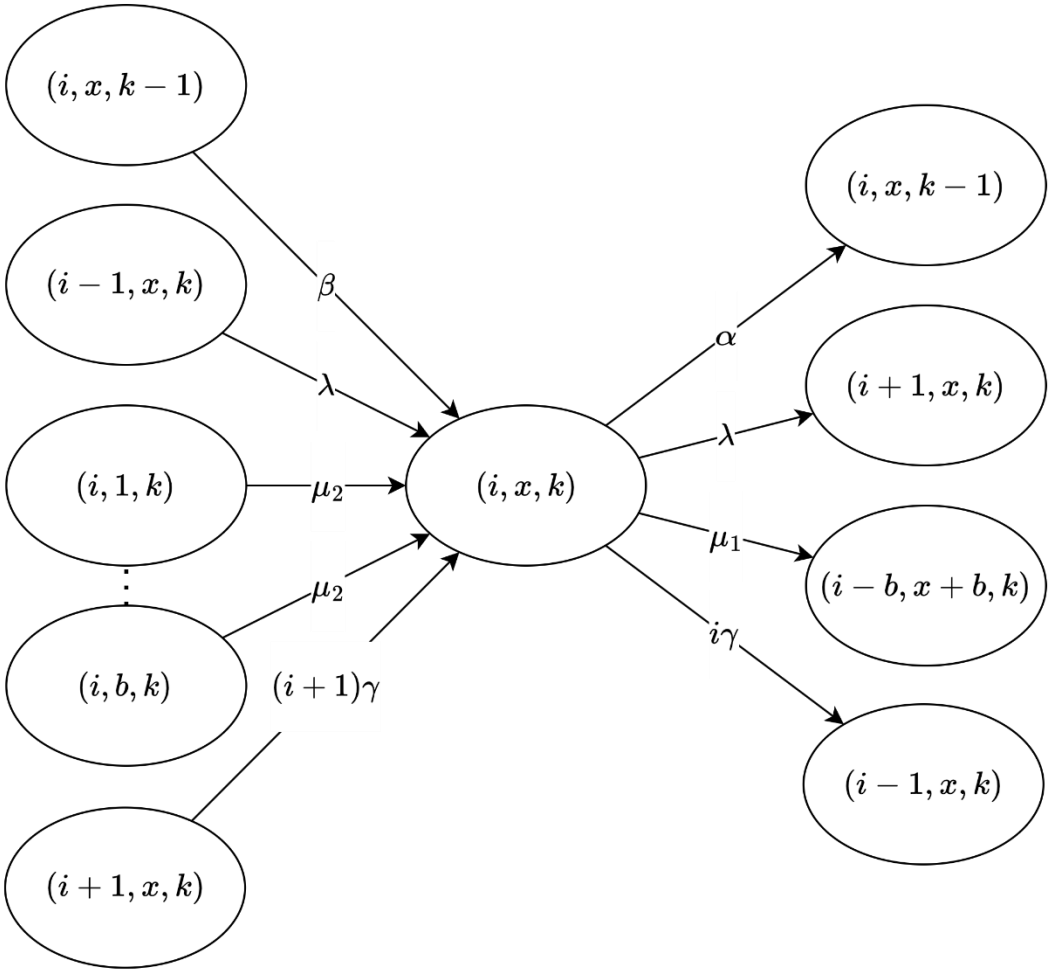


Figure 3-4: The state transition diagram of Case 12: $i = N - b, x = 0, k = 1$

### 3.3.2. Iterative Algorithm

We use the iterative algorithm provided below, and perform calculations on the state balance equations until they converge, allowing us to determine the steady-state distribution of the system.

### Iterative algorithm:

**Step 1**: Initialize $\pi(i,x,k)^{old} = \frac{1}{|s|}$ for all $(i,x,k) \in S$, where $|S|$ is the total number of feasible states.

**Step 2**: Substitute $\pi(i,x,k)^{old}$ into the balance equations from Case 1 to Case 16 to find $\pi(i,x,k)^{new}$, $\forall i,x,k$.

**Step 3**: Normalize $\pi(i,x,k)^{new}$, $\forall i,x,k$.

**Step 4**: If $\sqrt{\sum\sum\sum_{(i,x,k)\in S}|\pi(i,x,k)^{old}-\pi(i,x,k)^{new}|^2}<\varepsilon$, then stop the iteration. Otherwise, set $\pi(i,x,k)^{old}=\pi(i,x,k)^{new},\ \forall i,x,k$, and return to **Step 2**.

In our analysis, the convergence threshold $\varepsilon$ is set to $10^{-8}$, and the algorithm typically converges after about 72 iterations.

### 3.3.1. Performance Measure

After obtaining the steady-state probabilities $\pi_{i,x,k}$ through the iterative algorithm, we proceed to compute several performance metrics to evaluate the effectiveness of the system.

First of all, the average number of customers in the whole system, denoted by $L$, is given by:

$$L=\sum_{k=0}^{1}\sum_{i=1}^{N} i\pi_{i,0,k}+\sum_{k=0}^{1}\sum_{i=0}^{N-b}\sum_{x=1}^{b}(i+x)\pi_{i,x,k} \tag{3-49}$$

Second, the average number of customers in customer queue, denoted by $L_c$, is given by:

$$L_c=\sum_{k=0}^{1}\sum_{i=1}^{N} i\pi_{i,0,k}+\sum_{k=0}^{1}\sum_{i=0}^{N-b}\sum_{x=1}^{b} i\pi_{i,x,k} \tag{3-50}$$

Third, the average number of customers in block queue, denoted by $L_b$, is given by:

$$L_b=\sum_{k=0}^{1}\sum_{i=0}^{N-b}\sum_{x=1}^{b} x\pi_{i,x,k} \tag{3-51}$$

Fourth, the blocking probability of the system, denoted by $P_b$, is given by:

$$P_b=\sum_{k=0}^{1}\pi_{N,0,k}+\sum_{k=0}^{1}\sum_{x=1}^{b}\pi_{N-b,x,k} \tag{3-52}$$

Fifth, the impatient probability of the system, denoted by $P_{im}$, is given by:

$$P_{im} = \frac{\gamma L_c}{\lambda(1 - P_b)} \tag{3-53}$$

Sixth, the throughput of the system, denoted by $T_h$, is given by:

$$T_h = \sum_{i=0}^{N-b} \sum_{x=1}^{b} \mu_2 x \pi_{N-b,x,k} \tag{3-54}$$

Seventh, the average waiting time in the customer queue, denoted by $W_c$, is given by:

$$W_c = \frac{L_c}{\lambda(1 - P_b)} \tag{3-55}$$

Eighth, the average waiting time in the block queue, denoted by $W_b$, is given by:

$$W_b = \frac{L_b}{\lambda(1 - (P_b + (1 - P_b)P_{im}))} \tag{3-56}$$

Ninth, the average waiting time in the system, denoted by $W$, is given by:

$$W = W_c + W_b \tag{3-57}$$

Finally, the average number of blocks participating in the consensus process per unit of time, denoted by $B_n$, is given below.

$$B_n = \sum_{k=0}^{1} \sum_{i=0}^{N-b} \sum_{x=1}^{b} \pi_{i,x,k} \tag{3-58}$$

## 3.4. Scenario 4: Two-Class Customer with Impatience

In this scenario, we consider a two-class customer system without impatience, where the arrivals of high-priority and low-priority customers follow the independent Poisson processes, with arrival rates denoted by $\lambda_H$ and $\lambda_L$, respectively. Customers are served based on the non-preemptive priority discipline, where high-priority customers are placed ahead of low-priority ones in the queue, but ongoing service cannot be interrupted.

The service process is divided into block generation and consensus phases. After arriving at the customer queue, customers wait for the block generation process, which occurs at a rate of $\mu_{1_H}$ and $\mu_{1_L}$ for high-priority and low-priority customers, respectively. Each block is generated according to the partial batch policy, i.e., each block can contain 1 to $b$ customers of the same policy class. Once a block is formed, it is transferred to the block queue, where the consensus process is carried out at the service rate denoted by $\mu_{2_H}$ and $\mu_{2_L}$ for high-priority and low-priority customers, respectively.

To account for customer impatience, we assume that customers in the customer queue may leave the system if they wait too long. The patience time is assumed to follow an exponential distribution, with impatience rates $\gamma_H$ and $\gamma_L$ for high-priority and low-priority customers, respectively. Customers in the block queue are assumed to be committed and will not abandon once service has started.

In addition, we consider the operational reliability of the system by incorporating the possibility of the system state alternating between ON and OFF periods. During ON periods, both block generation and consensus operations are allowed to proceed, while during OFF periods, these operations are suspended. The durations of both ON and OFF periods are exponentially distributed. The transition rates from ON to OFF and from OFF to ON are given by $\alpha$ and $\beta$, respectively.

The $\lambda_{eff}$ and $\mu_{eff}$ of this scenario is shown as below, which $P_e = P_b + (1 - P_b)P_{im}$:

$$\lambda_{eff} = \lambda(1 - P_e) \tag{3-59}$$

$$\mu_{eff} = b\left(\frac{1}{\mu_1} + \frac{1}{\mu_2}\right)^{-1} \frac{\beta}{\alpha + \beta} \tag{3-60}$$

### 3.4.1. State Balance Equations

The system under consideration is described as a five-dimensional Markov chain denoted by $(i, j, x, y, k)$, where $i$ and $j$ represent the number of high-priority and low-priority customers in the customer queue, respectively. $x$ and $y$ represent the number of high-priority and low-priority customers in the block queue, respectively. And $k$ denotes the system state. When the block queue is empty (i.e., $x = 0$ and $y = 0$), the maximum number of customers allowed in the customer queue is $N$, implying that $i + j \leq N$. However, when the block queue is occupied (i.e., $x > 0$ or $y > 0$), the maximum number of customers in the customer queue is reduced to $N - b$, and thus $i + j \leq N - b$.

Customers are scheduled according to the non-preemptive priority discipline, where high-priority customers are always placed ahead of low-priority ones in the queue, but service already in progress cannot be interrupted. When a block is generated, it must contain customer(s) of only one priority class, and is transferred into the block queue as a batch for processing without preemption.

The system state $k = 1$ indicates that the system is in the ON state, where customers can enter the customer queue and both block generation and consensus operations can proceed. On the other hand, when $k = 0$, the system is in the OFF state, during which only customer arrivals to the queue are permitted, while block generation and consensus are suspended. The state space can be denoted as follows:

$$S = \left\{ (i, j, x, y, k) \middle| \begin{Bmatrix} 0 \leq i \leq N, 0 \leq j \leq N - b \\ if\ x = 0\ and\ y = 0 : i + j \leq N \\ if\ 0 < x \leq b\ or\ 0 < y \leq b : i + j \leq N - b \end{Bmatrix} \right\} \tag{3-61}$$

Hence, the total number of feasible states is given by:

$$|S| = 2\left[(N+1)(N-b+1) - \frac{(N-b)(N-b+1)}{2}\right] + \frac{2b(N-b+1)(N-b+2)}{2} \tag{3-62}$$

For example, if $N = 40$ and $b = 15$, the number of feasible states is 22524. The steady state probability of state $(i, j, x, y, k)$ is denoted as $\pi_{i,j,x,y,k}$. In this scenario, the feasible states can be categorized into 99 distinct cases, as described below.

(a) $x = 0, y = 0, k = 0$

Case 1: $i = 0, j = 0$

$$(\beta + \lambda_H + \lambda_L)\pi_{0,0,0,0,0} = \alpha\pi_{0,0,0,0,1} + \gamma_H\pi_{1,0,0,0,0} + \gamma_L\pi_{0,1,0,0,0}$$

Case 2: $i = 0, 1 \le j \le N - b - 1$

$$(\beta + \lambda_H + \lambda_L + j\gamma_L)\pi_{0,j,0,0,0} = \alpha\pi_{0,j,0,0,1} + \lambda_L\pi_{0,j-1,0,0,0} + \gamma_H\pi_{1,j,0,0,0} + (j+1)\gamma_L\pi_{0,j+1,0,0,0}$$

Case 3: $i = 0, j = N - b$

$$(\beta + \lambda_H + (N-b)\gamma_L)\pi_{0,N-b,0,0,0} = \alpha\pi_{0,N-b,0,0,1} + \lambda_L\pi_{0,N-b-1,0,0,0} + \gamma_H\pi_{1,N-b,0,0,0}$$

Case 4: $1 \le i \le b - 1, j = 0$

$$(\beta + \lambda_H + \lambda_L + i\gamma_H)\pi_{i,0,0,0,0} = \alpha\pi_{i,0,0,0,1} + \lambda_H\pi_{i-1,0,0,0,0} + (i+1)\gamma_H\pi_{i+1,0,0,0,0} + \gamma_L\pi_{i,1,0,0,0}$$

Case 5: $1 \le i \le b - 1, 1 \le j \le N - b - 1$

$$\begin{aligned}(\beta + \lambda_H + \lambda_L + i\gamma_H + j\gamma_L)\pi_{i,j,0,0,0} &= \alpha\pi_{i,j,0,0,1} + \lambda_H\pi_{i-1,j,0,0,0} + \lambda_L\pi_{i,j-1,0,0,0} + (i+1)\gamma_H\pi_{i+1,j,0,0,0} \\ &+ (j+1)\gamma_L\pi_{i,j+1,0,0,0}\end{aligned}$$

Case 6: $1 \le i \le b - 1, j = N - b$

$$\begin{aligned}(\beta + \lambda_H + i\gamma_H + (N-b)\gamma_L)\pi_{i,N-b,0,0,0} &= \alpha\pi_{i,N-b,0,0,1} + \lambda_H\pi_{i-1,N-b,0,0,0} + \lambda_L\pi_{i,N-b-1,0,0,0} \\ &+ (i+1)\gamma_H\pi_{i+1,N-b,0,0,0}\end{aligned}$$

Case 7: $i = b, j = 0$

$$(\beta + \lambda_H + \lambda_L + b\gamma_H)\pi_{b,0,0,0,0} = \alpha\pi_{b,0,0,0,1} + \lambda_H\pi_{b-1,0,0,0,0} + (b+1)\gamma_H\pi_{b+1,0,0,0,0} + \gamma_L\pi_{b,1,0,0,0}$$

Case 8: $i = b, 1 \le j \le N - b - 1$

$$\begin{aligned}(\beta + \lambda_H + \lambda_L + b\gamma_H + j\gamma_L)\pi_{b,j,0,0,0} &= \alpha\pi_{b,j,0,0,1} + \lambda_H\pi_{b-1,j,0,0,0} + \lambda_L\pi_{b,j-1,0,0,0} + (b+1)\gamma_H\pi_{b+1,j,0,0,0} \\ &+ (j+1)\gamma_L\pi_{b,j+1,0,0,0}\end{aligned}$$

Case 9: $i = b, j = N - b$

$$(\beta + b\gamma_H + (N-b)\gamma_L)\pi_{b,N-b,0,0,0} = \alpha\pi_{b,N-b,0,0,1} + \lambda_H\pi_{b-1,N-b,0,0,0} + \lambda_L\pi_{b,N-b-1,0,0,0}$$

Case 10: $b+1 \le i \le N-b, j = 0$

$$(\beta + \lambda_H + \lambda_L + i\gamma_H)\pi_{i,0,0,0,0} = \alpha\pi_{i,0,0,0,1} + \lambda_H\pi_{i-1,0,0,0,0} + (i+1)\gamma_H\pi_{i+1,0,0,0,0} + \gamma_L\pi_{i,1,0,0,0}$$

Case 11: $b+1 \le i \le N-b, 1 \le j \le N-1-i$

$$(\beta + \lambda_H + \lambda_L + i\gamma_H + j\gamma_L)\pi_{i,j,0,0,0} = \alpha\pi_{i,j,0,0,1} + \lambda_H\pi_{i-1,j,0,0,0} + \lambda_L\pi_{i,j-1,0,0,0} + (i+1)\gamma_H\pi_{i+1,j,0,0,0} + (j+1)\gamma_L\pi_{i,j+1,0,0,0}$$

Case 12: $b+1 \le i \le N-b, j = N-i$

$$(\beta + i\gamma_H + j\gamma_L)\pi_{i,N-i,0,0,0} = \alpha\pi_{i,N-i,0,0,1} + \lambda_H\pi_{i-1,N-i,0,0,0} + \lambda_L\pi_{i,N-i-1,0,0,0}$$

Case 13: $N-b+1 \le i \le N-2, j = 0$

$$(\beta + \lambda_H + \lambda_L + i\gamma_H)\pi_{i,0,0,0,0} = \alpha\pi_{i,0,0,0,1} + \lambda_H\pi_{i-1,0,0,0,0} + (i+1)\gamma_H\pi_{i+1,0,0,0,0} + \gamma_L\pi_{i,1,0,0,0}$$

Case 14: $N-b+1 \le i \le N-2, 1 \le j \le N-1-i$

$$(\beta + \lambda_H + \lambda_L + i\gamma_H + j\gamma_L)\pi_{i,j,0,0,0} = \alpha\pi_{i,j,0,0,1} + \lambda_H\pi_{i-1,j,0,0,0} + \lambda_L\pi_{i,j-1,0,0,0} + (i+1)\gamma_H\pi_{i+1,j,0,0,0} + (j+1)\gamma_L\pi_{i,j+1,0,0,0}$$

Case 15: $N-b+1 \le i \le N-2, j = N-i$

$$(\beta + i\gamma_H + j\gamma_L)\pi_{i,N-i,0,0,0} = \alpha\pi_{i,N-i,0,0,1} + \lambda_H\pi_{i-1,N-i,0,0,0} + \lambda_L\pi_{i,N-i-1,0,0,0}$$

Case 16: $i = N-1, j = 0$

$$(\beta + \lambda_H + \lambda_L + (N-1)\gamma_H)\pi_{N-1,0,0,0,0} = \alpha\pi_{N-1,0,0,0,1} + \lambda_H\pi_{N-2,0,0,0,0} + N\gamma_H\pi_{N,0,0,0,0} + 1\gamma_L\pi_{N-1,1,0,0,0}$$

Case 17: $i = N-1, j = 1$

$$(\beta + (N-1)\gamma_H + \gamma_L)\pi_{N-1,1,0,0,0} = \alpha\pi_{N-1,1,0,0,1} + \lambda_H\pi_{N-2,1,0,0,0} + \lambda_L\pi_{N-1,0,0,0,0}$$

Case 18: $i = N, j = 0$

$$(\beta + N\gamma_H)\pi_{N,0,0,0,0} = \alpha\pi_{N,0,0,0,1} + \lambda_H\pi_{N-1,0,0,0,0}$$

(b) $1 \le x \le b, y = 0, k = 0$

Case 19: $i = 0, j = 0$

$$(\beta + \lambda_H + \lambda_L)\pi_{0,0,x,0,0} = \alpha\pi_{0,0,x,0,1} + \gamma_H\pi_{1,0,x,0,0} + \gamma_L\pi_{0,1,x,0,0}$$

Case 20: $i = 0, 1 \le j \le N - b - 1$

$$(\beta + \lambda_H + \lambda_L + j\gamma_L)\pi_{0,j,x,0,0} = \alpha\pi_{0,j,x,0,1} + \lambda_L\pi_{0,j-1,x,0,0} + \gamma_H\pi_{1,j,x,0,0} + (j+1)\gamma_L\pi_{0,j+1,x,0,0}$$

Case 21: $i = 0, j = N - b$

$$(\beta + (N-b)\gamma_L)\pi_{0,N-b,x,0,0} = \alpha\pi_{0,N-b,x,0,1} + \lambda_L\pi_{0,N-b-1,x,0,0}$$

Case 22: $1 \le i \le N - b - 2, j = 0$

$$(\beta + \lambda_H + \lambda_L + i\gamma_H)\pi_{i,0,x,0,0} = \alpha\pi_{i,0,x,0,1} + \lambda_H\pi_{i-1,0,x,0,0} + (i+1)\gamma_H\pi_{i+1,0,x,0,0} + \gamma_L\pi_{i,1,x,0,0}$$

Case 23: $1 \le i \le N - b - 2, 1 \le j \le N - b - 1 - i$

$$(\beta + \lambda_H + \lambda_L + i\gamma_H + j\gamma_L)\pi_{i,j,x,0,0} = \alpha\pi_{i,j,x,0,1} + \lambda_H\pi_{i-1,j,x,0,0} + \lambda_L\pi_{i,j-1,x,0,0} + (i+1)\gamma_H\pi_{i+1,j,x,0,0} + (j+1)\gamma_L\pi_{i,j+1,x,0,0}$$

Case 24: $1 \le i \le N - b - 2, j = N - b - i$

$$(\beta + i\gamma_H + (N-b-i)\gamma_L)\pi_{i,N-b-i,x,0,0} = \alpha\pi_{i,N-b-i,x,0,1} + \lambda_H\pi_{i-1,N-b-i,x,0,0} + \lambda_L\pi_{i,N-b-i-1,x,0,0}$$

Case 25: $i = N - b - 1, j = 0$

$$(\beta + \lambda_H + \lambda_L + (N-b-1)\gamma_H)\pi_{N-b-1,0,x,0,0} = \alpha\pi_{N-b-1,0,x,0,1} + \lambda_H\pi_{N-b-2,0,x,0,0} + (N-b)\gamma_H\pi_{N-b,x,0,0,0} + \gamma_L\pi_{N-b-1,1,x,0,0}$$

Case 26: $i = N - b - 1, j = 1$

$$(\beta + (N-b-1)\gamma_H + 1\gamma_L)\pi_{N-b-1,1,x,0,0} = \alpha\pi_{N-b-1,1,x,0,1} + \lambda_H\pi_{N-b-2,1,x,0,0} + \lambda_L\pi_{N-b-1,0,x,0,0}$$

Case 27: $i = N-b, j = 0$

$$(\beta + (N-b)\gamma_H)\pi_{N-b,0,x,0,0} = \alpha\pi_{N-b,0,x,0,1} + \lambda_H\pi_{N-b-1,0,x,0,0}$$

(c) $x = 0, 1 \le y \le b, k = 0$

Case 28: $i = 0, j = 0$

$$(\beta + \lambda_H + \lambda_L)\pi_{0,0,x,0,0} = \alpha\pi_{0,0,x,0,1} + \gamma_{H\pi_{1,0,x,0,0}} + \gamma_L\pi_{0,1,x,0,0}$$

Case 29: $i = 0, 1 \le j \le N-b-1$

$$(\beta + \lambda_H + \lambda_L + j\gamma_L)\pi_{0,j,x,0,0} = \alpha\pi_{0,j,x,0,1} + \lambda_{L\pi_{0,j-1,x,0,0}} + \gamma_H\pi_{1,j,x,0,0} + (j+1)\gamma_L\pi_{0,j+1,x,0,0}$$

Case 30: $i = 0, j = N-b$

$$(\beta + j\gamma_L)\pi_{0,N-b,x,0,0} = \alpha\pi_{0,N-b,x,0,1} + \lambda_L\pi_{0,N-b-1,x,0,0}$$

Case 31: $1 \le i \le N-b-2, j = 0$

$$(\beta + \lambda_H + \lambda_L + i\gamma_H)\pi_{i,0,x,0,0} = \alpha\pi_{i,0,x,0,1} + \lambda_H\pi_{i-1,0,x,0,0} + (i+1)\gamma_H\pi_{i+1,0,x,0,0} + \gamma_L\pi_{i,1,x,0,0}$$

Case 32: $1 \le i \le N-b-2, 1 \le j \le N-b-1-i$

$$(\beta + \lambda_H + \lambda_L + i\gamma_H + j\gamma_L)\pi_{i,j,x,0,0} = \alpha\pi_{i,j,x,0,1} + \lambda_H\pi_{i-1,j,x,0,0} + \lambda_L\pi_{i,j-1,x,0,0} + (i+1)\gamma_H\pi_{i+1,j,x,0,0} + (j+1)\gamma_L\pi_{i,j+1,x,0,0}$$

Case 33: $1 \le i \le N-b-2, j = N-b-i$

$$(\beta + i\gamma_H + j\gamma_L)\pi_{i,N-b-i,x,0,0} = \alpha\pi_{i,N-b-i,x,0,1} + \lambda_H\pi_{i-1,N-b-i,x,0,0} + \lambda_L\pi_{i,N-b-i-1,x,0,0}$$

Case 34: $i = N-b-1, j = 0$

$$(\beta + \lambda_H + \lambda_L + i\gamma_H)\pi_{i,0,x,0,0} = \alpha\pi_{i,0,x,0,1} + \lambda_H\pi_{i-1,0,x,0,0} + (i+1)\gamma_H\pi_{i+1,0,x,0,0} + \gamma_L\pi_{i,1,x,0,0}$$

Case 35: $i = N - b - 1, j = 1$

$$(\beta + i\gamma_H + j\gamma_L)\pi_{i,1,x,0,0} = \alpha\pi_{i,1,x,0,1} + \lambda_H\pi_{i-1,1,x,0,0} + \lambda_L\pi_{i,0,x,0,0}$$

Case 36: $i = N - b, j = 0$

$$(\beta + i\gamma_H)\pi_{i,0,x,0,0} = \alpha\pi_{i,0,x,0,1} + \lambda_H\pi_{i-1,0,x,0,0}$$

(d) $x = 0, y = 0, k = 1$

Case 37: $i = 0, j = 0$

$$(\beta + \lambda_H + \lambda_L)\pi_{0,0,0,0,1} = \alpha\pi_{0,0,0,0,0} + \sum_{t=1}^{b} \mu_{2_H}\pi_{0,0,t,0,1} + \sum_{t=1}^{b} \mu_{2_L}\pi_{0,0,0,t,1} + \gamma_H\pi_{1,0,0,0,1} + \gamma_L\pi_{0,1,0,0,1}$$

Case 38: $i = 0, 1 \le j \le N - b - 1$

$$(\beta + \lambda_H + \lambda_L + \mu_{1_L} + j\gamma_L)\pi_{0,j,0,0,1} = \alpha\pi_{0,j,0,0,0} + \lambda_L\pi_{0,j-1,0,0,1} + \sum_{t=1}^{b} \mu_{2_H}\pi_{0,j,t,0,1} + \sum_{t=1}^{b} \mu_{2_L}\pi_{0,j,0,t,1} + \gamma_H\pi_{1,j,0,0,1} + (j+1)\gamma_L\pi_{0,j+1,0,0,1}$$

Case 39: $i = 0, j = N - b$

$$(\beta + \lambda_H + \mu_{1_L} + j\gamma_L)\pi_{0,N-b,0,0,1} = \alpha\pi_{0,N-b,0,0,0} + \lambda_L\pi_{0,N-b-1,0,0,1} + \sum_{t=1}^{b} \mu_{2_H}\pi_{0,N-b,t,0,1} + \sum_{t=1}^{b} \mu_{2_L}\pi_{0,N-b,0,t,1} + \gamma_H\pi_{1,N-b,0,0,1}$$

Case 40: $1 \le i \le b - 1, j = 0$

$$(\beta + \lambda_H + \lambda_L + \mu_{1_H} + i\gamma_H)\pi_{i,0,0,0,1}$$

$$= \alpha\pi_{i,0,0,0,0} + \lambda_H\pi_{i-1,0,0,0,1} + \sum_{t=1}^{b} \mu_{2_H}\pi_{i,0,t,0,1} + \sum_{t=1}^{b} \mu_{2_L}\pi_{i,0,0,t,1}$$

$$+ (i+1)\gamma_H\pi_{i+1,0,0,0,1} + \gamma_L\pi_{i,1,0,0,1}$$

Case 41: $1 \le i \le b-1, 1 \le j \le N-b-i$

$$(\beta + \lambda_H + \lambda_L + \mu_{1_H} + i\gamma_H + j\gamma_L)\pi_{i,j,0,0,1}$$

$$= \alpha\pi_{i,j,0,0,0} + \lambda_H\pi_{i-1,j,0,0,1} + \lambda_L\pi_{i,j-1,0,0,1} + \sum_{t=1}^{b} \mu_{2_H}\pi_{i,j,t,0,1}$$

$$+ \sum_{t=1}^{b} \mu_{2_L}\pi_{i,j,0,t,1} + (i+1)\gamma_H\pi_{i+1,j,0,0,1} + (j+1)\gamma_L\pi_{i,j+1,0,0,1}$$

Case 42: $1 \le i \le b-1, N-b+1-i \le j \le N-b-1$

$$(\beta + \lambda_H + \lambda_L + \mu_{1_H} + i\gamma_H + j\gamma_L)\pi_{i,j,0,0,1}$$

$$= \alpha\pi_{i,j,0,0,0} + \lambda_H\pi_{i-1,j,0,0,1} + \lambda_L\pi_{i,j-1,0,0,1} + (i+1)\gamma_H\pi_{i+1,j,0,0,1}$$

$$+ (j+1)\gamma_L\pi_{i,j+1,0,0,1}$$

Case 43: $1 \le i \le b-1, j = N-b$

$$(\beta + \lambda_H + \mu_{1_H} + i\gamma_H + j\gamma_L)\pi_{i,N-b,0,0,1}$$

$$= \alpha\pi_{i,N-b,0,0,0} + \lambda_H\pi_{i-1,N-b,0,0,1} + \lambda_L\pi_{i,N-b-1,0,0,1}$$

$$+ (i+1)\gamma_H\pi_{i+1,N-b,0,0,1}$$

Case 44: $i = b, j = 0$

$$(\beta + \lambda_H + \lambda_L + \mu_{1_H} + i\gamma_H)\pi_{2,0,0,0,1}$$

$$= \alpha\pi_{2,0,0,0,0} + \lambda_H\pi_{2-1,0,0,0,1} + \sum_{t=1}^{b} \mu_{2_H}\pi_{2,0,t,0,1} + \sum_{t=1}^{b} \mu_{2_L}\pi_{2,0,0,t,1}$$

$$+ (i+1)\gamma_H\pi_{2+1,0,0,0,1} + \gamma_L\pi_{2,1,0,0,1}$$

Case 45: $i = b, 1 \le j \le N-b-i$

$$\left(\beta + \lambda_H + \lambda_L + \mu_{1_H} + i\gamma_H + j\gamma_L\right)\pi_{2,j,0,0,1}$$

$$= \alpha\pi_{2,j,0,0,0} + \lambda_H\pi_{2-1,j,0,0,1} + \lambda_L\pi_{2,j-1,0,0,1} + \sum_{t=1}^{b}\mu_{2_H}\pi_{2,j,t,0,1}$$

$$+ \sum_{t=1}^{b}\mu_{2_L}\pi_{2,j,0,t,1} + (b+1)\gamma_H\pi_{2+1,j,0,0,1} + (j+1)\gamma_L\pi_{2,j+1,0,0,1}$$

Case 46: $i = b, N - b + 1 - b \le j \le N - b - 1$

$$\left(\beta + \lambda_H + \lambda_L + \mu_{1_H} + b\gamma_H + j\gamma_L\right)\pi_{2,j,0,0,1}$$
$$= \alpha\pi_{2,j,0,0,0} + \lambda_H\pi_{2-1,j,0,0,1} + \lambda_L\pi_{2,j-1,0,0,1} + (b+1)\gamma_H\pi_{2+1,j,0,0,1}$$
$$+ (j+1)\gamma_L\pi_{2,j+1,0,0,1}$$

Case 47: $i = b, j = N - b$

$$\left(\beta + \mu_{1_H} + b\gamma_H + j\gamma_L\right)\pi_{2,N-b,0,0,1}$$
$$= \alpha\pi_{2,N-b,0,0,0} + \lambda_H\pi_{2-1,N-b,0,0,1} + \lambda_L\pi_{2,N-b-1,0,0,1}$$

Case 48: $b + 1 \le i \le N - b - 1, j = 0$

$$\left(\beta + \lambda_H + \lambda_L + \mu_{1_H} + i\gamma_H\right)\pi_{i,0,0,0,1}$$

$$= \alpha\pi_{i,0,0,0,0} + \lambda_H\pi_{i-1,0,0,0,1} + \sum_{t=1}^{b}\mu_{2_H}\pi_{i,0,t,0,1} + \sum_{t=1}^{b}\mu_{2_L}\pi_{i,0,0,t,1}$$

$$+ (i+1)\gamma_H\pi_{i+1,0,0,0,1} + \gamma_L\pi_{i,1,0,0,1}$$

Case 49: $b + 1 \le i \le N - b - 1, 1 \le j \le N - b - i$

$$\left(\beta + \lambda_H + \lambda_L + \mu_{1_H} + i\gamma_H + j\gamma_L\right)\pi_{i,j,0,0,1}$$

$$= \alpha\pi_{i,j,0,0,0} + \lambda_H\pi_{i-1,j,0,0,1} + \lambda_L\pi_{i,j-1,0,0,1} + \sum_{t=1}^{b}\mu_{2_H}\pi_{i,j,t,0,1}$$

$$+ \sum_{t=1}^{b}\mu_{2_L}\pi_{i,j,0,t,1} + (i+1)\gamma_H\pi_{i+1,j,0,0,1} + (j+1)\gamma_L\pi_{i,j+1,0,0,1}$$

Case 50: $b + 1 \le i \le N - b - 1, N - b + 1 - i \le j \le N - 1 - i$

$$(\beta + \lambda_H + \lambda_L + \mu_{1_H} + i\gamma_H + j\gamma_L)\pi_{i,j,0,0,1} = \alpha\pi_{i,j,0,0,0} + \lambda_H\pi_{i-1,j,0,0,1} + \lambda_L\pi_{i,j-1,0,0,1} + (i+1)\gamma_H\pi_{i+1,j,0,0,1} + (j+1)\gamma_L\pi_{i,j+1,0,0,1}$$

Case 51: $b+1 \le i \le N-b-1, j = N-i$

$$(\beta + \mu_{1_H} + i\gamma_H + j\gamma_L)\pi_{i,N-i,0,0,1} = \alpha\pi_{i,N-i,0,0,0} + \lambda_H\pi_{i-1,N-i,0,0,1} + \lambda_L\pi_{i,N-i-1,0,0,1}$$

Case 52: $i = N-b, j = 0$

$$(\beta + \lambda_H + \lambda_L + \mu_{1_H} + (N-b)\gamma_H)\pi_{N-b,0,0,0,1} = \alpha\pi_{N-b,0,0,0,0} + \lambda_H\pi_{N-b-1,0,0,0,1} + \sum_{t=1}^{b}\mu_{2_H}\pi_{N-b,0,t,0,1} + \sum_{t=1}^{b}\mu_{2_L}\pi_{N-b,0,0,t,1} + (N-b+1)\gamma_H\pi_{N-b+1,0,0,0,1} + \gamma_L\pi_{N-b,1,0,0,1}$$

Case 53: $i = N-b, 1 \le j \le b-1$

$$(\beta + \lambda_H + \lambda_L + \mu_{1_H} + (N-b)\gamma_H + j\gamma_L)\pi_{N-b,j,0,0,1} = \alpha\pi_{N-b,j,0,0,0} + \lambda_H\pi_{N-b-1,j,0,0,1} + \lambda_L\pi_{N-b,j-1,0,0,1} + (N-b+1)\gamma_H\pi_{N-b+1,j,0,0,1} + (j+1)\gamma_L\pi_{N-b,j+1,0,0,1}$$

Case 54: $i = N-b, j = b$

$$(\beta + \mu_{1_H} + (N-b)\gamma_H + j\gamma_L)\pi_{N-b,b,0,0,1} = \alpha\pi_{N-b,b,0,0,0} + \lambda_H\pi_{N-b-1,b,0,0,1} + \lambda_L\pi_{N-b,b-1,0,0,1}$$

Case 55: $N-b+1 \le i \le N-2, j = 0$

$$(\beta + \lambda_H + \lambda_L + \mu_{1_H} + i\gamma_H)\pi_{i,0,0,0,1} = \alpha\pi_{i,0,0,0,0} + \lambda_H\pi_{i-1,0,0,0,1} + (i+1)\gamma_H\pi_{i+1,0,0,0,1} + \gamma_L\pi_{i,1,0,0,1}$$

Case 56: $N-b+1 \le i \le N-2, 1 \le j \le N-1-i$

$$(\beta + \lambda_H + \lambda_L + \mu_{1_H} + i\gamma_H + j\gamma_L)\pi_{i,j,0,0,1} = \alpha\pi_{i,j,0,0,0} + \lambda_H\pi_{i-1,j,0,0,1} + \lambda_L\pi_{i,j-1,0,0,1} + (i+1)\gamma_H\pi_{i+1,j,0,0,1} + (j+1)\gamma_L\pi_{i,j+1,0,0,1}$$

Case 57: $N-b+1 \le i \le N-2, j = N-i$

$$(\beta + \mu_{1_H} + i\gamma_H + j\gamma_L)\pi_{i,N-i,0,0,1} = \alpha\pi_{i,N-i,0,0,0} + \lambda_H\pi_{i-1,N-i,0,0,1} + \lambda_L\pi_{i,N-i-1,0,0,1}$$

Case 58: $i = N-1, j = 0$

$$(\beta + \lambda_H + \lambda_L + \mu_{1_H} + (N-1)\gamma_H)\pi_{N-1,0,0,0,1} = \alpha\pi_{N-1,0,0,0,0} + \lambda_H\pi_{N-1-1,0,0,0,1} + N\gamma_H\pi_{N-1+1,0,0,0,1} + \gamma_L\pi_{N-1,1,0,0,1}$$

Case 59: $i = N-1, j = 1$

$$(\beta + \mu_{1_H} + (N-1)\gamma_H + \gamma_L)\pi_{N-1,1,0,0,1} = \alpha\pi_{N-1,1,0,0,0} + \lambda_H\pi_{N-1-1,1,0,0,1} + \lambda_L\pi_{N-1,0,0,0,1}$$

Case 60: $i = N, j = 0$

$$(\beta + \mu_{1_H} + N\gamma_H)\pi_{N,0,0,0,1} = \alpha\pi_{N,0,0,0,0} + \lambda_H\pi_{N-1,0,0,0,1}$$

(e) $1 \le x < b, y = 0, k = 1$

Case 61: $i = 0, j = 0$

$$(\beta + \lambda_H + \lambda_L + \mu_{2_H})\pi_{0,0,x,0,1} = \alpha\pi_{0,0,x,0,0} + \mu_{1_H}\pi_{x,0,0,0,1} + \gamma_H\pi_{1,0,x,0,1} + \gamma_L\pi_{0,1,x,0,1}$$

Case 62: $i = 0, 1 \le j \le N-b-1$

$$(\beta + \lambda_H + \lambda_L + \mu_{2_H} + j\gamma_L)\pi_{0,j,x,0,1} = \alpha\pi_{0,j,x,0,0} + \lambda_L\pi_{0,j-1,x,0,1} + \mu_{1_H}\pi_{x,j,0,0,1} + \gamma_H\pi_{1,j,x,0,1} + (j+1)\gamma_L\pi_{0,j+1,x,0,1}$$

Case 63: $i = 0, j = N-b$

$$(\beta + \mu_{2_H} + j\gamma_L)\pi_{0,N-b,x,0,1} = \alpha\pi_{0,N-b,x,0,0} + \lambda_L\pi_{0,N-b-1,x,0,1} + \mu_{1_H}\pi_{x,N-b,0,0,1}$$

Case 64: $1 \le i \le N-b-2, j = 0$

$$(\beta + \lambda_H + \lambda_L + \mu_{2_H} + i\gamma_H)\pi_{i,0,x,0,1} = \alpha\pi_{i,0,x,0,0} + \lambda_H\pi_{i-1,0,x,0,1} + (i+1)\gamma_H\pi_{i+1,0,x,0,1} + \gamma_L\pi_{i,1,x,0,1}$$

Case 65: $1 \le i \le N-b-2, 1 \le j \le N-b-1-i$

$$(\beta + \lambda_H + \lambda_L + \mu_{2_H} + i\gamma_H + j\gamma_L)\pi_{i,j,x,0,1} = \alpha\pi_{i,j,x,0,0} + \lambda_H\pi_{i-1,j,x,0,1} + \lambda_L\pi_{i,j-1,x,0,1} + (i+1)\gamma_H\pi_{i+1,j,x,0,1} + (j+1)\gamma_L\pi_{i,j+1,x,0,1}$$

Case 66: $1 \le i \le N-b-2, j = N-b-i$

$$(\beta + \mu_{2_H} + i\gamma_H + j\gamma_L)\pi_{i,N-b-i,x,0,1} = \alpha\pi_{i,N-b-i,x,0,0} + \lambda_H\pi_{i-1,N-b-i,x,0,1} + \lambda_L\pi_{i,N-b-i-1,x,0,1}$$

Case 67: $i = N-b-1, j = 0$

$$(\beta + \lambda_H + \lambda_L + \mu_{2_H} + i\gamma_H)\pi_{N-b-1,0,x,0,1} = \alpha\pi_{N-b-1,0,x,0,0} + \lambda_H\pi_{N-b-2,0,x,0,1} + (N-b)\gamma_H\pi_{N-b,x,0,1} + \gamma_L\pi_{N-b-1,1,x,0,1}$$

Case 68: $i = N-b-1, j = 1$

$$(\beta + \mu_{2_H} + i\gamma_H + j\gamma_L)\pi_{N-b-1,1,x,0,1} = \alpha\pi_{N-b-1,1,x,0,0} + \lambda_H\pi_{N-b-2,1,x,0,1} + \lambda_L\pi_{N-b-1,0,x,0,1}$$

Case 69: $i = N-b, j = 0$

$$(\beta + \mu_{2_H} + i\gamma_H)\pi_{N-b,0,x,0,1} = \alpha\pi_{N-b,0,x,0,0} + \lambda_H\pi_{N-b-1,0,x,0,1}$$

(f) $x = b, y = 0, k = 1$

Case 70: $i = 0, j = 0$

$$(\beta + \lambda_H + \lambda_L + \mu_{2_H})\pi_{0,0,b,0,1} = \alpha\pi_{0,0,b,0,0} + \mu_{1_H}\pi_{2,0,0,0,1} + \gamma_H\pi_{1,0,b,0,1} + \gamma_L\pi_{0,1,b,0,1}$$

Case 71: $i = 0, 1 \le j \le N-b-1$

$$(\beta + \lambda_H + \lambda_L + \mu_{2_H} + j\gamma_L)\pi_{0,j,b,0,1} = \alpha\pi_{0,j,b,0,0} + \lambda_L\pi_{0,j-1,b,0,1} + \mu_{1_H}\pi_{2,j,0,0,1} + \gamma_H\pi_{1,j,b,0,1} + (j+1)\gamma_L\pi_{0,j+1,b,0,1}$$

Case 72: $i = 0, j = N-b$

$$(\beta + \mu_{2_H} + j\gamma_L)\pi_{0,N-b,b,0,1} = \alpha\pi_{0,N-b,b,0,0} + \lambda_L\pi_{0,N-b-1,b,0,1} + \mu_{1_H}\pi_{2,N-b,0,0,1}$$

Case 73: $1 \le i \le N-b-2, j = 0$

$$(\beta + \lambda_H + \lambda_L + \mu_{2_H} + i\gamma_H)\pi_{i,0,b,0,1} = \alpha\pi_{i,0,b,0,0} + \lambda_H\pi_{i-1,0,b,0,1} + \mu_{1_H}\pi_{i+b,0,0,0,1} + (i+1)\gamma_H\pi_{i+1,0,b,0,1} + \gamma_L\pi_{i,1,b,0,1}$$

Case 74: $1 \le i \le N-b-2, 1 \le j \le N-b-1-i$

$$(\beta + \lambda_H + \lambda_L + \mu_{2_H} + i\gamma_H + j\gamma_L)\pi_{i,j,b,0,1} = \alpha\pi_{i,j,b,0,0} + \lambda_H\pi_{i-1,j,b,0,1} + \lambda_L\pi_{i,j-1,b,0,1} + \mu_{1_H}\pi_{i+b,j,0,0,1} + (i+1)\gamma_H\pi_{i+1,j,b,0,1} + (j+1)\gamma_L\pi_{i,j+1,b,0,1}$$

Case 75: $1 \le i \le N-b-2, j = N-b-i$

$$(\beta + \mu_{2_H} + i\gamma_H + j\gamma_L)\pi_{i,N-b-i,b,0,1} = \alpha\pi_{i,N-b-i,b,0,0} + \lambda_H\pi_{i-1,N-b-i,b,0,1} + \lambda_L\pi_{i,N-b-i-1,b,0,1} + \mu_{1_H}\pi_{i+b,N-b-i,0,0,1}$$

Case 76: $i = N-b-1, j = 0$

$$(\beta + \lambda_H + \lambda_L + \mu_{2_H} + i\gamma_H)\pi_{N-b-1,0,b,0,1} = \alpha\pi_{N-b-1,0,b,0,0} + \lambda_H\pi_{N-b-2,0,b,0,1} + \mu_{1_H}\pi_{N-1,0,0,0,1} + (N-b)\gamma_H\pi_{N-b,0,b,0,1} + \gamma_L\pi_{N-b-1,1,b,0,1}$$

Case 77: $i = N-b-1, j = 1$

$$(\beta + \mu_{2_H} + i\gamma_H + j\gamma_L)\pi_{N-b-1,1,b,0,1} = \alpha\pi_{N-b-1,1,b,0,0} + \lambda_H\pi_{N-b-2,1,b,0,1} + \lambda_L\pi_{N-b-1,0,b,0,1} + \mu_{1_H}\pi_{N-1,1,0,0,1}$$

Case 78: $i = N-b, j = 0$

$$(\beta + \mu_{2_H} + i\gamma_H)\pi_{N-b,0,b,0,1} = \alpha\pi_{N-b,0,b,0,0} + \lambda_H\pi_{N-b-1,0,b,0,1} + \mu_{1_H}\pi_{N,N-b,0,0,1}$$

(g) $x = 0, 1 \le y < b, k = 1$

Case 79: $i = 0, j = 0$

$$(\beta + \lambda_H + \lambda_L + \mu_{2_L})\pi_{0,0,0,y,1} = \alpha\pi_{0,0,0,y,0} + \mu_{q_L}\pi_{0,y,0,0,1} + \gamma_H\pi_{1,0,0,y,1} + \gamma_L\pi_{0,1,0,y,1}$$

Case 80: $1 \le i \le N-b-1, j = 0$

$$(\beta + \lambda_H + \lambda_L + \mu_{2_L} + i\gamma_H)\pi_{i,0,0,y,1} = \alpha\pi_{i,0,0,y,0} + \lambda_H\pi_{i-1,0,0,y,1} + (i+1)\gamma_H\pi_{i+1,0,0,y,1} + \gamma_L\pi_{i,1,0,y,1}$$

Case 81: $i = N-b, j = 0$

$$(\beta + \mu_{2_L} + i\gamma_H)\pi_{N-b,0,0,y,1} = \alpha\pi_{N-b,0,0,y,0} + \lambda_H\pi_{N-b-1,0,0,y,1}$$

Case 82: $i = 0, 1 \le j \le N - b - 2$

$$(\beta + \lambda_H + \lambda_L + \mu_{2_L} + j\gamma_L)\pi_{0,j,0,y,1} = \alpha\pi_{0,j,0,y,0} + \lambda_L\pi_{0,j-1,0,y,1} + \gamma_H\pi_{1,j,0,y,1} + (j+1)\gamma_L\pi_{0,j+1,0,y,1}$$

Case 83: $1 \le i \le N - b - 1 - j, 1 \le j \le N - b - 2$

$$(\beta + \lambda_H + \lambda_L + \mu_{2_L} + i\gamma_H + j\gamma_L)\pi_{i,j,0,y,1} = \alpha\pi_{i,j,0,y,0} + \lambda_H\pi_{i-1,j,0,y,1} + \lambda_L\pi_{i,j-1,0,y,1} + (i+1)\gamma_H\pi_{i+1,j,0,y,1} + (j+1)\gamma_L\pi_{i,j+1,0,y,1}$$

Case 84: $i = N - b - j, 1 \le j \le N - b - 2$

$$(\beta + \mu_{2_L} + i\gamma_H + j\gamma_L)\pi_{N-b-j,j,0,y,1} = \alpha\pi_{N-b-j,j,0,y,0} + \lambda_H\pi_{N-b-j-1,j,0,y,1} + \lambda_L\pi_{N-b-j,j-1,0,y,1}$$

Case 85: $i = 0, j = N - b - 1$

$$(\beta + \lambda_H + \lambda_L + \mu_{2_L} + j\gamma_L)\pi_{0,N-b-1,0,y,1} = \alpha\pi_{0,N-b-1,0,y,0} + \lambda_L\pi_{0,N-b-2,0,y,1} + \gamma_H\pi_{1,N-b-1,0,y,1} + (N-b)\gamma_L\pi_{0,N-b,0,y,1}$$

Case 86: $i = 1, j = N - b - 1$

$$(\beta + \mu_{2_L} + i\gamma_H + j\gamma_L)\pi_{1,N-b-1,0,y,1} = \alpha\pi_{1,N-b-1,0,y,0} + \lambda_H\pi_{0,N-b-1,0,y,1} + \lambda_L\pi_{1,N-b-2,0,y,1}$$

Case 87: $i = 0, j = N - b$

$$(\beta + \mu_{2_L} + j\gamma_L)\pi_{0,N-b,0,y,1} = \alpha\pi_{0,N-b,0,y,0} + \lambda_L\pi_{0,N-b-1,0,y,1}$$

(h) $x = 0, y = b, k = 1$

Case 88: $i = 0, j = 0$

$$(\beta + \lambda_H + \lambda_L + \mu_{2_L})\pi_{0,0,0,b,1} = \alpha\pi_{0,0,0,b,0} + \mu_{1_L}\pi_{0,b,0,0,1} + \gamma_H\pi_{1,0,0,b,1} + \gamma_L\pi_{0,1,0,b,1}$$

Case 89: $1 \le i \le N - b - 1, j = 0$

$$(\beta + \lambda_H + \lambda_L + \mu_{2_L} + i\gamma_H)\pi_{i,0,0,b,1} = \alpha\pi_{i,0,0,b,0} + \lambda_H\pi_{i-1,0,0,b,1} + (i+1)\gamma_H\pi_{i+1,0,0,b,1} + \gamma_L\pi_{i,1,0,b,1}$$

Case 90: $i = N - b, j = 0$

$$(\beta + \mu_{2_L} + i\gamma_H)\pi_{N-b,0,0,b,1} = \alpha\pi_{N-b,0,0,b,0} + \lambda_H\pi_{N-b-1,0,0,b,1}$$

Case 91: $i = 0, 1 \le j \le N - 2b$

$$\begin{aligned}(\beta + \lambda_H + \lambda_L + \mu_{2_L} + j\gamma_L)\pi_{0,j,0,b,1} &= \alpha\pi_{0,j,0,b,0} + \lambda_L\pi_{0,j-1,0,b,1} + \mu_{1_L}\pi_{0,j+b,0,0,1} + \gamma_H\pi_{1,j,0,b,1} \\ &+ (j+1)\gamma_L\pi_{0,j+1,0,b,1}\end{aligned}$$

Case 92: $1 \le i \le N - b - 1 - j, 1 \le j \le N - 2b$

$$\begin{aligned}(\beta + \lambda_H + \lambda_L + \mu_{2_L} + i\gamma_H + j\gamma_L)\pi_{i,j,0,b,1} &= \alpha\pi_{i,j,0,b,0} + \lambda_H\pi_{i-1,j,0,b,1} + \lambda_L\pi_{i,j-1,0,b,1} + (i+1)\gamma_H\pi_{i+1,j,0,b,1} \\ &+ (j+1)\gamma_L\pi_{i,j+1,0,b,1}\end{aligned}$$

Case 93: $i = N - b - j, 1 \le j \le N - 2b$

$$(\beta + \mu_{2_L} + i\gamma_H + j\gamma_L)\pi_{N-b-j,j,0,b,1} = \alpha\pi_{N-b-j,j,0,b,0} + \lambda_H\pi_{N-b-j-1,j,0,b,1} + \lambda_L\pi_{N-b-j,j-1,0,b,1}$$

Case 94: $i = 0, N - 2b + 1 \le j \le N - b - 2$

$$(\beta + \lambda_H + \lambda_L + \mu_{2_L} + j\gamma_L)\pi_{0,j,0,b,1} = \alpha\pi_{0,j,0,b,0} + \lambda_L\pi_{0,j-1,0,b,1} + \gamma_H\pi_{1,j,0,b,1} + (j+1)\gamma_L\pi_{0,j+1,0,b,1}$$

Case 95: $1 \le i \le N - b - 1 - j, N - 2b + 1 \le j \le N - b - 2$

$$\begin{aligned}(\beta + \lambda_H + \lambda_L + \mu_{2_L} + i\gamma_H + j\gamma_L)\pi_{i,j,0,b,1} &= \alpha\pi_{i,j,0,b,0} + \lambda_H\pi_{i-1,j,0,b,1} + \lambda_L\pi_{i,j-1,0,b,1} + (i+1)\gamma_H\pi_{i+1,j,0,b,1} \\ &+ (j+1)\gamma_L\pi_{i,j+1,0,b,1}\end{aligned}$$

Case 96: $i = N - b - j, N - 2b + 1 \le j \le N - b - 2$

$$(\beta + \mu_{2_L} + i\gamma_H + j\gamma_L)\pi_{N-b-j,j,0,b,1} = \alpha\pi_{N-b-j,j,0,b,0} + \lambda_H\pi_{N-b-j-1,j,0,b,1} + \lambda_L\pi_{N-b-j,j-1,0,b,1}$$

Case 97: $i = 0, j = N - b - 1$

$$(\beta + \lambda_H + \lambda_L + \mu_{2_L} + j\gamma_L)\pi_{0,N-b-1,0,b,1} = \alpha\pi_{0,N-b-1,0,b,0} + \lambda_L\pi_{0,N-b-2,0,b,1} + \gamma_H\pi_{1,N-b-1,0,b,1} + (N-b)\gamma_L\pi_{0,N-b,0,b,1}$$

Case 98: $i = 1, j = N - b - 1$

$$(\beta + \mu_{2_L} + i\gamma_H + j\gamma_L)\pi_{1,N-b-1,0,b,1} = \alpha\pi_{1,N-b-1,0,b,0} + \lambda_H\pi_{0,N-b-1,0,b,1} + \lambda_L\pi_{1,N-b-2,0,b,1}$$

Case 99: $i = 0, j = N - b$

$$(\beta + \mu_{2_L} + j\gamma_L)\pi_{0,N-b,0,b,1} = \alpha\pi_{0,N-b,0,b,0} + \lambda_L\pi_{0,N-b-1,0,b,1}$$

Given the large number of equations presented above, it is impractical to illustrate all the corresponding state transition diagrams. Therefore, we focus on a relatively complex case, specifically Case 45, as a representative example, shown in Figure 3-5.

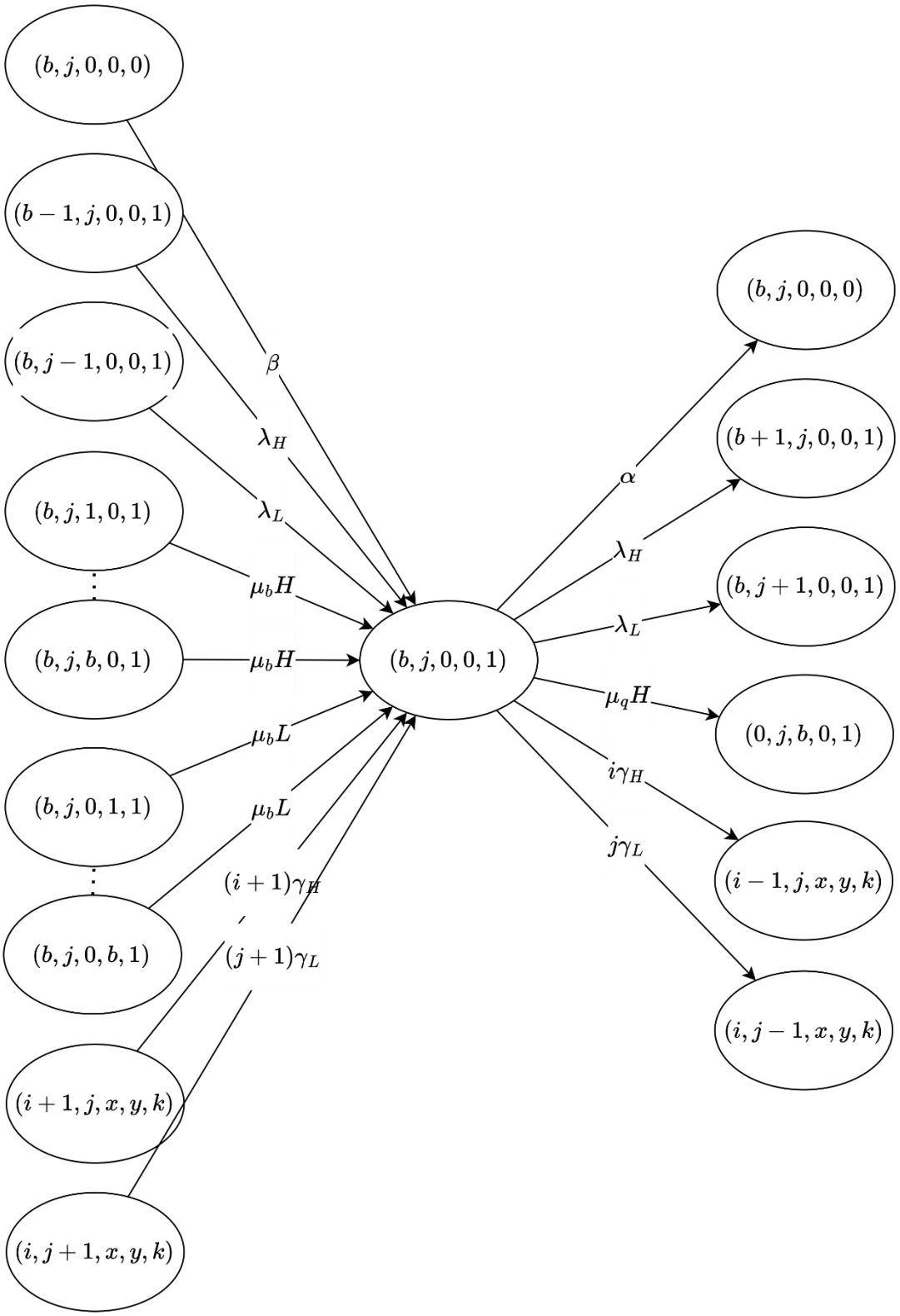


Figure 3-5: The state transition diagram of Case 45: $i = b, 1 \leq j \leq N - b - i, x = 0, y = 0, k = 0$

### 3.4.2. Iterative Algorithm

We use the iterative algorithm provided below, and perform calculations on the state balance equations until they converge, allowing us to determine the steady-state distribution of the system.

## Iterative algorithm:

**Step 1**: Initialize $\pi(i,j,x,y,k)^{old} = \frac{1}{|s|}$ for all $(i,x,k) \in S$, where $|S|$ is the total number of feasible states.

**Step 2**: Substitute $\pi(i,j,x,y,k)^{old}$ into the balance equations from Case 1 to Case 16 to find $\pi(i,j,x,y,k)^{new}$, $\forall i,x,k$.

**Step 3**: Normalize $\pi(i,x,k)^{new}$, $\forall i,x,k$.

**Step 4**: If $\sqrt{\sum\sum\sum_{(i,x,k)\in S}|\pi(i,j,x,y,k)^{old} - \pi(i,j,x,y,k)^{new}|^2} < \varepsilon$, then stop the iteration. Otherwise, set $\pi(i,j,x,y,k)^{old} = \pi(i,j,x,y,k)^{new}$, $\forall i,x,k$, and return to **Step 2**.

In our analysis, the convergence threshold $\varepsilon$ is set to $10^{-8}$, and the algorithm typically converges after about 80 iterations.

### 3.4.3. Performance Measure

After obtaining the steady-state probabilities $\pi_{i,j,x,y,k}$ through the iterative algorithm, we proceed to compute several performance metrics to evaluate the effectiveness of the system.

First of all, the average number of high-priority and low-priority customers in the whole system, denoted by $L_H$ and $L_L$, respectively, is given by:

$$L_H = \sum_{k=0}^{1}\sum_{j=0}^{N}\sum_{i=0}^{N-j} i\pi_{i,j,0,0,k} + \sum_{k=0}^{1}\sum_{x=1}^{b}\sum_{j=0}^{N-b}\sum_{i=0}^{N-b-j} (i+x)\pi_{i,j,x,0,k} + \sum_{k=0}^{1}\sum_{y=1}^{b}\sum_{j=0}^{N-b}\sum_{i=0}^{N-b-j} i\pi_{i,j,0,y,k} \tag{3-63}$$

$$L_L = \sum_{k=0}^{1}\sum_{j=0}^{N}\sum_{i=0}^{N-j} j\pi_{i,j,0,0,k} + \sum_{k=0}^{1}\sum_{x=1}^{b}\sum_{j=0}^{N-b}\sum_{i=0}^{N-b-j} j\pi_{i,j,x,0,k} + \sum_{k=0}^{1}\sum_{y=1}^{b}\sum_{j=0}^{N-b}\sum_{i=0}^{N-b-j} (j+y)\pi_{i,j,0,y,k} \quad (3\text{-}64)$$

The average number of customers in the whole system, denoted by $L$, is given by:

$$L = L_H + L_L \quad (3\text{-}65)$$

Second, the average number of high-priority and low-priority customers in customer queue, denoted by $L_{c_H}$ and $L_{c_L}$, respectively, is given by:

$$L_{c_H} = \sum_{k=0}^{1}\sum_{j=0}^{N}\sum_{i=0}^{N-j} i\pi_{i,j,0,0,k} + \sum_{k=0}^{1}\sum_{x=1}^{b}\sum_{j=0}^{N-b}\sum_{i=0}^{N-b-j} i\pi_{i,j,x,0,k} + \sum_{k=0}^{1}\sum_{y=1}^{b}\sum_{j=0}^{N-b}\sum_{i=0}^{N-b-j} i\pi_{i,j,0,y,k} \quad (3\text{-}66)$$

$$L_{c_L} = \sum_{k=0}^{1}\sum_{j=0}^{N}\sum_{i=0}^{N-j} j\pi_{i,j,0,0,k} + \sum_{k=0}^{1}\sum_{x=1}^{b}\sum_{j=0}^{N-b}\sum_{i=0}^{N-b-j} j\pi_{i,j,x,0,k} + \sum_{k=0}^{1}\sum_{y=1}^{b}\sum_{j=0}^{N-b}\sum_{i=0}^{N-b-j} j\pi_{i,j,0,y,k} \quad (3\text{-}67)$$

The average number of customers in customer queue, denoted by $L_c$, is given by:

$$L_c = L_{c_H} + L_{c_L} \quad (3\text{-}68)$$

Third, the average number of high-priority and low-priority customers in block queue, denoted by $L_{b_H}$ and $L_{b_L}$, respectively, is given by:

$$L_{b_H} = \sum_{k=0}^{1}\sum_{x=1}^{b}\sum_{j=0}^{N-b}\sum_{i=0}^{N-b-j} x\pi_{i,j,x,0,k} \quad (3\text{-}69)$$

$$L_{b_L} = \sum_{k=0}^{1}\sum_{y=1}^{b}\sum_{j=0}^{N-b}\sum_{i=0}^{N-b-j} y\pi_{i,j,0,y,k} \quad (3\text{-}70)$$

The average number of customers in block queue, denoted by $L_b$, is given by:

$$L_b = L_{b_H} + L_{b_L} \tag{3-71}$$

Fourth, the blocking probability of high-priority and low-priority customers in the system, denoted by $P_{b_H}$ and $P_{b_L}$, respectively, is given by:

$$\begin{aligned} P_{b_H} = &\sum_{k=0}^{1}\sum_{j=0}^{N-b}\pi_{N-j,j,0,0,k} + \sum_{k=0}^{1}\sum_{x=1}^{b}\sum_{j=0}^{N-b}\pi_{N-b-j,j,x,0,k} \\ &+ \sum_{k=0}^{1}\sum_{y=1}^{b}\sum_{j=0}^{N-b}\pi_{N-b-j,j,0,y,k} \end{aligned} \tag{3-72}$$

$$\begin{aligned} P_{b_L} = &\sum_{k=0}^{1}\sum_{i=0}^{b}\pi_{i,N-b,0,0,k} + \sum_{k=0}^{1}\sum_{j=0}^{N-b-1}\pi_{N-j,j,0,0,k} \\ &+ \sum_{k=0}^{1}\sum_{x=1}^{b}\sum_{j=0}^{N-b}\pi_{N-b-j,j,x,0,k} + \sum_{k=0}^{1}\sum_{y=1}^{b}\sum_{j=0}^{N-b}\pi_{N-b-j,j,0,y,k} \end{aligned} \tag{3-73}$$

The blocking probability of the system, denoted by $P_b$, is given by:

$$P_b = \frac{\lambda_H P_{b_H} + \lambda_L P_{b_L}}{\lambda_H + \lambda_L} \tag{3-74}$$

Fifth, the impatient probability of high-priority and low-priority customers in the system, denoted by $P_{im_H}$ and $P_{im_L}$, respectively, is given by:

$$P_{im_H} = \frac{\gamma_H L_{c_H}}{\lambda_H(1 - P_{b_H})} \tag{3-75}$$

$$P_{im_L} = \frac{\gamma_L L_{c_L}}{\lambda_L(1 - P_{b_L})} \tag{3-76}$$

The impatient probability of the system, denoted by $P_{im}$, is given by:

$$P_{im} = \frac{\gamma_H L_{c_H} + \gamma_L L_{c_L}}{\lambda_H\left(1 - P_{b_H}\right) + \lambda_L(1 - P_{b_L})} \tag{3-77}$$

Sixth, the throughput of high-priority and low-priority customers in the system, denoted by $T_{h_H}$ and $T_{h_L}$, respectively, is given by:

$$T_{h_H} = \sum_{x=1}^{b} \sum_{i=1}^{N-b} \sum_{j=0}^{N-b} \mu_{2_H} x \pi_{i,j,x,0,1} \tag{3-78}$$

$$T_{h_L} = \sum_{y=1}^{b} \sum_{i=1}^{N-b} \sum_{j=0}^{N-b} \mu_{2_L} y \pi_{i,j,x,0,1} \tag{3-79}$$

The throughput of the system, denoted by $T_h$, is given by:

$$T_h = T_{h_H} + T_{h_L} \tag{3-80}$$

Seventh, the average waiting time of the high-priority and low-priority customers in the customer queue, denoted by $W_{c_H}$ and $W_{c_L}$, respectively, is given by:

$$W_{c_H} = \frac{L_{c_H}}{\lambda_H (1 - P_{b_H})} \tag{3-81}$$

$$W_{c_L} = \frac{L_{c_L}}{\lambda_L (1 - P_{b_L})} \tag{3-82}$$

The average waiting time in the customer queue, denoted by $W_c$, is given by:

$$W_c = \frac{L_c}{(\lambda_H + \lambda_L)(1 - P_b)} \tag{3-83}$$

Eighth, the average waiting time of the high-priority and low-priority customers in the block queue, denoted by $W_{b_H}$ and $W_{b_L}$, respectively, is given by:

$$W_{b_H} = \frac{L_{b_H}}{\lambda_H (1 - (P_{b_H} + (1 - P_{b_H}) P_{im_H})} \tag{3-84}$$

$$W_{b_L} = \frac{L_{b_L}}{\lambda_L (1 - P_{b_L} + (1 - P_{b_L}) P_{im_L})} \tag{3-85}$$

The average waiting time in the block queue, denoted by $W_b$, is given by:

$$W_b = \frac{L_b}{\lambda_H (1 - (P_{b_H} + (1 - P_{b_H}) P_{im_H}) + \lambda_L (1 - P_{b_L} + (1 - P_{b_L}) P_{im_L})} \tag{3-86}$$

Ninth, the average waiting time of the high-priority and low-priority customers in the system, denoted by $W_H$ and $W_L$, respectively, is given by:

$$W_H = W_{c_H} + W_{b_H} \tag{3-87}$$

$$W_L = W_{c_L} + W_{b_L} \tag{3-88}$$

The average waiting time in the block queue, denoted by $W$ is given by:

$$W = W_H + W_L \tag{3-89}$$

Finally, the average number of high-priority and low-priority blocks participating in the consensus process, denoted by $B_{n_H}$ and $B_{n_L}$, is given by:

$$B_{n_H} = \sum_{k=0}^{1} \sum_{x=0}^{b} \sum_{j=1}^{N-b} \sum_{i=0}^{N-j} \pi_{i,j,x,0,k} \tag{3-90}$$

$$B_{n_L} = \sum_{k=0}^{1} \sum_{y=0}^{b} \sum_{j=1}^{N-b} \sum_{i=0}^{N-j} \pi_{i,j,0,y,k} \tag{3-91}$$

The average number of blocks participating in the consensus process per unit of time, denoted by $B_n$, is given by:

$$B_n = B_{n_H} + B_{n_L} \tag{3-92}$$

# 4. Simulation Model

In this chapter, we present a detailed explanation of four simulation scenarios, each corresponding to a different configuration of blockchain queueing behavior. These scenarios are designed to reflect the structural and behavioral differences introduced by customer priority and impatience. All simulation models incorporate both First-Come-First-Served (FCFS) and non-preemptive priority disciplines, as appropriate to each case.

The first simulation model represents a single-class customer system without impatience. In this case, customers arrive and are served strictly in arrival order, and no abandonment occurs even if the waiting time is long. The second simulation model introduces two customer classes, high-priority and low-priority, handled with non-preemptive scheduling but without impatience. High-priority customers are always placed ahead of low-priority customers in the queue, but service-in-progress of any customer cannot be interrupted.

The third simulation model considers a single-class system with impatience, where customers may abandon the queue if they wait too long. This adds a stochastic abandonment dynamic based on patience thresholds. The final simulation model incorporates both customer priority and impatience. High-priority and low-priority customers are managed with non-preemptive priority, and both classes have their own impatience rates. This complex setting allows us to examine how prioritization and abandonment interact in a congested blockchain environment.

In all cases, the simulation captures system dynamics under partial batch service, and models ON/OFF channel behavior, where the service is suspended during OFF periods. These scenarios are simulated independently to compare their performance metrics, including throughput, queue lengths, waiting time, blocking probability, and, where applicable, abandonment probability.

## 4.1. Scenario 1: Single-Class Customer without Impatience

In this simulation model, we consider a blockchain system that handles a single class of customers, where customers arrive according to a Poisson process and are served under the First-Come-First-Served (FCFS) discipline.

The system consists of two queues: the customer queue, where customers wait for block generation, and the block queue, where customers participate in the consensus

process after being grouped into a block. Block generation follows a partial batch service policy, allowing 1 to $b$ customers to form a block. Once a block is formed, it is transferred to the block queue. Upon completion of the consensus process, all customers in the block exit the system.

During the OFF state, caused by interruptions such as attacks or connectivity issues, both block generation and consensus processes are suspended, although new customers may still arrive and be admitted. During the ON state, all services resume normally. To preserve system integrity, a constraint is imposed on the maximum number of customers allowed in the customer queue: when the block queue is empty, up to $N$ customers may wait; otherwise, the limit is reduced to $N - b$.

Since customer impatience is not considered in this model, all customers remain in the queue until they are served. This makes the first scenario a baseline case for performance comparison, focusing on metrics such as throughput, average queue length, and system utilization under a stable environment with uninterrupted customer service.

### 4.1.1. Main program

The main program executes a series of steps to simulate the blockchain queuing system, illustrated in Figure 4-1. At the beginning of each simulation run, all relevant variables are initialized. This includes resetting statistical parameters, setting the next block generation time and next departure time to infinity, marking the system status as ON, initializing the block generation status as idle, and setting the customer queue limit to $N$.

Next, the system parameters are configured. These include the maximum customer queue capacity ($N$), the maximum number of customers per block ($b$), the arrival rate ($\lambda$), the block generation rate ($\mu_1$), the consensus rate ($\mu_2$), and the ON/OFF switching rates ($\alpha$ and $\beta$) for the system channel.

The program generates the next arrival time and channel switch time using exponential random variables based on the corresponding system parameters. During the simulation, it compares the scheduled times of four events and selects the earliest event to execute its corresponding subprogram.

Finally, a while loop is used to repeat the simulation until a predefined number of customer arrivals has been reached. Once this condition is met, the simulation terminates and the performance statistics are output.

### 4.1.2. Arrival Subprogram

Figure 4.2 illustrated the flow chart of the arrival subprogram, simulates the arrival of a new customer to the system. Upon invocation, the total number of arrivals is incremented, and the simulation time is updated to the scheduled arrival time. The time for the next arrival is then scheduled using an exponential interarrival time generated with the arrival rate $\lambda$. Then, the area calculation function is invoked to update all time-averaged statistics based on the elapsed time since the last event.

Next, the system checks whether the customer queue has reached its capacity limit.

- If the queue is full, the arriving customer is rejected, and the number of rejections is incremented.
- If the queue is not full, the arriving customer is admitted. In this case, both the number of customers in the system and in the queue are incremented, and the customer's arrival time is recorded in the queue log.

Finally, the system determines whether to initiate block generation:

- If the channel status is in ON state, and block generator is idle, and this customer is the only one in the queue, a new block generation event is scheduled based on an exponential random variable with rate $\mu_c$.
- If the block generator is busy or the channel is OFF, the next block generation time is set to infinity to suspend the process.

### 4.1.3. Block Generation Subprogram

Figure 4-3 illustrates the flow chart of the block generation subprogram, which simulates the initiation of a block generation process. When this event is triggered, the simulation time is updated to the scheduled block generation time. Then, the area calculation function is invoked to update all time-averaged statistics based on the elapsed time since the last event.

Next, the block generator status is set to busy, indicating that a block is currently being generated. To ensure sufficient space for the upcoming consensus process, the capacity limit of the customer queue is reduced from $N$ to $N - b$, where $b$ is the maximum number of customers allowed in a block.

The system then determines how many customers should be transferred from the queue into the block:

- If more than $b$ customers are waiting in the queue, exactly $b$ are selected.
- Otherwise, all remaining customers in the queue are moved into the block.

The number of customers transferred into the block is recorded, and the queue size is adjusted accordingly. A block departure event is then scheduled based on an exponential random variable with rate $\mu_2$. After this, the next block generation time is set to infinity to prevent immediate retriggering.

For each customer that enters the block, their corresponding arrival time is logged into the block log. These timestamps are subsequently used to compute the cumulative queueing time. This calculation is performed using the total waiting time function, which sums the time differences between the current simulation time and each customer's original queue entry time.

Finally, the corresponding entries in the queue log are removed to reflect that these customers have exited the queue and are now participating in the consensus process.

### 4.1.4. Block Departure Subprogram

Figure 4-4 illustrates the flow chart of the departure subprogram, which simulates the completion of a block consensus process. When this event is triggered, the simulation time is updated to the scheduled block departure time. Then, the area calculation function is invoked to update all time-averaged statistics based on the elapsed time since the last event.

At this point, the block generation status is reset to idle, and the customer queue capacity limit is restored to its original value $N$, allowing the queue to accept new customers at full capacity. The block departure event is considered completed and is therefore cleared.

The program then calculates the total time that the current block of customers spent in the consensus stage. This is achieved using the block time accumulation function, which computes the total time difference between the current simulation time and each customer's recorded entry into the block.

After consensus completion, the number of customers currently in the system is decreased by the number of customers in the departing block, and the total number of customers served is incremented accordingly. The block is now empty, and all associated entries in the block log are removed.

Finally, if there are still customers waiting in the queue, a new block generation event is scheduled based on an exponential random variable with rate $\mu_1$.

### 4.1.5. Switch Subprogram

Figure 4-5 illustrates the flow chart of the switch subprogram, which simulates the transition of the system between ON and OFF states. When this event is triggered, the simulation time is updated to the scheduled switch time. Then, the area calculation function is invoked to update all time-averaged statistics based on the elapsed time since the last event.

The system channel status is then toggled as follows:

- **If the system transitions from ON to OFF:**
  - The channel status is set to OFF.
  - The next switch event is scheduled based on an exponential random variable with rate $\beta$ (representing the OFF duration).
  - All ongoing service operations are suspended by setting both the block generation and block departure event times to infinity.
- **If the system transitions from OFF to ON:**
  - The channel status is set to ON.
  - The next switch event is scheduled based on an exponential random variable with rate $\alpha$ (representing the ON duration).
  - If there is at least one customer in the queue and the block generator is currently idle:
    - A block generation event is scheduled based on an exponential random variable with rate $\mu_1$.
  - If a block in consensus phase is suspended:
    - A block departure event is scheduled using an exponential random variable with rate $\mu_2$.

Through this subprogram, the simulation captures the stochastic availability of the system by alternating between operational and suspended phases, reflecting real-world unreliability such as downtime or external disruptions. During the ON period, block generation and consensus operations proceed as normal. During the OFF period, these processes are temporarily halted while new customer arrivals may still occur.

### 4.1.6. Flowchart Diagram

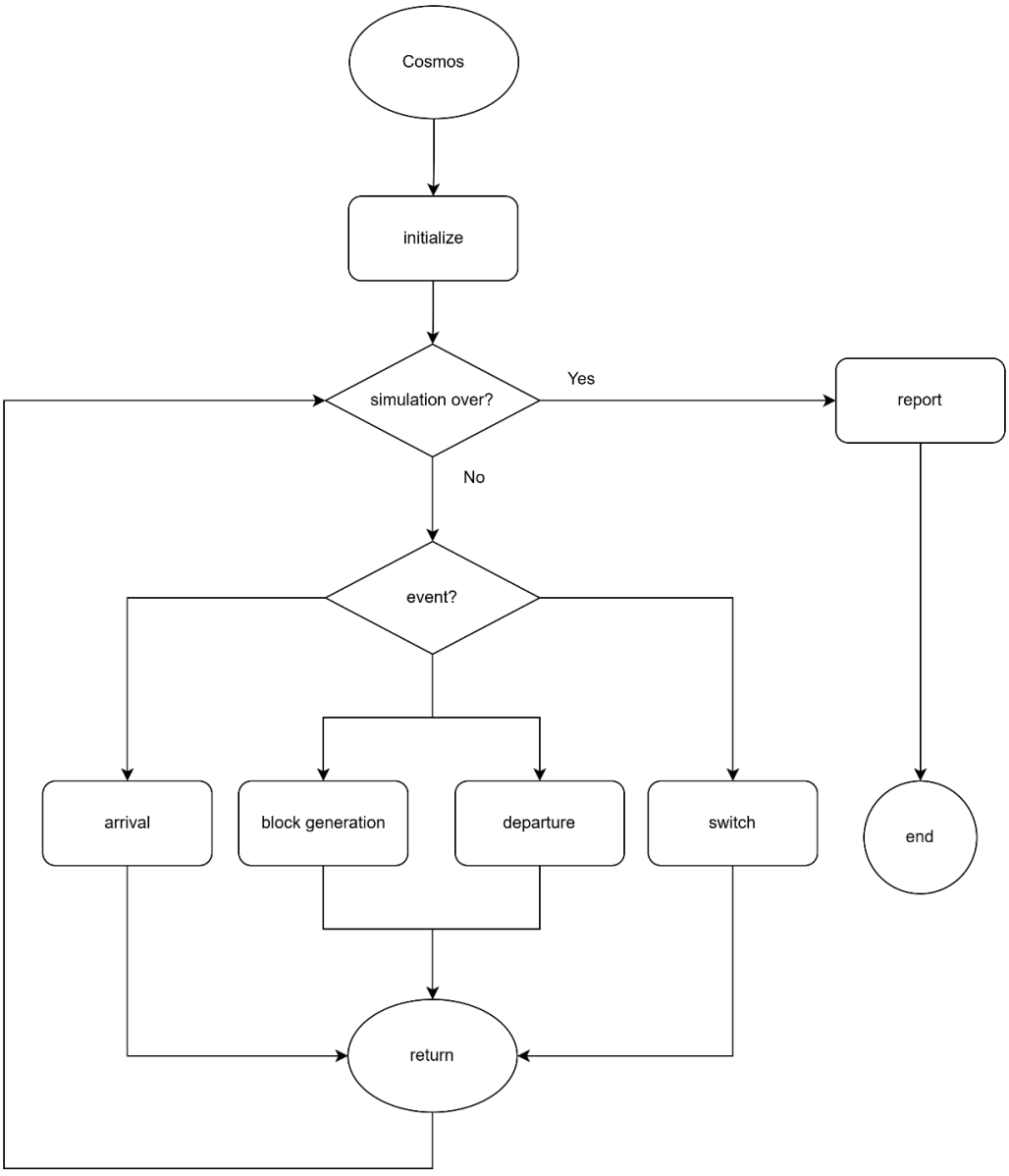


Figure 4-1: Flow chart of main program

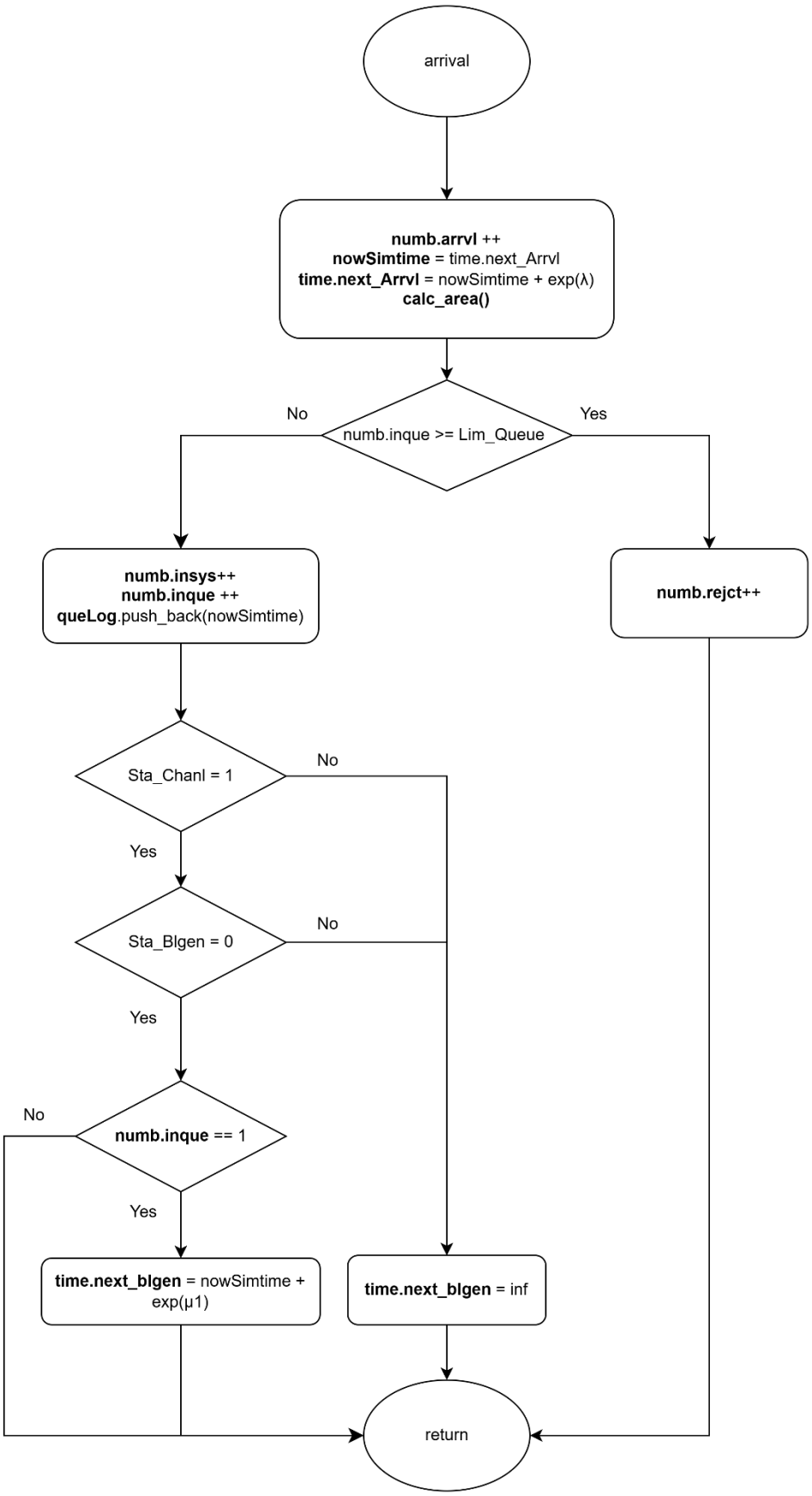


Figure 4-2: Flow chart of arrival subprogram

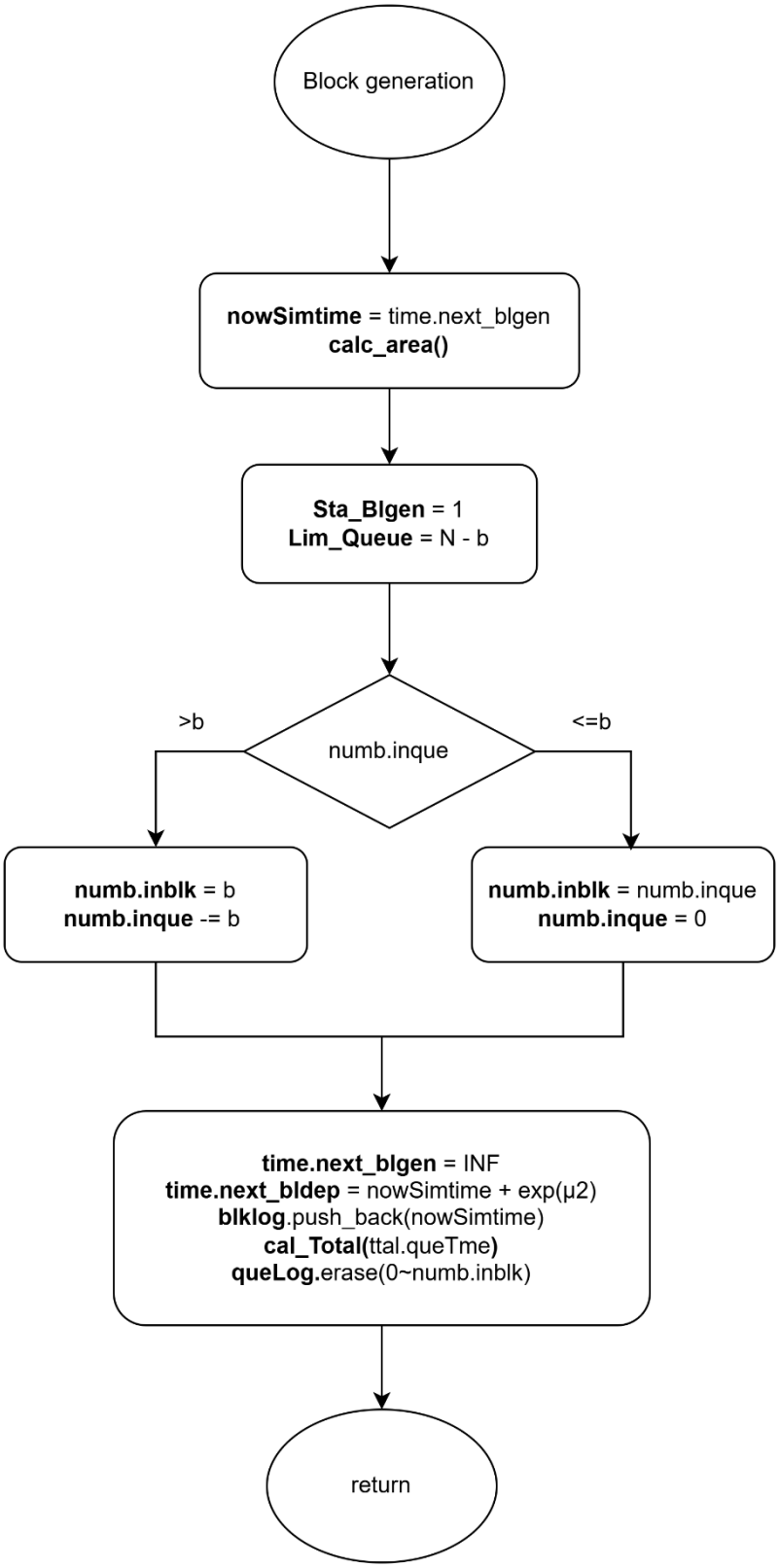


Figure 4-3: Flow chart of block generation subprogram

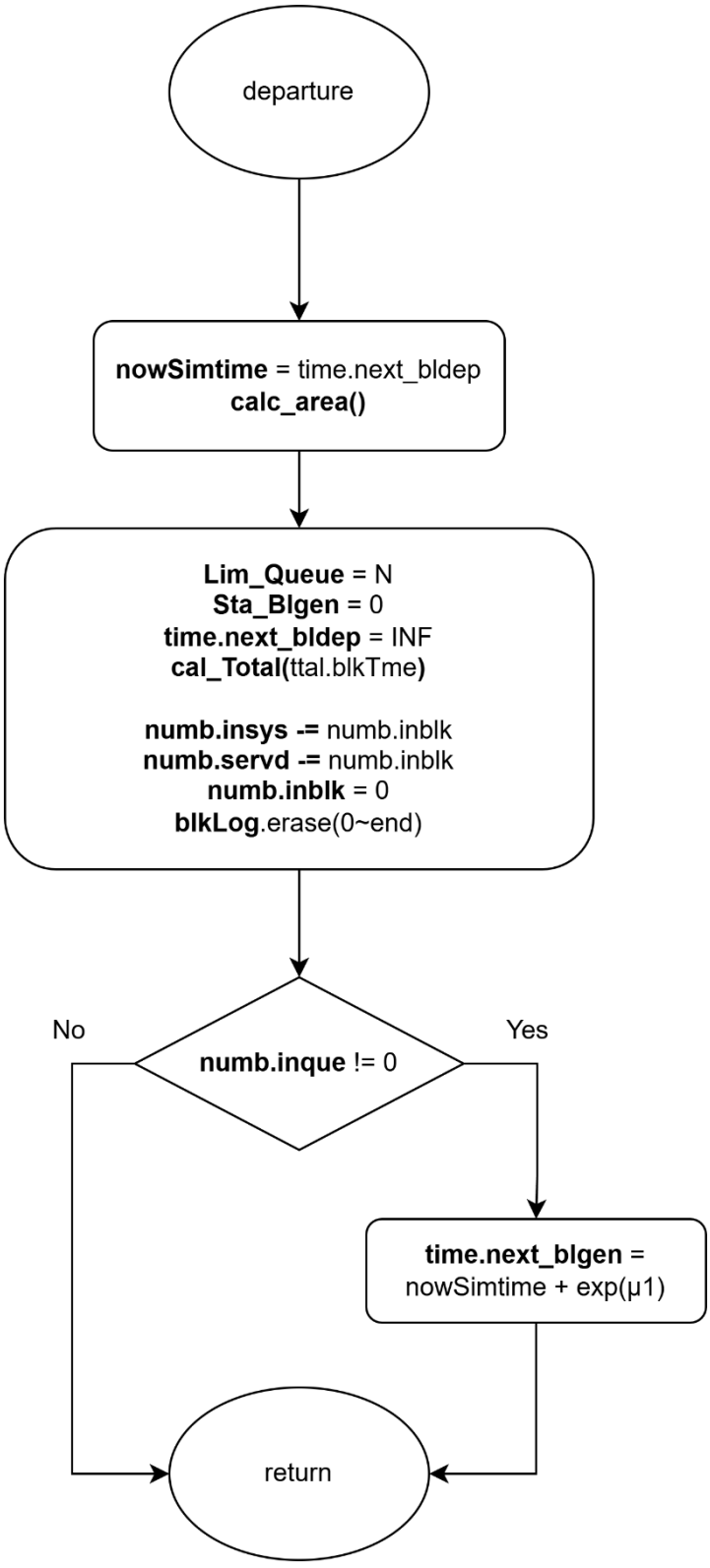


Figure 4-4: Flow chart of block departure subprogram

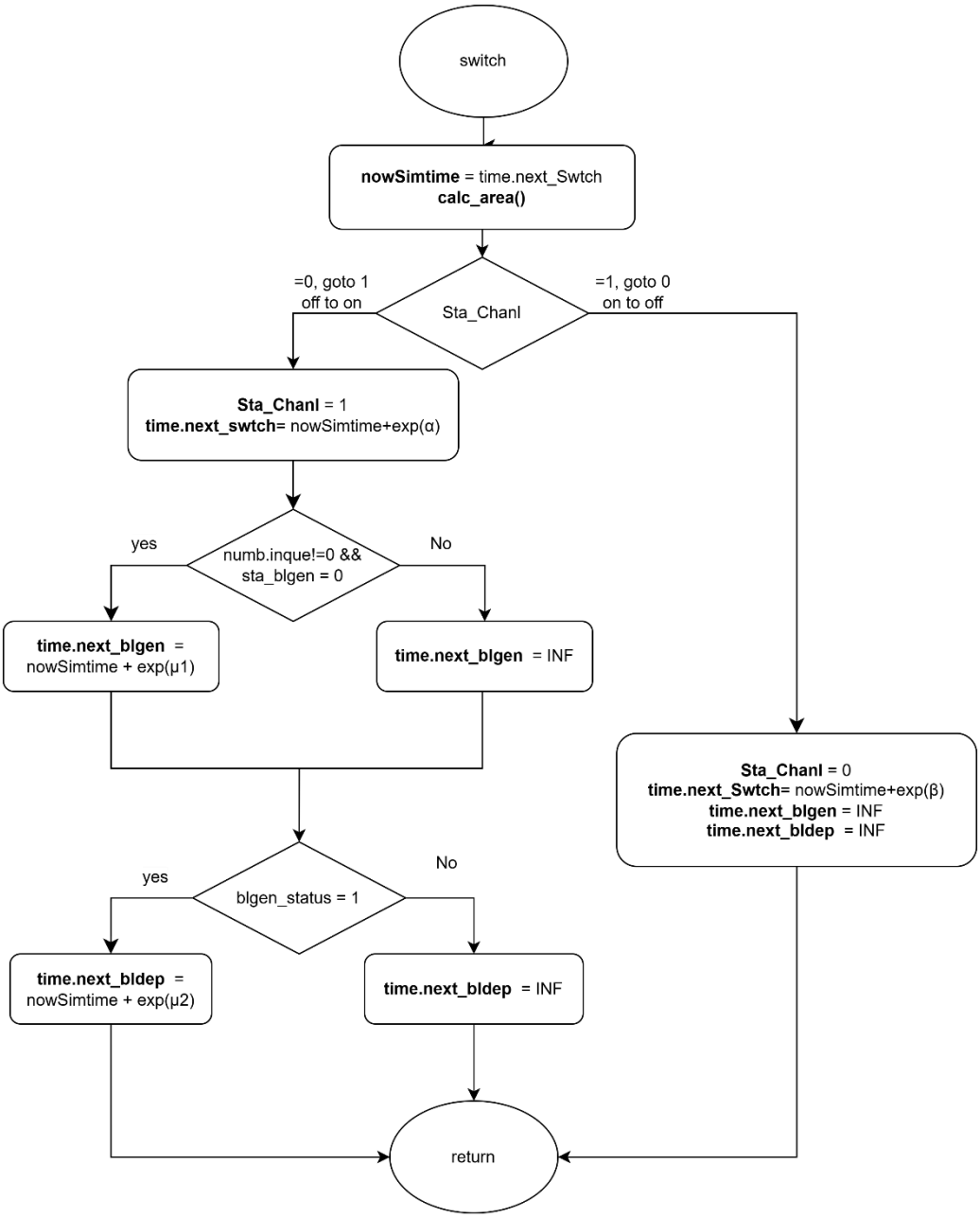


Figure 4-5: Flow chart of switch subprogram

### 4.1.7. Performance Index

To evaluate the system's performance, we compute several performance indices based on the simulated results obtained from the simulation.

First of all, the average number of customers in the whole system, denoted by $L$, is given by:

$$L = \frac{Area\_total\_cust\_in\_system}{sim\_time} \quad (4\text{-}1)$$

Second, the average number of customers in customer queue, denoted by $L_c$, is given by:

$$L_c = \frac{Area\_total\_cust\_in\_queue}{sim\_time} \quad (4\text{-}2)$$

Third, the average number of customers in block queue, denoted by $L_b$, is given by:

$$L_b = \frac{Area\_total\_cust\_in\_block}{sim\_time} \quad (4\text{-}3)$$

Fourth, the blocking probability of the system, denoted by $P_b$, is given by:

$$P_b = \frac{total\_rejected\_cust}{total\_num\_of\_arrival} \quad (4\text{-}4)$$

Fifth, the throughput of the system, denoted by $T_h$, is given by:

$$T_h = \frac{total\_served\_cust}{sim\_time} \quad (4\text{-}5)$$

Sixth, the average waiting time in the system, denoted by $W$, is given by:

$$W = \frac{total\_queue\_time + total\_block\_time}{total\_served\_cust} \quad (4\text{-}6)$$

Seventh, the average waiting time in the customer queue, denoted by $W_c$, is given by:

$$W_c = \frac{total\_queue\_time}{total\_served\_cust} \quad (4\text{-}7)$$

Eighth, the average waiting time in the block queue, denoted by $W_b$, is given by:

$$W_b = \frac{total\_block\_time}{total\_served\_cust} \quad (4\text{-}8)$$

Finally, the average number of blocks participating in the consensus process per

unit of time, denoted by $B_n$, is given below.

$$B_n = \frac{Area\_total\_num\_of\_block}{sim\_time} \tag{4-9}$$

## 4.2. Scenario 2: Two-Class Customer without Impatience

In this simulation model, we consider a blockchain system that handles two classes of customers: high-priority and low-priority customers. Customers of different priorities arrive according to an independent Poisson process, and customers of the same priority are served under the First-Come-First-Served (FCFS) discipline. Customers of different priorities are handled using a non-preemptive priority rule, where high-priority customers are always placed ahead of low-priority customers in the queue, but ongoing service for a low-priority block cannot be interrupted once initiated.

The system consists of two queues: the customer queue and the block queue. High-priority and low-priority customers both enter the customer queue upon arrival if possible. Block generation follows a partial batch service policy and operates on customers of only one priority class at a time. Each block can include at most $b$ customers. Once a block that is composed solely of high-priority or low-priority customer(s) is formed, it is transferred to the block queue. After completing the consensus process, all customers in the block exit the system.

The system alternates between ON and OFF periods to reflect external interruptions such as cyberattacks or connection failures. During the OFF state, both block generation and consensus process are suspended, though new customers may still arrive and be queued. During the ON state, all services resume as normal.

To preserve system integrity and fairness, different queue capacity constraints apply based on customer class and system state. For high-priority customers, the maximum number allowed in the customer queue is $N$ when the block queue is empty, and $N - b$ when it is occupied. For low-priority customers, the customer queue capacity is always limited to $N - b$, regardless of the block queue state.

Since this model does not include customer impatience, all arriving customers remain in the system until they are served. This scenario is designed to study the impact of non-preemptive priority scheduling on performance, focusing on metrics such as per-class throughput, queue length, and system utilization under stable yet priority-sensitive operation.

### 4.2.1. Main program

The main program executes a series of steps to simulate the blockchain queuing system with two classes of customers, as illustrated in Figure 4-6. At the beginning of each simulation run, all relevant variables are initialized. This includes resetting

statistical parameters, setting the next block generation time and next departure time to infinity, marking the system status as ON, initializing the block generation status as idle, and setting the customer queue limit to $N$.

Next, the system parameters are configured. These include the maximum customer queue capacity ($N$), the maximum number of customers per block ($b$), the arrival rates for high-priority and low-priority customers ($\lambda_H$ and $\lambda_L$), the block generation rates ($\mu_{1_H}$ and $\mu_{1_L}$), the consensus rates ($\mu_{2_H}$ and $\mu_{2_L}$), and the ON/OFF switching rates ($\alpha$ and $\beta$) for the system channel.

The program then generates the next arrival times for both high-priority and low-priority customers, as well as the channel switch time, using exponential random variables based on the corresponding system parameters. During the simulation, it compares the scheduled times of five events and selects the earliest event to execute its corresponding subprogram.

Finally, a while loop is used to repeat the simulation until a predefined number of customer arrivals has been reached. Once this condition is met, the simulation terminates and the performance statistics are output.

### 4.2.2. High-Priority Arrival Subprogram

As illustrated in Figure 4-7, the high-priority arrival subprogram simulates the arrival of a high-priority customer to the system. When this event is triggered, the simulation time is updated to the current arrival time. The arrival counters are incremented to reflect both the total number of customers and the number of high-priority arrivals. The next arrival event for high-priority customers is then scheduled based on an exponential random variable with rate $\lambda_H$. Immediately after, the area calculation function is invoked to update all time-averaged statistics based on the elapsed time since the last event.

Next, the system checks whether the customer queue has reached its capacity limit.

- If the queue is full, the arriving customer is rejected. In this case, both the total number of rejections and the number of high-priority rejections are incremented.
- If the queue is not full, the arriving customer is admitted. The number of customers in the system and in the queue are both incremented, along with their corresponding high-priority counts. The arrival time of the customer is recorded in the high-priority queue log.

Then, the system updates the unified queue log by merging the high-priority and low-priority arrival records, ensuring that high-priority entries appear first. The system then sets the priority flag to indicate that high-priority customers are currently at the head of the queue.

Finally, the system determines whether to initiate block generation:

- If the channel status is in ON state, and block generator is idle, and this customer is the only one in the queue, a new block generation event is scheduled based on an exponential random variable with rate $\mu_{1_H}$.
- If the block generator is busy or the channel is OFF, the next block generation time is set to infinity to suspend the process.

### 4.2.3. Low-Priority Arrival Subprogram

As illustrated in Figure 4-8, the low-priority subprogram simulates the arrival of a low-priority customer to the system. When this event is triggered, the simulation time is updated to the scheduled arrival time. The arrival counters are incremented to reflect both the total number of customers and the number of low-priority arrivals.

The next arrival event for low-priority customers is then scheduled based on an exponential random value with the rate $\lambda_L$. Then, the area calculation function is invoked to update all time-averaged statistics based on the elapsed time since the last event.

Next, the system checks whether the customer queue has reached its capacity limit. The rejection condition depends on the system's block generation state:

- If the number of low-priority customers in the queue has reached $N - b$, or
- The total number of customers in the queue has reached the current queue limit,

then the arriving customer is rejected. In this case, both the total number of rejections and the number of low-priority rejections are incremented.

If neither of the above two rejection conditions is met, the arriving customer is admitted. The number of customers in the system and in the queue are both incremented, along with their corresponding low-priority counts. The arrival time is recorded in the low-priority queue log. To maintain unified tracking, the system then refreshes the combined queue log by merging both priority records, ensuring high-priority entries appear first.

If there are no high-priority customers currently in the queue, the system sets the priority flag to indicate that low-priority customer is now at the head of the queue.

Finally, the system determines whether to initiate block generation:

- If the channel status is in ON state, and block generator is idle, and this customer is the only one in the queue, a new block generation event is scheduled based on an exponential random variable with rate $\mu_{1_L}$.
- If more than one customer is in the queue, the block generation time remains unchanged.
- If the block generator is busy or the channel is OFF, the next block generation time is set to infinity to suspend the process.

### 4.2.4. Block Generation Subprogram

As illustrated in Figure 4-9, the block generation subprogram simulates the initiation of a block generation process. When this event is triggered, the simulation time is updated to the scheduled block generation time. Then, the area calculation function is invoked to update all time-averaged statistics based on the elapsed time since the last event. The next block generation time is set to infinity to prevent immediate retriggering, and the block generator status is marked as active.

To reserve sufficient space in the queue for block formation, the maximum queue size is reduced from $N$ to $N - b$.

The system then determines the class of customers from which the block will be constructed, based on the current queue head status:

- **If the high-priority customer is at the head of the queue**:
    - If the number of high-priority customers exceeds the block size $b$, exactly $b$ of them are moved into the block.
    - Otherwise, all high-priority customers are included in the block. If low-priority customers remain in the queue, the queue head status is updated accordingly; otherwise, it is cleared.
    - The block is marked as high-priority, and the consensus process is scheduled based on an exponential random variable with rate $\mu_{2_H}$.
    - The total waiting time of high-priority customers in the block is calculated using their individual arrival times stored in the high-priority queue log, which is then updated by removing the corresponding entries, thereby reflecting their transition into the consensus stage.

- **If low-priority customer is at the head of the queue**:
    - If there are more than $b$ low-priority customers, exactly $b$ are selected for the block.
    - Otherwise, all remaining low-priority customers are included, and the queue is emptied.
    - The block is marked as low-priority, and the block departure event is scheduled based on an exponential random variable with rate $\mu_{2_L}$.
    - The total waiting time of low-priority customers is calculated using their queue log, which is then updated accordingly.

After determining the block content, the system updates the overall waiting time in the queue for all customers, using the unified queue log. The combined queue log is then refreshed to reflect the current state of the customer queue.

Finally, the block log is updated with the entry time for each customer in the new block. This log is used for downstream statistics related to block-based consensus activity.

### 4.2.5. Block Departure Subprogram

As illustrated in Figure 4-10, the block departure subprogram simulates the completion of a block's consensus process. When this event is triggered, the simulation time is updated to the current departure time. The area calculation function is then invoked to update all time-averaged statistics based on the elapsed time since the last event. The next block departure time is set to infinity to suspend further departure scheduling until a new block is formed.

The block generator status is reset to idle, and the customer queue capacity limit is restored to its original value $N$, allowing the system to admit new arrivals without restriction.

The total waiting time of the block in the consensus stage is accumulated using the block log. The total number of customers currently in the system is decreased by the number of customers in the departing block, and the number of customers served is incremented accordingly. The block is then cleared.

The next processing depends on the priority class of the departing block:

- If it is a high-priority block, the corresponding class-specific consensus time is updated, and the number of high-priority customers in the system and served counters are adjusted.
- If it is a low-priority block, the low-priority statistics are updated in a similar manner.

After processing, the system clears the block log and resets the priority status of the block.

Finally, the system checks whether customers remain in the queue. If so, a new block generation event is scheduled based on the class of the customer at the head of the queue:

- If the high-priority customer is at the head of the queue, the next block generation is scheduled based on an exponential random variable with rate $\mu_{1_H}$.
- If the low-priority customer is at the head of the queue, the event is scheduled with rate $\mu_{1_L}$.

If the queue is empty, no block generation is scheduled, and the next block departure time is set to infinity.

### 4.2.6. Switch Subprogram

As illustrated in Figure 4-11, the switch subprogram simulates the transition of the system between ON and OFF operational states. When this event is triggered, the simulation time is updated to the current switch time. The area calculation function is then invoked to update all time-averaged statistics based on the elapsed time since the last event.

The system channel status is then toggled as follows:

- **If the system transitions from ON to OFF:**
  - The channel status is updated to OFF.
  - The next switch event is scheduled based on an exponential random variable with rate β (representing the OFF duration).
  - All pending block generation and block departure events are suspended by setting their scheduled times to infinity.
- **If the system transitions from OFF to ON:**
  - The channel status is updated to ON.

- The next switch event is scheduled based on an exponential random variable with rate α (representing the ON duration).
- If there is at least one customer in the queue and the block generator is idle:
  - A block generation event is scheduled based on an exponential random variable with rate $\mu_{1_H}$ or $\mu_{1_L}$, depending on whether high- or low-priority customer is at the head of the queue.
- If a block in consensus phase is suspended:
  - A block departure event is scheduled using the appropriate rate ($\mu_{2_H}$ or $\mu_{2_L}$) based on the class of the current block.

Through this subprogram, the system emulates the effects of environmental disruptions such as connectivity loss or cyberattacks by alternating between active and inactive service periods. During the ON state, queuing, block generation and consensus operations proceed. During the OFF state, only customer arrivals are allowed, while block generation and consensus operation are temporarily halted.

### 4.2.7. Flowchart Diagram

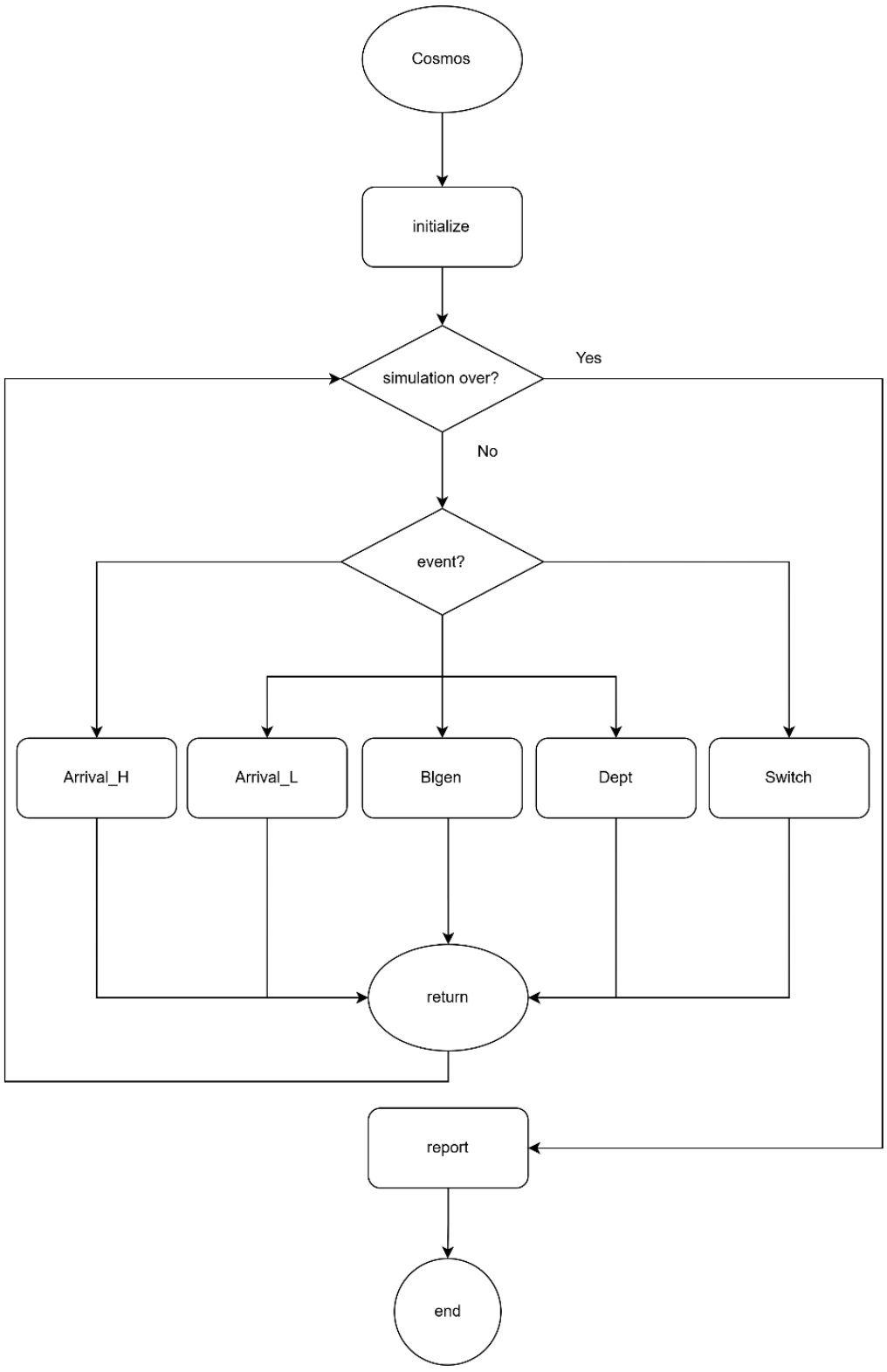


Figure 4-6: Flow chart of main program

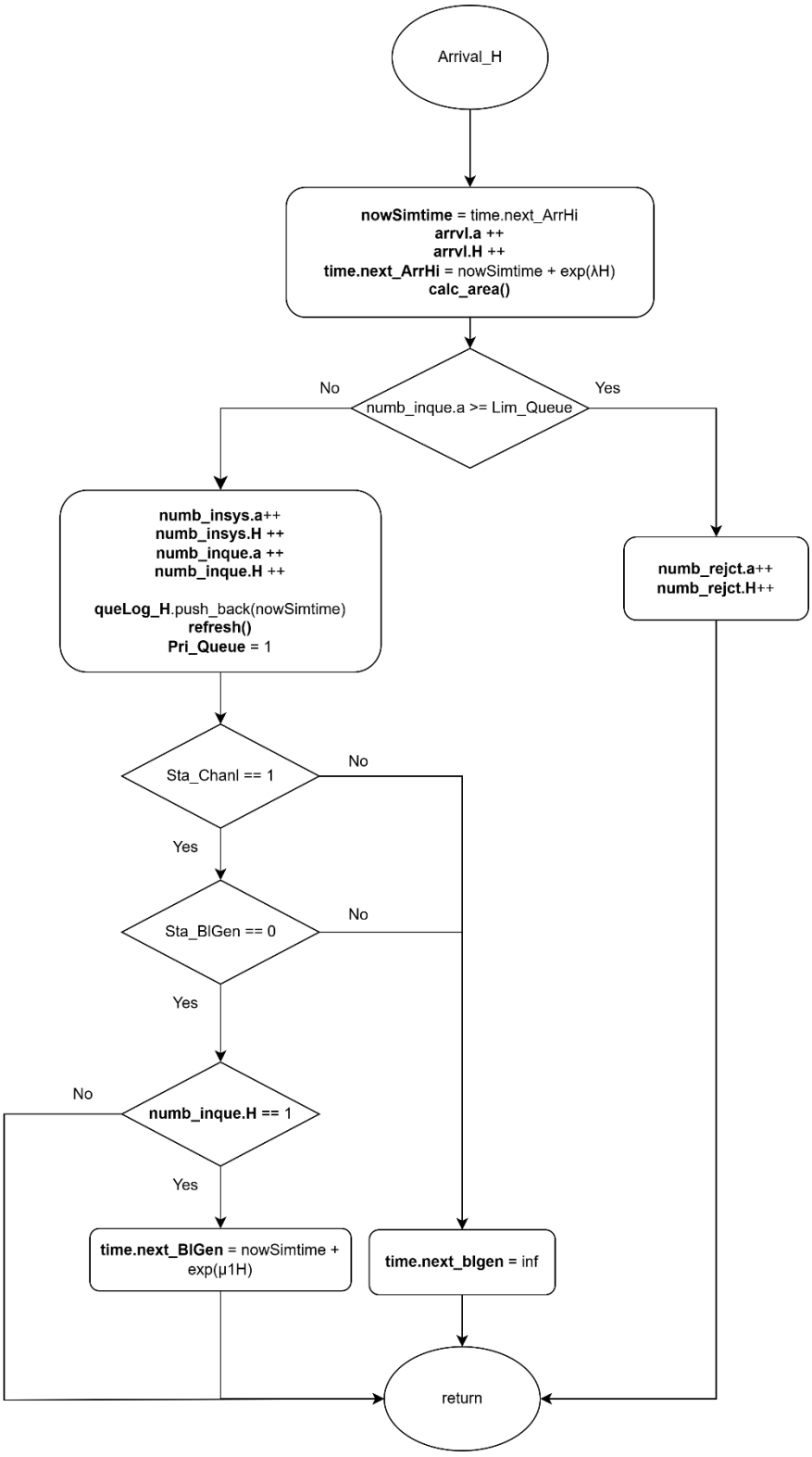


Figure 4-7: Flow chart of high-priority arrival subprogram

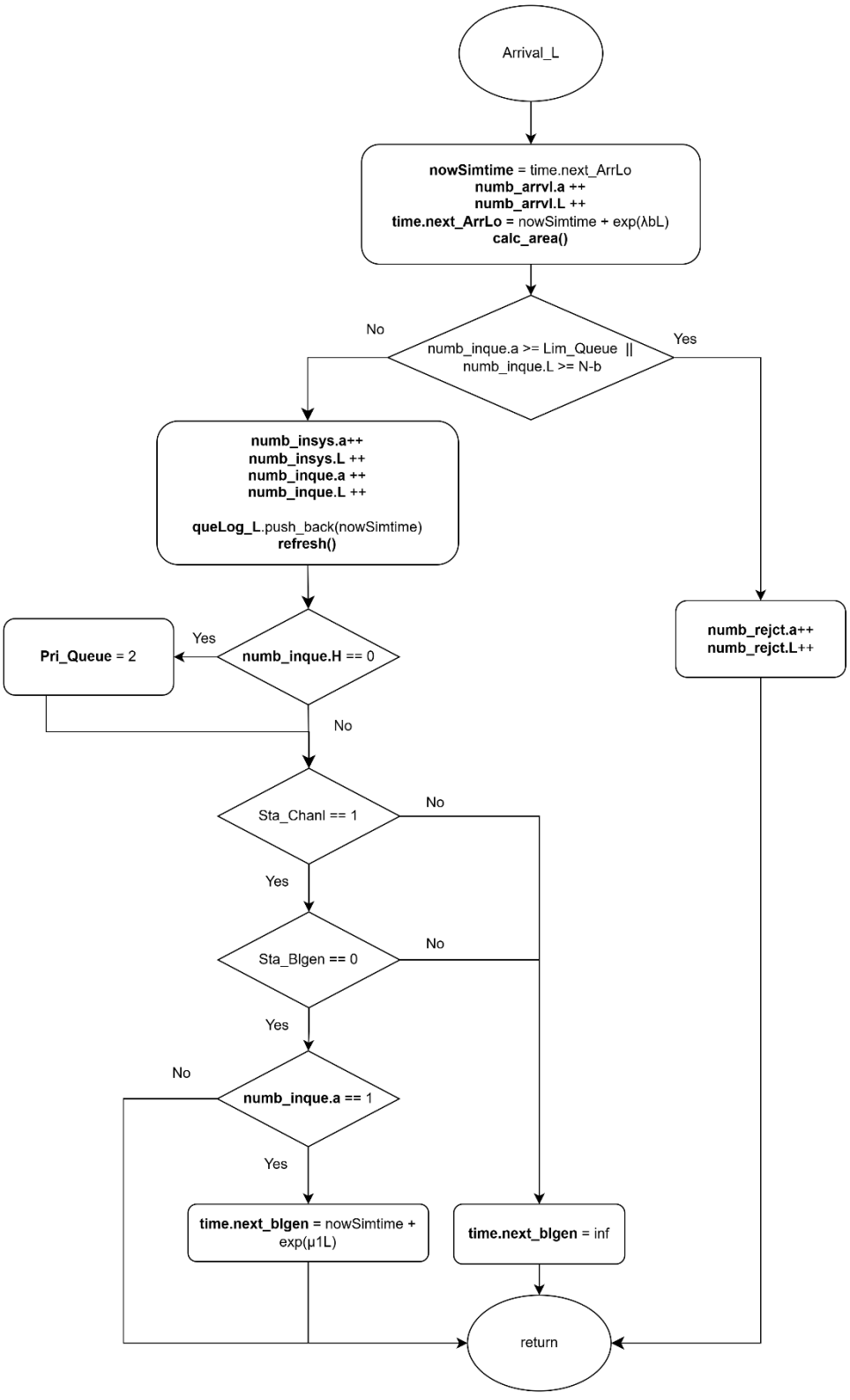


Figure 4-8: Flow chart of low-priority arrival subprogram

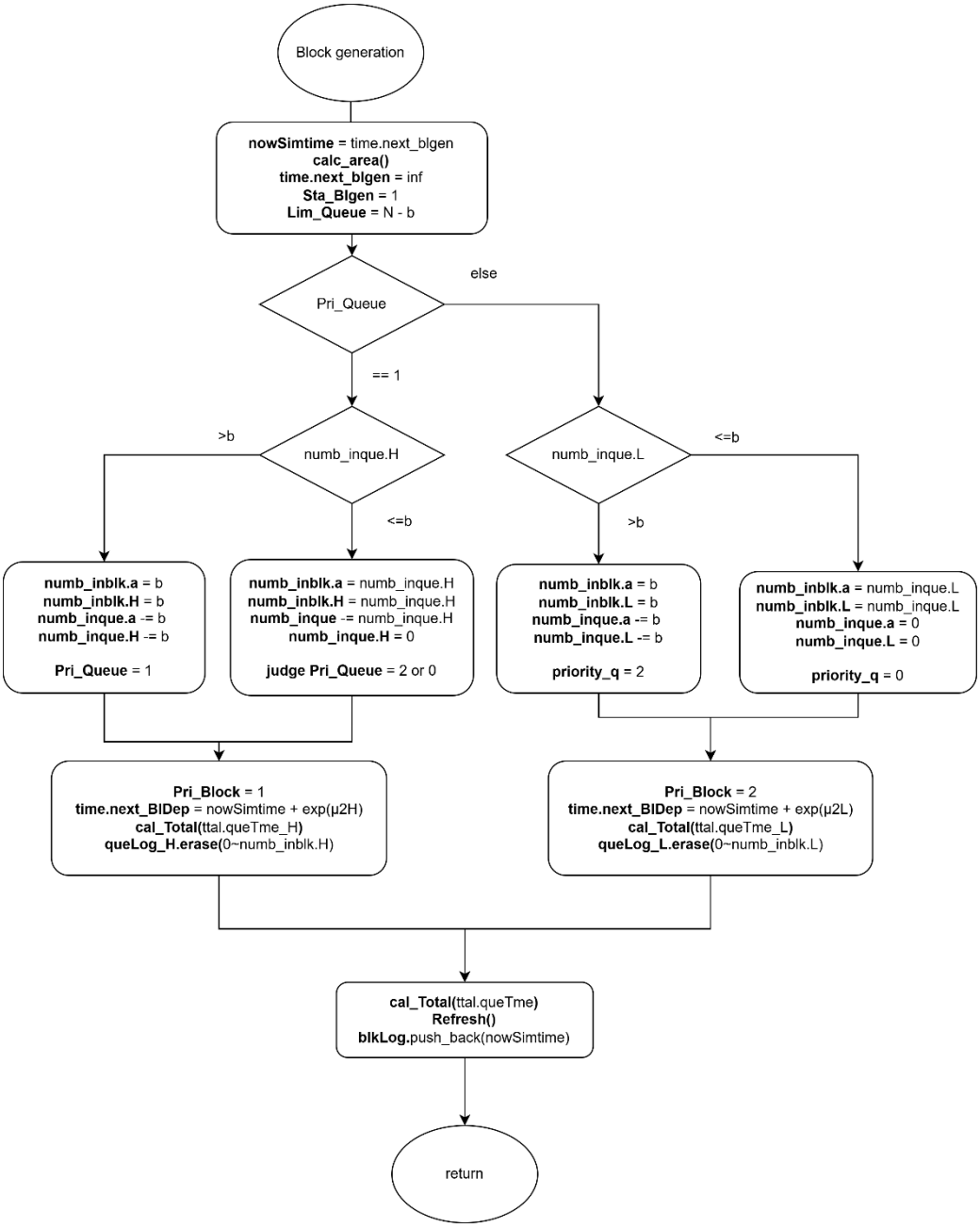


Figure 4-9: Flow chart of block generation subprogram

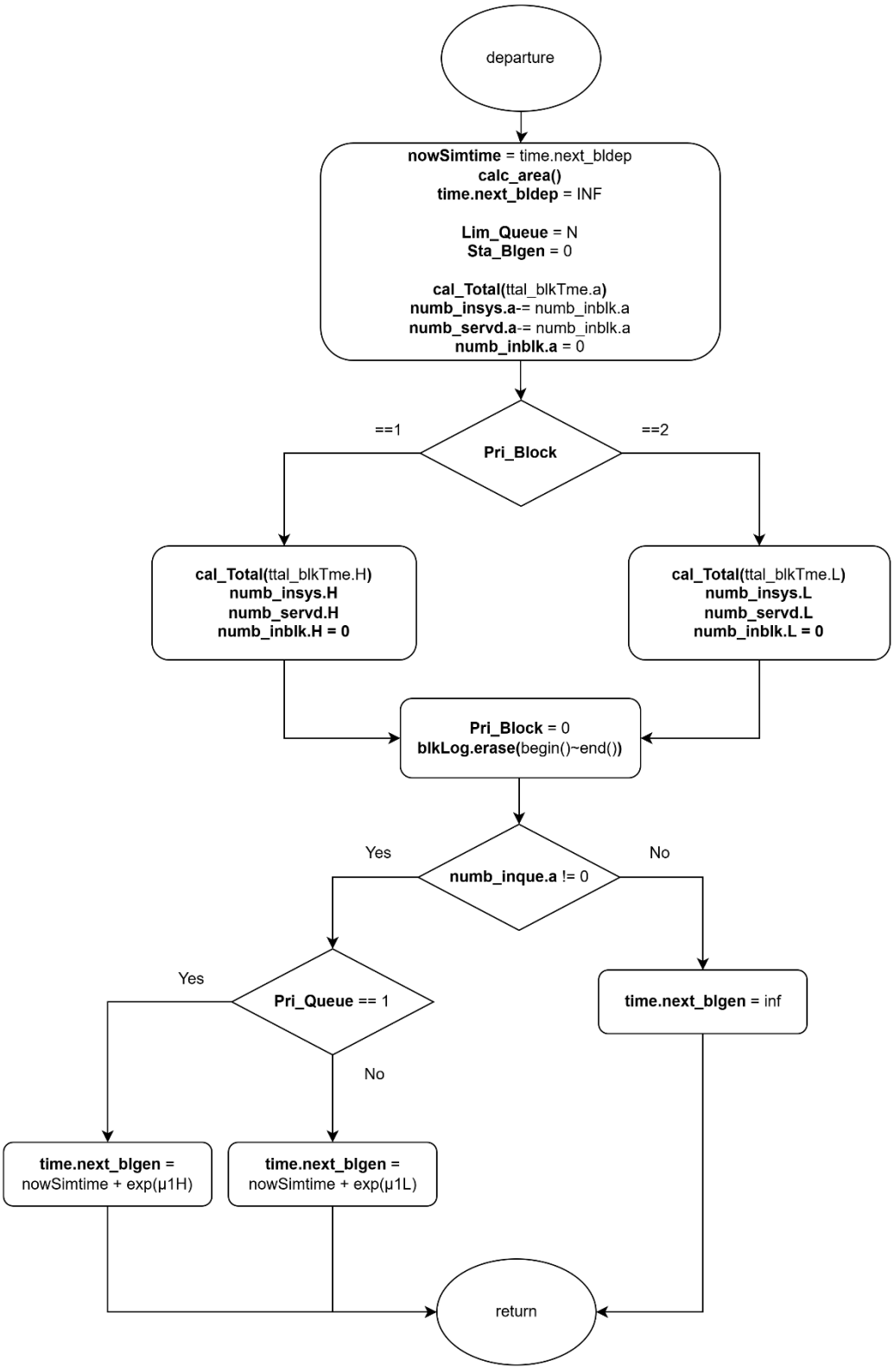


Figure 4-10: Flow chart of block departure subprogram

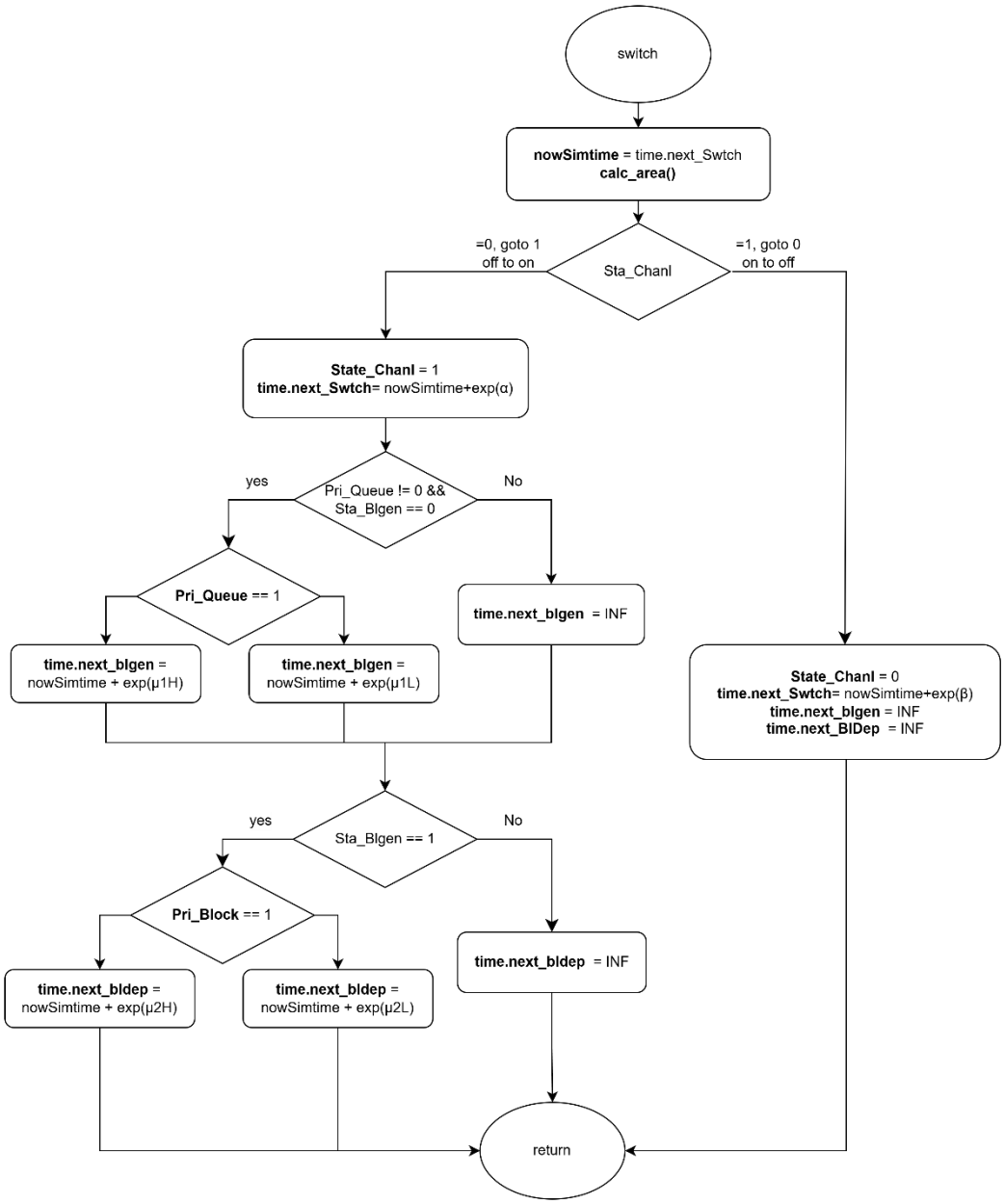


Figure 4-11: Flow chart of switch subprogram

## 4.2.8. Performance Index

First of all, the average number of high-priority and low-priority customers in the whole system, denoted by $L_H$ and $L_L$, respectively, is given by:

$$L_H = \frac{Area\_total\_cust\_in\_system\_H}{sim\_time} \tag{4-10}$$

$$L_L = \frac{Area\_total\_cust\_in\_system\_L}{sim\_time} \tag{4-11}$$

The average number of customers in the whole system, denoted by $L$, is given by:

$$L = \frac{Area\_total\_cust\_in\_system}{sim\_time} \tag{4-12}$$

Second, the average number of high-priority and low-priority customers in customer queue, denoted by $L_{c_H}$ and $L_{c_L}$, respectively, is given by:

$$L_{c_H} = \frac{Area\_total\_cust\_in\_queue\_H}{sim\_time} \tag{4-13}$$

$$L_{c_L} = \frac{Area\_total\_cust\_in\_queue\_L}{sim\_time} \tag{4-14}$$

The average number of customers in customer queue, denoted by $L_c$, is given by:

$$L_c = \frac{Area\_total\_cust\_in\_queue}{sim\_time} \tag{4-15}$$

Third, the average number of high-priority and low-priority customers in block queue, denoted by $L_{b_H}$ and $L_{b_L}$, respectively, is given by:

$$L_{b_H} = \frac{Area\_total\_cust\_in\_block\_H}{sim\_time} \tag{4-16}$$

$$L_{b_L} = \frac{Area\_total\_cust\_in\_block\_L}{sim\_time} \tag{4-17}$$

The average number of customers in block queue, denoted by $L_b$, is given by:

$$L_b = \frac{Area\_total\_cust\_in\_block}{sim\_time} \tag{4-18}$$

Fourth, the blocking probability of high-priority and low-priority customers in the system, denoted by $P_{b_H}$ and $P_{b_L}$, respectively, is given by:

$$P_{b_H} = \frac{total\_rejected\_cust\_H}{total\_num\_of\_arrival\_H} \tag{4-19}$$

$$P_{b_L} = \frac{total\_rejected\_cust\_L}{total\_num\_of\_arrival\_L} \tag{4-20}$$

The blocking probability of the system, denoted by $P_b$, is given by:

$$P_b = \frac{total\_rejected\_cust}{total\_num\_of\_arrival} \quad (4\text{-}21)$$

Fifth, the throughput of high-priority and low-priority customers in the system, denoted by $T_{h_H}$ and $T_{h_L}$, respectively, is given by:

$$T_{h_H} = \frac{total\_served\_cust\_H}{sim\_time} \quad (4\text{-}22)$$

$$T_{h_L} = \frac{total\_served\_cust\_L}{sim\_time} \quad (4\text{-}23)$$

The throughput of the system, denoted by $T_h$, is given by:

$$T_h = \frac{total\_served\_cust}{sim\_time} \quad (4\text{-}24)$$

Sixth, the average waiting time of the high-priority and low-priority customers in the system, denoted by $W_H$ and $W_L$, respectively, is given by:

$$W_H = \frac{total\_queue\_time\_H + total\_block\_time\_H}{total\_served\_cust\_H} \quad (4\text{-}25)$$

$$W_L = \frac{total\_queue\_time\_L + total\_block\_time\_L}{total\_served\_cust\_L} \quad (4\text{-}26)$$

The average waiting time in the system, denoted by $W$, is given by:

$$W = \frac{total\_queue\_time + total\_block\_time}{total\_served\_cust} \quad (4\text{-}27)$$

Seventh, the average waiting time of the high-priority and low-priority customers in the customer queue, denoted by $W_{c_H}$ and $W_{c_L}$, respectively, is given by:

$$W_{c_H} = \frac{total\_queue\_time\_H}{total\_served\_cust\_H} \quad (4\text{-}28)$$

$$W_{c_L} = \frac{total\_queue\_time\_L}{total\_served\_cust\_L} \quad (4\text{-}29)$$

The average waiting time in the customer queue, denoted by $W_c$, is given by:

$$W_c = \frac{total\_queue\_time}{total\_served\_cust} \quad (4\text{-}30)$$

Eighth, the average waiting time of the high-priority and low-priority customers in the block queue, denoted by $W_{b_H}$ and $W_{b_L}$, respectively, is given by:

$$W_{b_H} = \frac{total\_block\_time\_H}{total\_served\_cust\_H} \tag{4-31}$$

$$W_{b_L} = \frac{total\_block\_time\_L}{total\_served\_cust\_L} \tag{4-32}$$

The average waiting time in the block queue, denoted by $W_b$, is given by:

$$W_b = \frac{total\_block\_time}{total\_served\_cust} \tag{4-33}$$

Finally, the average number of high-priority and low-priority blocks participating in the consensus process per unit of time, denoted by $B_{n_H}$ and $B_{n_L}$, respectively, is given by:

$$B_{n_H} = \frac{Area\_total\_num\_of\_block\_H}{sim\_time} \tag{4-34}$$

$$B_{n_L} = \frac{Area\_total\_num\_of\_block\_L}{sim\_time} \tag{4-35}$$

The average number of blocks participating in the consensus process per unit of time within a block, denoted by $B_n$, is given by:

$$B_n = \frac{Area\_total\_num\_of\_block}{sim\_time} \tag{4-36}$$

## 4.3. Scenario 3: Single-Class Customer with Impatience

In this simulation model, we consider a blockchain system that handles a single class of customers, where customers arrive according to a Poisson process and are served under the First-Come-First-Served (FCFS) discipline.

The system consists of two queues: the customer queue, where customers wait for block generation, and the block queue, where customers participate in the consensus process after being grouped into a block. Block generation follows a partial batch service policy, allowing 1 to $b$ customers to form a block. Once a block is formed, it is transferred to the block queue. Upon completion of the consensus process, all customers in the block exit the system.

During the OFF state, caused by interruptions such as attacks or connectivity issues, both block generation and consensus processes are suspended, although new customers may still arrive and be admitted. During the ON state, all services resume normally. To preserve system integrity, a constraint is imposed on the maximum number of customers allowed in the customer queue: when the block queue is empty, up to $N$ customers may wait; otherwise, the limit is reduced to $N - b$.

In this scenario, customers may abandon the system if they wait in the queue for too long without being served. Upon entering the customer queue, each customer is assigned a personal impatience threshold based on an exponential random variable with rate $\gamma$. If this threshold expires before the customer is grouped into a block, the customer abandons the queue and is removed from the system.

This scenario captures the impact of impatience-driven abandonment on system performance. It allows us to evaluate additional metrics such as impatience probability and its effect on throughput, queue occupancy, and overall system responsiveness under variable service delays and transient interruptions.

### 4.3.1. Main program

The main program executes a series of steps to simulate the blockchain queuing system with customer impatience, as illustrated in Figure 4-12. At the beginning of each simulation run, all relevant variables are initialized. This includes resetting statistical parameters, setting the next block generation time and next departure time to infinity, marking the system status as ON, initializing the block generation status as idle, and setting the customer queue limit to $N$.

Next, the system parameters are configured. These include the maximum customer queue capacity ($N$), the maximum number of customers per block ($b$), the arrival rate ($\lambda$), the block generation rate ($\mu_1$), the consensus rate ($\mu_2$), the impatience rate ($\gamma$), and the ON/OFF switching rates ($\alpha$ and $\beta$) for the system channel.

The program generates the next arrival time and channel switch time using exponential random variables based on the corresponding system parameters. Impatience threshold will be dynamically scheduled when customers enter the queue, rather than at initialization. During the simulation, it compares the scheduled times of five events and selects the earliest event to execute its corresponding subprogram.

Finally, a while loop is used to repeat the simulation until a predefined number of customer arrivals has been reached. Once this condition is met, the simulation terminates and the performance statistics are output.

### 4.3.2. Arrival Subprogram

Figure 4-13 illustrated the flow chart of the arrival subprogram, simulates the arrival of a new customer to the system. Upon invocation, the total number of arrivals is incremented, and the simulation time is updated to the scheduled arrival time. The time for the next arrival is then scheduled using an exponential interarrival time generated with the arrival rate $\lambda$. Then, the area calculation function is invoked to update all time-averaged statistics based on the elapsed time since the last event.

Next, the system checks whether the customer queue has reached its capacity limit.

- If the queue is full, the arriving customer is rejected, and the number of rejections is incremented.
- If the queue is not full, the customer is admitted to the queue. In this case, the number of customers in the system and in the queue are incremented. The customer's arrival time and corresponding impatience threshold are recorded as a pair in the queue log. The impatience threshold is generated based on an exponential random variable with rate $\gamma$, and represents the maximum amount of time the customer is willing to wait before abandoning the system.

The system determines whether to initiate block generation:

- If the channel status is in ON state, and block generator is idle, and this customer is the only one in the queue, a new block generation event is scheduled based on an exponential random variable with rate $\mu_1$.

- If the block generator is busy or the channel is OFF, the next block generation time is set to infinity to suspend the process.

Finally, the impatience tracker is updated by identifying the customer with the earliest threshold in the queue. The next impatience event is scheduled to occur at this earliest threshold point.

### 4.3.3. Block Generation Subprogram

Figure 4-14 illustrates the flow chart of the block generation subprogram, which simulates the initiation of a block generation process. When this event is triggered, the simulation time is updated to the scheduled block generation time. Then, the area calculation function is invoked to update all time-averaged statistics based on the elapsed time since the last event.

Next, the block generator status is set to busy, indicating that a block is currently being generated. To ensure sufficient space for the upcoming consensus process, the capacity limit of the customer queue is reduced from $N$ to $N - b$, where $b$ is the maximum number of customers allowed in a block.

The system then determines how many customers should be transferred from the queue into the block:

- If more than $b$ customers are waiting in the queue, exactly $b$ are selected.
- Otherwise, all remaining customers in the queue are moved into the block.

The number of customers transferred into the block is recorded, and the queue size is adjusted accordingly. A block departure event is then scheduled based on an exponential random variable with rate $\mu_2$. After this, the next block generation time is set to infinity to prevent immediate retriggering.

For each customer that enters the block, their corresponding arrival time is logged into the block log. These timestamps are subsequently used to compute the cumulative queueing time. This calculation is performed using the total waiting time function, which sums the time differences between the current simulation time and each customer's original queue entry time.

Finally, the corresponding entries in the queue log are removed to reflect that these customers have exited the queue and are now participating in the consensus process. The system then updates the impatience tracker by identifying the next customer with

the earliest remaining impatience threshold. The next impatience event is scheduled accordingly.

### 4.3.4. Block Departure Subprogram

Figure 4-15 illustrates the flow chart of the departure subprogram, which simulates the completion of a block consensus process. When this event is triggered, the simulation time is updated to the scheduled block departure time. Then, the area calculation function is invoked to update all time-averaged statistics based on the elapsed time since the last event.

At this point, the block generation status is reset to idle, and the customer queue capacity limit is restored to its original value $N$, allowing the queue to accept new customers at full capacity. The block departure event is considered completed and is therefore cleared.

The program then calculates the total time that the current block of customers spent in the consensus stage. This is achieved using the block time accumulation function, which computes the total time difference between the current simulation time and each customer's recorded entry into the block.

After consensus completion, the number of customers currently in the system is decreased by the number of customers in the departing block, and the total number of customers served is incremented accordingly. The block is now empty, and all associated entries in the block log are removed.

Finally, if there are still customers waiting in the queue, a new block generation event is scheduled based on an exponential random variable with rate $\mu_1$.

### 4.3.5. Switch Subprogram

Figure 4-16 illustrates the flow chart of the switch subprogram, which simulates the transition of the system between ON and OFF states. When this event is triggered, the simulation time is updated to the scheduled switch time. Then, the area calculation function is invoked to update all time-averaged statistics based on the elapsed time since the last event.

The system channel status is then toggled as follows:

- **If the system transitions from ON to OFF:**
    - The channel status is set to OFF.

    - The next switch event is scheduled based on an exponential random variable with rate $\beta$ (representing the OFF duration).
    - All ongoing service operations are suspended by setting both the block generation and block departure event times to infinity.
- **If the system transitions from OFF to ON:**
    - The channel status is set to ON.
    - The next switch event is scheduled based on an exponential random variable with rate $\alpha$ (representing the ON duration).
    - If there is at least one customer in the queue and the block generator is currently idle:
        - A block generation event is scheduled based on an exponential random variable with rate $\mu_1$.
    - If a block in consensus phase is suspended:
        - A block departure event is scheduled using an exponential random variable with rate $\mu_2$.

Through this subprogram, the simulation captures the stochastic availability of the system by alternating between operational and suspended phases, reflecting real-world unreliability such as downtime or external disruptions. During the ON period, block generation and consensus operations proceed as normal. During the OFF period, these processes are temporarily halted while new customer arrivals may still occur.

### 4.3.6. Impatience subprogram

Figure 4-17 illustrates the flow chart of the impatience subprogram, which simulates the event in which a customer abandons the system after exceeding their impatience threshold. When this event is triggered, the simulation time is updated to the scheduled impatience threshold. The area calculation function is then invoked to update all time-averaged statistics based on the elapsed time since the last event.

The customer whose impatience threshold has expired is removed from the queue and from the system. Accordingly, the number of customers in the system and in the queue are both decremented, and the total number of impatient departures is incremented.

The actual waiting time of the abandoning customer is calculated as the difference between their impatience threshold and their recorded arrival time. This value is accumulated as part of the total impatient time, which is used to compute impatience-related performance metrics.

The customer's entry is then removed from the queue log to reflect their departure. Following this update, the system searches for the next earliest impatience threshold among the remaining customers in the queue. The impatience tracker is updated, and the next impatience event is scheduled accordingly.

### 4.3.7. Flowchart Diagram

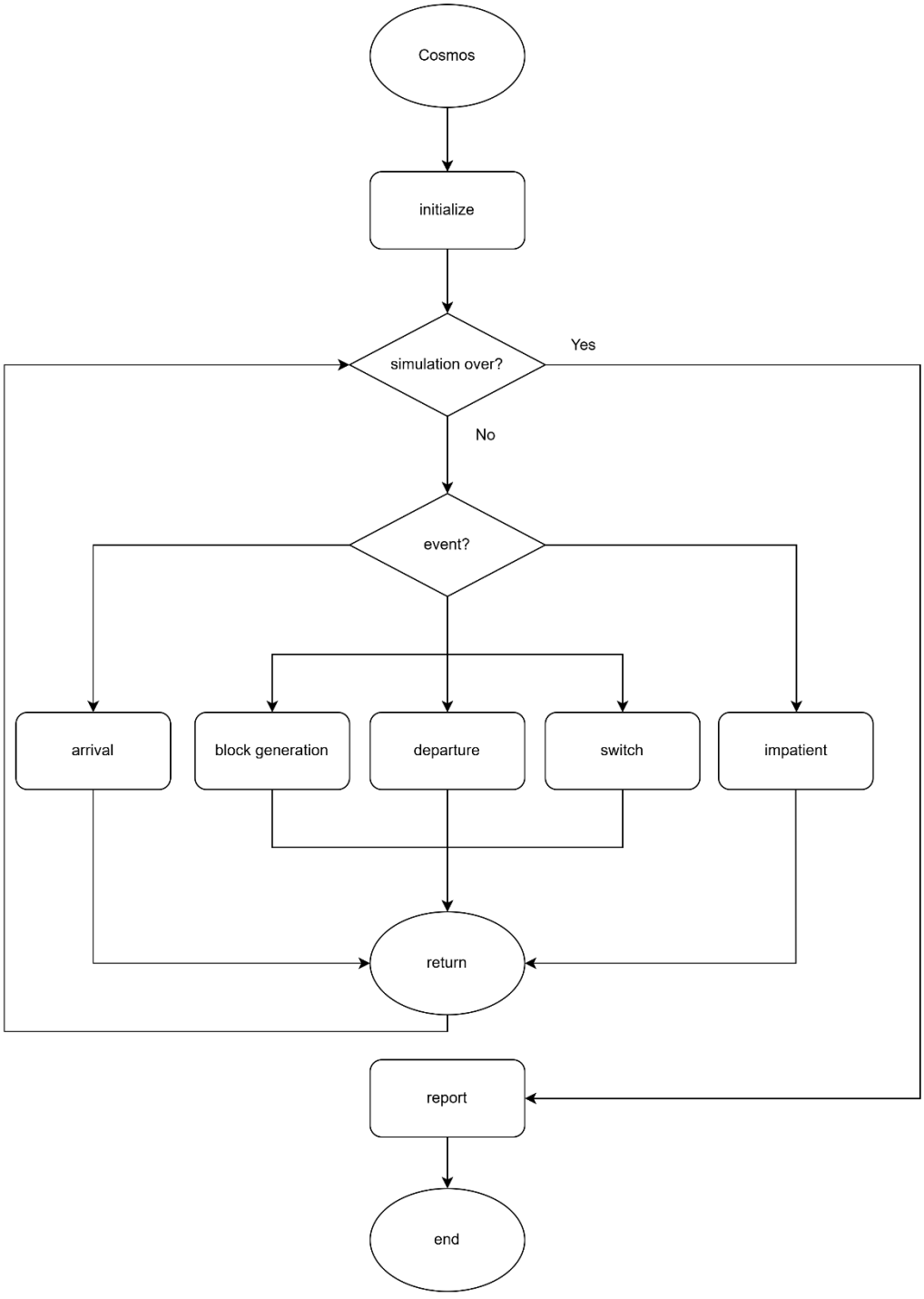


Figure 4-12: Flow chart of main program

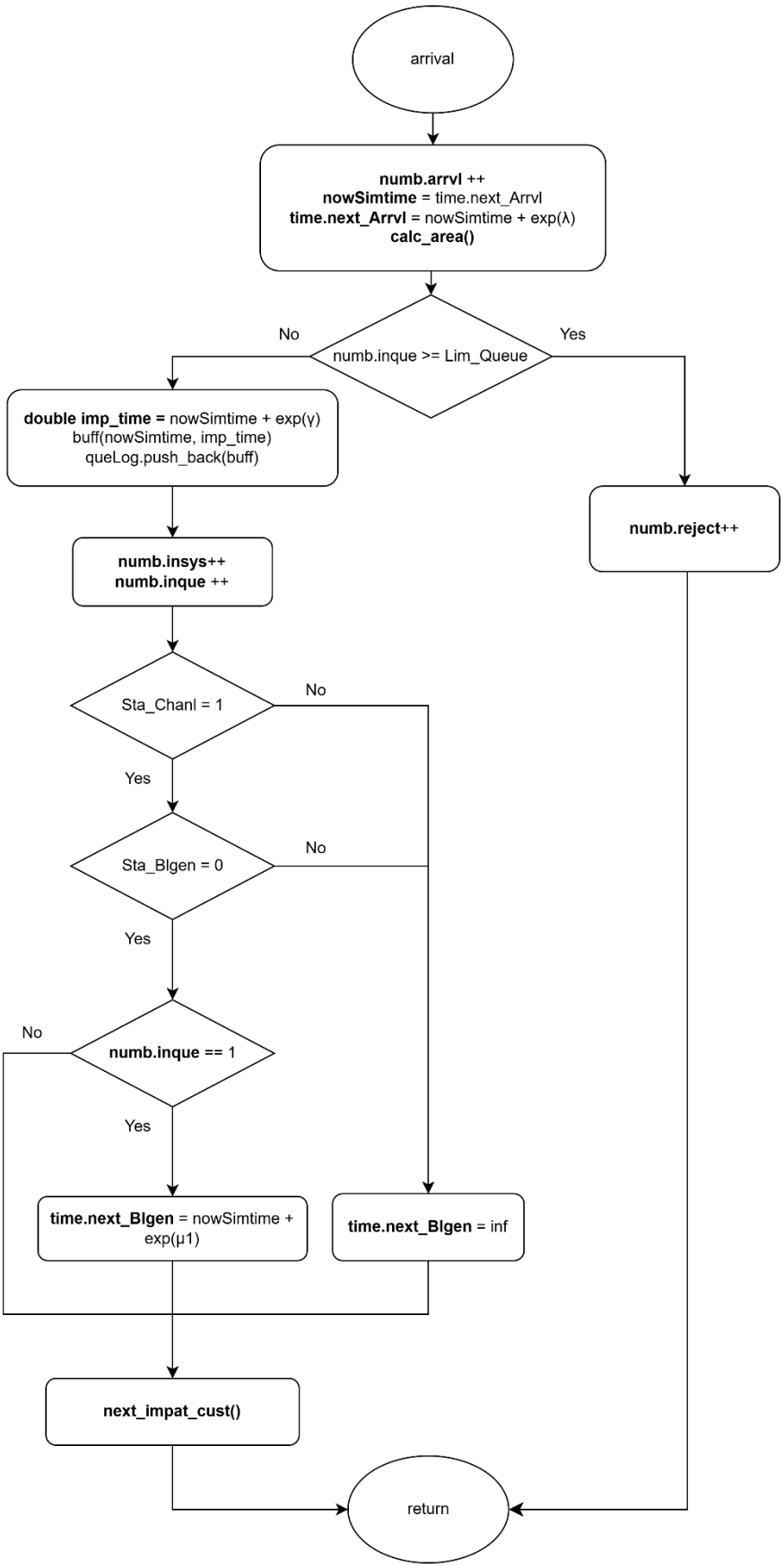


Figure 4-13: Flow chart of arrival subprogram

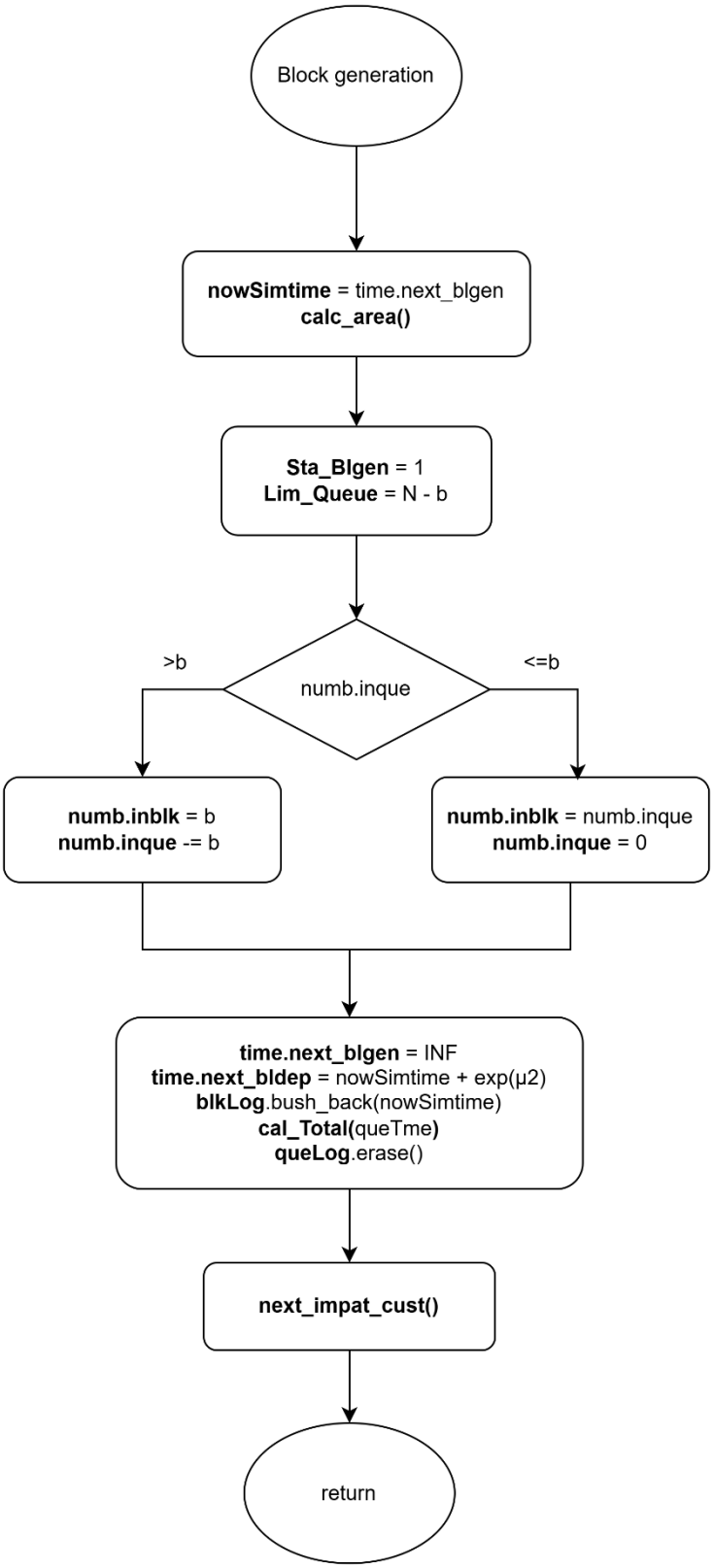


Figure 4-14: Flow chart of block generation subprogram

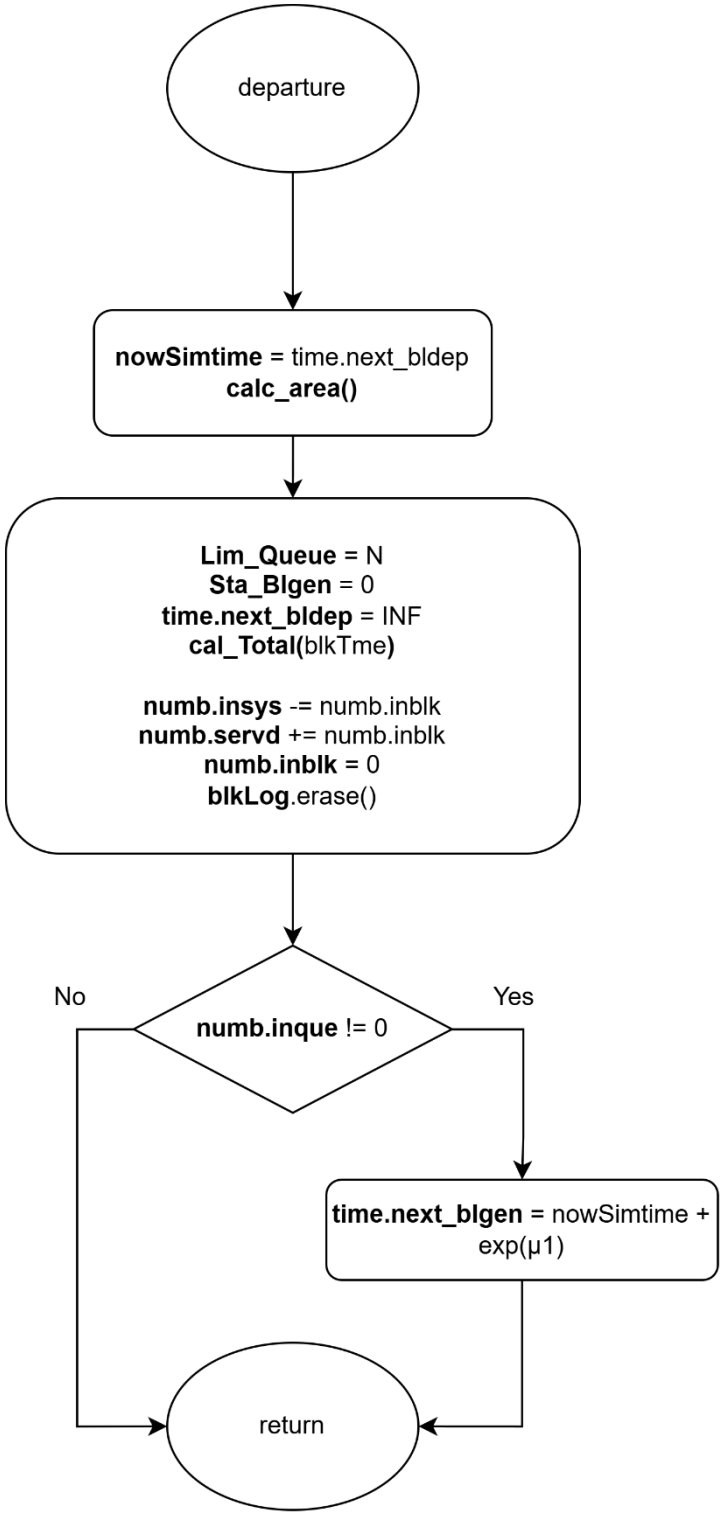


Figure 4-15: Flow chart of block departure subprogram

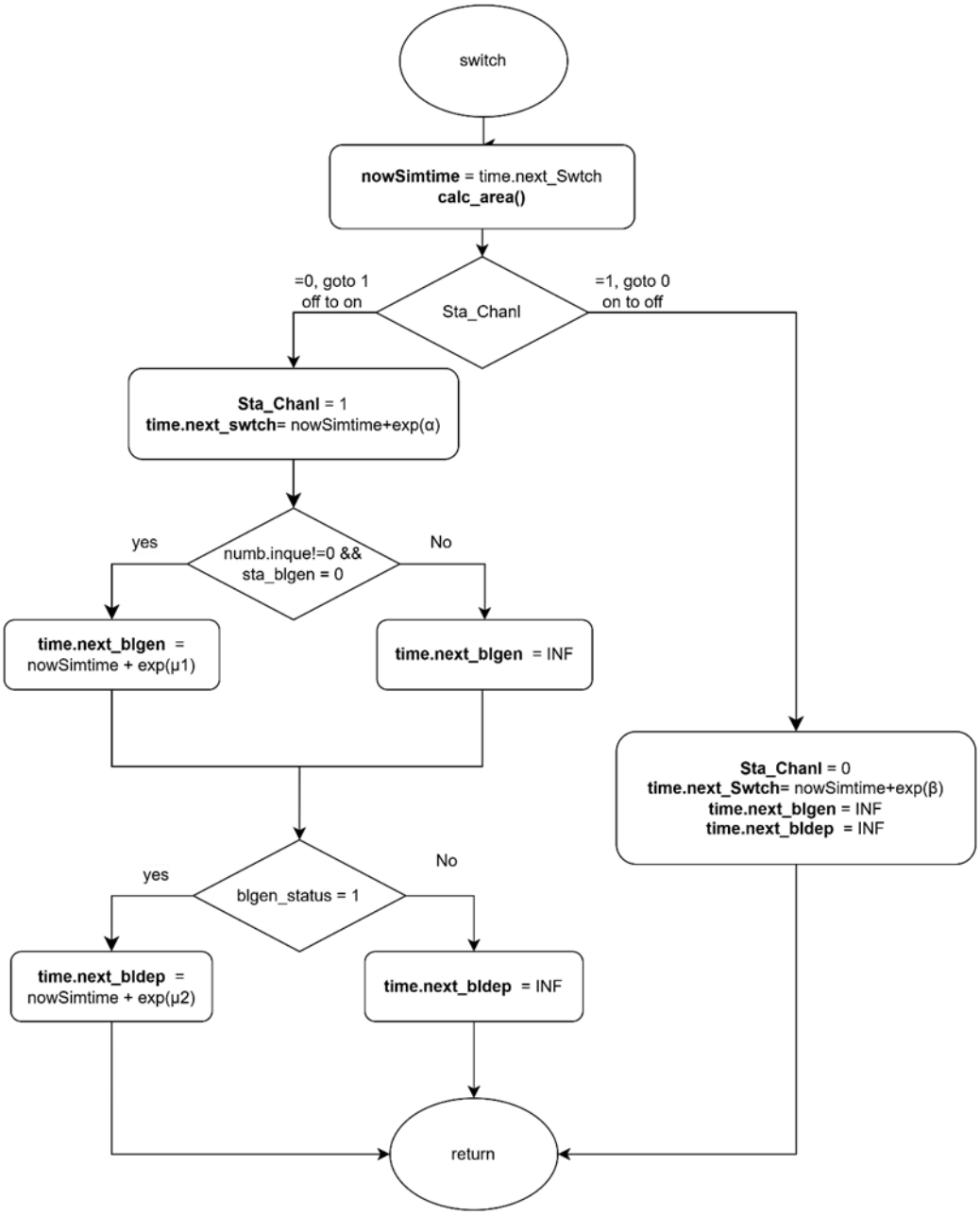


Figure 4-16: Flow chart of switch subprogram

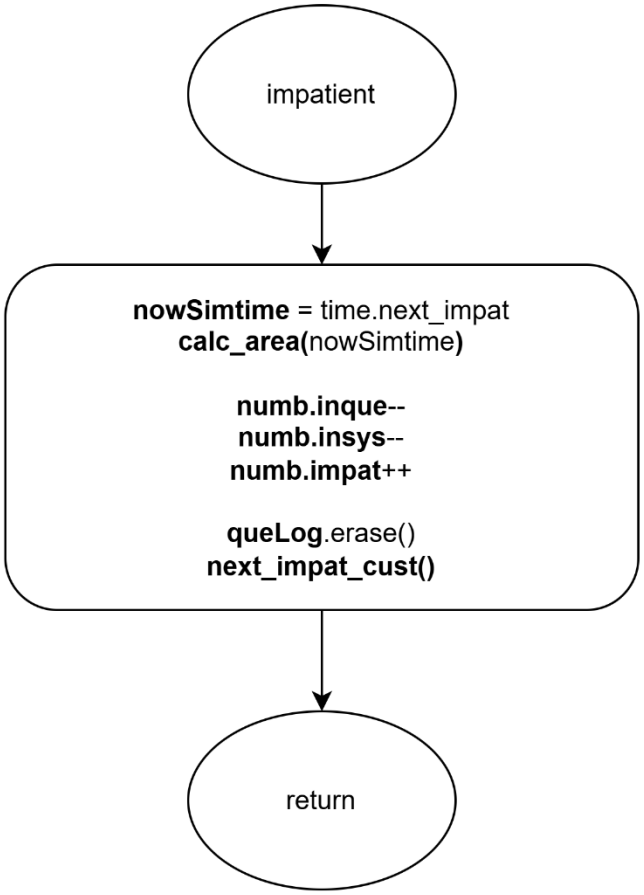


Figure 4-17: Flow chart of impatience subprogram

### 4.3.8. Performance Index

First of all, the average number of customers in the whole system, denoted by $L$, is given by:

$$L = \frac{Area\_total\_cust\_in\_system}{sim\_time} \tag{4-37}$$

Second, the average number of customers in customer queue, denoted by $L_c$, is given by:

$$L_c = \frac{Area\_total\_cust\_in\_queue}{sim\_time} \tag{4-38}$$

Third, the average number of customers in block queue, denoted by $L_b$, is given by:

$$L_b = \frac{Area\_total\_cust\_in\_block}{sim\_time} \quad (4\text{-}39)$$

Fourth, the blocking probability of the system, denoted by $P_b$, is given by:

$$P_b = \frac{total\_rejected\_cust}{total\_num\_of\_arrival} \quad (4\text{-}40)$$

Fifth, the impatient probability of the system, denoted by $P_{im}$, is given by:

$$P_{im} = \frac{total\_impat\_cust}{total\_num\_of\_arrival - total\_rejected\_cust} \quad (4\text{-}41)$$

Sixth, the throughput of the system, denoted by $T_h$, is given by:

$$T_h = \frac{total\_served\_cust}{sim\_time} \quad (4\text{-}42)$$

Seventh, the average waiting time in the customer queue, denoted by $W_c$, is given by:

$$W_c = \frac{total\_queue\_time + total\_impat\_time}{total\_served\_cust + total\_impat\_cust} \quad (4\text{-}43)$$

Eighth, the average waiting time in the block queue, denoted by $W_b$, is given by:

$$W_b = \frac{total\_block\_time}{total\_served\_cust} \quad (4\text{-}44)$$

Ninth, the average waiting time in the system, denoted by $W$, is given by:

$$W = W_c + W_b \quad (4\text{-}45)$$

Finally, the average number of blocks participating in the consensus process per unit of time, denoted by $B_n$, is given below.

$$B_n = \frac{Area\_total\_num\_of\_block}{sim\_time} \quad (4\text{-}46)$$

## 4.4. Scenario 4: Two-Class Customer with Impatience

In this simulation model, we consider a blockchain system that handles two classes of customers: high-priority and low-priority customers. Customers of different priorities arrive according to an independent Poisson process, and customers of the same priority are served under the First-Come-First-Served (FCFS) discipline. Customers of different priorities are handled using a non-preemptive priority rule, where high-priority customers are always placed ahead of low-priority customers in the queue, but ongoing service for a low-priority block cannot be interrupted once initiated.

The system consists of two queues: the customer queue and the block queue. High-priority and low-priority customers both enter the customer queue upon arrival if possible. Block generation follows a partial batch service policy and operates on customers of only one priority class at a time. Each block can include at most $b$ customers. Once a block that is composed solely of high-priority or low-priority customer(s) is formed, it is transferred to the block queue. After completing the consensus process, all customers in the block exit the system.

The system alternates between ON and OFF periods to reflect external interruptions such as cyberattacks or connection failures. During the OFF state, both block generation and consensus process are suspended, though new customers may still arrive and be queued. During the ON state, all services resume as normal.

To preserve system integrity and fairness, different queue capacity constraints apply based on customer class and system state. For high-priority customers, the maximum number allowed in the customer queue is $N$ when the block queue is empty, and $N - b$ when it is occupied. For low-priority customers, the customer queue capacity is always limited to $N - b$, regardless of the block queue state.

In this scenario, customers may abandon the system if they wait in the queue too long without being served. Upon entering the customer queue, each of high-priority and low-priority customers are assigned a personal impatience threshold based on an exponential random variable with rate $\gamma_H$ and $\gamma_L$, respectively. If this threshold expires before the customer is grouped into a block, the customer abandons the queue and is removed from the system.

This scenario is designed to study the combined effects of non-preemptive priority scheduling and customer impatience. It enables performance evaluation in terms of class-specific metrics such as throughput, queue length, waiting time, blocking probability, and impatience probability under dynamic and congested conditions.

### 4.4.1. Main program

The main program executes a series of steps to simulate the blockchain queuing system with two classes of customers, as illustrated in Figure 4-18. At the beginning of each simulation run, all relevant variables are initialized. This includes resetting statistical parameters, setting the next block generation time and next departure time to infinity, marking the system status as ON, initializing the block generation status as idle, and setting the customer queue limit to $N$.

Next, the system parameters are configured. These include the maximum customer queue capacity ($N$), the maximum number of customers per block ($b$), the arrival rates for high-priority and low-priority customers ($\lambda_H$ and $\lambda_L$), the block generation rates ($\mu_{1_H}$ and $\mu_{1_L}$), the consensus rates ($\mu_{2_H}$ and $\mu_{2_L}$), the impatience rates ($\gamma_H$ and $\gamma_L$), and the ON/OFF switching rates ($\alpha$ and $\beta$) for the system channel.

The program then generates the next arrival times for both high-priority and low-priority customers, as well as the channel switch time, using exponential random variables based on the corresponding system parameters. During the simulation, it compares the scheduled times of five events and selects the earliest event to execute its corresponding subprogram.

Finally, a while loop is used to repeat the simulation until a predefined number of customer arrivals has been reached. Once this condition is met, the simulation terminates and the performance statistics are output.

### 4.4.2. High-Priority Arrival Subprogram

As illustrated in Figure 4-19, the high-priority arrival subprogram simulates the arrival of a high-priority customer to the system. When this event is triggered, the simulation time is updated to the current arrival time. The arrival counters are incremented to reflect both the total number of customers and the number of high-priority arrivals. The next arrival event for high-priority customers is then scheduled based on an exponential random variable with rate $\lambda_H$. Then, the area calculation function is invoked to update all time-averaged statistics based on the elapsed time since the last event.

Next, the system checks whether the customer queue has reached its capacity limit.

- If the queue is full, the arriving customer is rejected. In this case, both the total number of rejections and the number of high-priority rejections are incremented.

- If the queue is not full, the arriving customer is admitted. The number of customers in the system and in the queue are both incremented, along with their corresponding high-priority counts. The customer's arrival time and corresponding impatience threshold are recorded as a pair in the queue log. The impatience threshold is generated based on an exponential random variable with rate $\gamma_H$, and represents the maximum amount of time the customer is willing to wait before abandoning the system.

Then, the system updates the unified queue log by merging the high-priority and low-priority arrival records, ensuring that high-priority entries appear first. The system then sets the priority flag to indicate that high-priority customers are currently at the head of the queue.

The system determines whether to initiate block generation:

- If the channel status is in ON state, and block generator is idle, and this customer is the only one in the queue, a new block generation event is scheduled based on an exponential random variable with rate $\mu_{1_H}$.
- If the block generator is busy or the channel is OFF, the next block generation time is set to infinity to suspend the process.

Finally, the impatience tracker is updated by identifying the customer with the earliest threshold in the queue. The next impatience event is scheduled to occur at this earliest threshold point.

### 4.4.3. Low-Priority Arrival Subprogram

As illustrated in Figure 4-20, the low-priority subprogram simulates the arrival of a low-priority customer to the system. When this event is triggered, the simulation time is updated to the scheduled arrival time. The arrival counters are incremented to reflect both the total number of customers and the number of low-priority arrivals.

The next arrival event for low-priority customers is then scheduled based on an exponential random value with the rate $\lambda_L$. Then, the area calculation function is invoked to update all time-averaged statistics based on the elapsed time since the last event.

Next, the system checks whether the customer queue has reached its capacity limit. The rejection condition depends on the system’s block generation state:

- If the number of low-priority customers in the queue has reached $N - b$, or
- The total number of customers in the queue has reached the current queue limit,

then the arriving customer is rejected. In this case, both the total number of rejections and the number of low-priority rejections are incremented.

If neither of the above two rejection conditions is met, the arriving customer is admitted. The number of customers in the system and in the queue are both incremented, along with their corresponding low-priority counts. The customer's arrival time and corresponding impatience threshold are recorded as a pair in the low-priority queue log. The impatience threshold is generated based on an exponential random variable with rate $\gamma_L$, and represents the maximum amount od time the customer is willing to wait before abandoning the system.

To maintain unified tracking, the system then refreshes the combined queue log by merging both priority records, ensuring high-priority entries appear first. If there are no high-priority customers currently in the queue, the system sets the priority flag to indicate that low-priority customer is now at the head of the queue.

Finally, the system determines whether to initiate block generation:

- If the channel status is in ON state, and block generator is idle, and this customer is the only one in the queue, a new block generation event is scheduled based on an exponential random variable with rate $\mu_{1_L}$.
- If the block generator is busy or the channel is OFF, the next block generation time is set to infinity to suspend the process.

Finally, the impatience tracker is updated by identifying the customer with the earliest threshold in the queue. The next impatience event is scheduled to occur at this earliest threshold point.

### 4.4.4. Block Generation Subprogram

As illustrated in Figure 4-21, the block generation subprogram simulates the initiation of a block generation process. When this event is triggered, the simulation time is updated to the scheduled block generation time. Then, the area calculation function is invoked to update all time-averaged statistics based on the elapsed time since the last event. The next block generation time is set to infinity to prevent immediate retriggering, and the block generator status is marked as active.

To reserve sufficient space in the queue for block formation, the maximum queue size is reduced from $N$ to $N - b$.

The system then determines the class of customers from which the block will be constructed, based on the current queue head status:

- **If the high-priority customer is at the head of the queue**:
    - If the number of high-priority customers exceeds the block size $b$, exactly $b$ of them are moved into the block.
    - Otherwise, all high-priority customers are included in the block. If low-priority customers remain in the queue, the queue head status is updated accordingly; otherwise, it is cleared.
    - The block is marked as high-priority, and the consensus process is scheduled based on an exponential random variable with rate $\mu_{2_H}$.
    - The total waiting time of high-priority customers in the block is calculated using their individual arrival times stored in the high-priority queue log, which is then updated by removing the corresponding entries, thereby reflecting their transition into the consensus stage.
- **If low-priority customer is at the head of the queue**:
    - If there are more than $b$ low-priority customers, exactly $b$ are selected for the block.
    - Otherwise, all remaining low-priority customers are included, and the queue is emptied.
    - The block is marked as low-priority, and the block departure event is scheduled based on an exponential random variable with rate $\mu_{2_L}$.
    - The total waiting time of low-priority customers is calculated using their queue log, which is then updated accordingly.

After the block is formed, the system updates the overall waiting time in the queue for all customers, using the unified queue log. The combined queue log is then refreshed to reflect the current state of the customer queue.

Next, the block log is updated with the entry time for each customer in the new block. This log is used for downstream statistics related to block-based consensus activity.

Finally, the impatience tracker is updated by identifying the customer with the earliest threshold in the queue. The next impatience event is scheduled to occur at this earliest threshold point.

### 4.4.5. Block Departure Subprogram

As illustrated in Figure 4-22, the block departure subprogram simulates the completion of a block's consensus process. When this event is triggered, the simulation time is updated to the current departure time. The area calculation function is then invoked to update all time-averaged statistics based on the elapsed time since the last event. The next block departure time is set to infinity to suspend further departure scheduling until a new block is formed.

The block generator status is reset to idle, and the customer queue capacity limit is restored to its original value $N$, allowing the system to admit new arrivals without restriction.

The total waiting time of the block in the consensus stage is accumulated using the block log. The total number of customers currently in the system is decreased by the number of customers in the departing block, and the number of customers served is incremented accordingly. The block is then cleared.

The next processing depends on the priority class of the departing block:

- If it is a high-priority block, the corresponding class-specific consensus time is updated, and the number of high-priority customers in the system and served counters are adjusted.
- If it is a low-priority block, the low-priority statistics are updated in a similar manner.

After processing, the system clears the block log and resets the priority status of the block.

Finally, the system checks whether customers remain in the queue. If so, a new block generation event is scheduled based on the class of the customer at the head of the queue:

- If the high-priority customer is at the head of the queue, the next block generation is scheduled based on an exponential random variable with rate $\mu_{1_H}$.
- If the low-priority customer is at the head of the queue, the event is scheduled with rate $\mu_{1_L}$.

If the queue is empty, no block generation is scheduled, and the next block departure time is set to infinity.

### 4.4.6. Switch Subprogram

As illustrated in Figure 4-23, the switch subprogram simulates the transition of the system between ON and OFF operational states. When this event is triggered, the simulation time is updated to the current switch time. The area calculation function is then invoked to update all time-averaged statistics based on the elapsed time since the last event.

The system channel status is then toggled as follows:

- **If the system transitions from ON to OFF:**
  - The channel status is updated to OFF.
  - The next switch event is scheduled based on an exponential random variable with rate β (representing the OFF duration).
  - All pending block generation and block departure events are suspended by setting their scheduled times to infinity.
- **If the system transitions from OFF to ON:**
  - The channel status is updated to ON.
  - The next switch event is scheduled based on an exponential random variable with rate α (representing the ON duration).
  - If there is at least one customer in the queue and the block generator is idle:
    - A block generation event is scheduled based on an exponential random variable with rate $\mu_{1_H}$ or $\mu_{1_L}$, depending on whether the high- or low-priority customer is at the head of the queue.
  - If a block in consensus phase is suspended:
    - A block departure event is scheduled using the appropriate rate ($\mu_{2_H}$ or $\mu_{2_L}$) based on the class of the current block.

Through this subprogram, the system emulates the effects of environmental disruptions such as connectivity loss or cyberattacks by alternating between active and inactive service periods. During the ON state, queuing, block generation and consensus operations proceed. During the OFF state, only customer arrivals are allowed, while block generation and consensus operation are temporarily halted.

### 4.4.7. Impatience subprogram

Figure 4-24 illustrates the flow chart of the impatience subprogram, which simulates the event in which a customer abandons the system after exceeding their impatience threshold. When this event is triggered, the simulation time is updated to

the scheduled impatience threshold. The area calculation function is then invoked to update all time-averaged statistics based on the elapsed time since the last event.

The customer whose impatience threshold has expired is removed from the queue and from the system. Depending on the customer's priority class, the number of customer in the system and in the queue are both decremented, and the total number of impatient departures is incremented for both the system and the corresponding class.

The actual waiting time of the abandoning customer is calculated as the difference between their impatience threshold and their recorded arrival time. This value is accumulated as part of the total impatient time, which is used to compute impatience-related performance metrics.

The customer's entry is then removed from the corresponding priority queue log to reflect their departure. The unified queue log is refreshed to reflect the current state of the system.

After this update, the system checks the remaining customers in the queue. If the queue is empty, the next block generation time is set to infinity, and the priority flag is cleared. If at least one customer remains, the system sets the queue head priority to high if at least one high-priority customer is still waiting; otherwise, it sets it to low.

Finally, the impatience tracker is updated by identifying the customer with the earliest threshold in the queue. The next impatience event is scheduled to occur at this earliest threshold point.

### 4.4.8. Flowchart Diagram

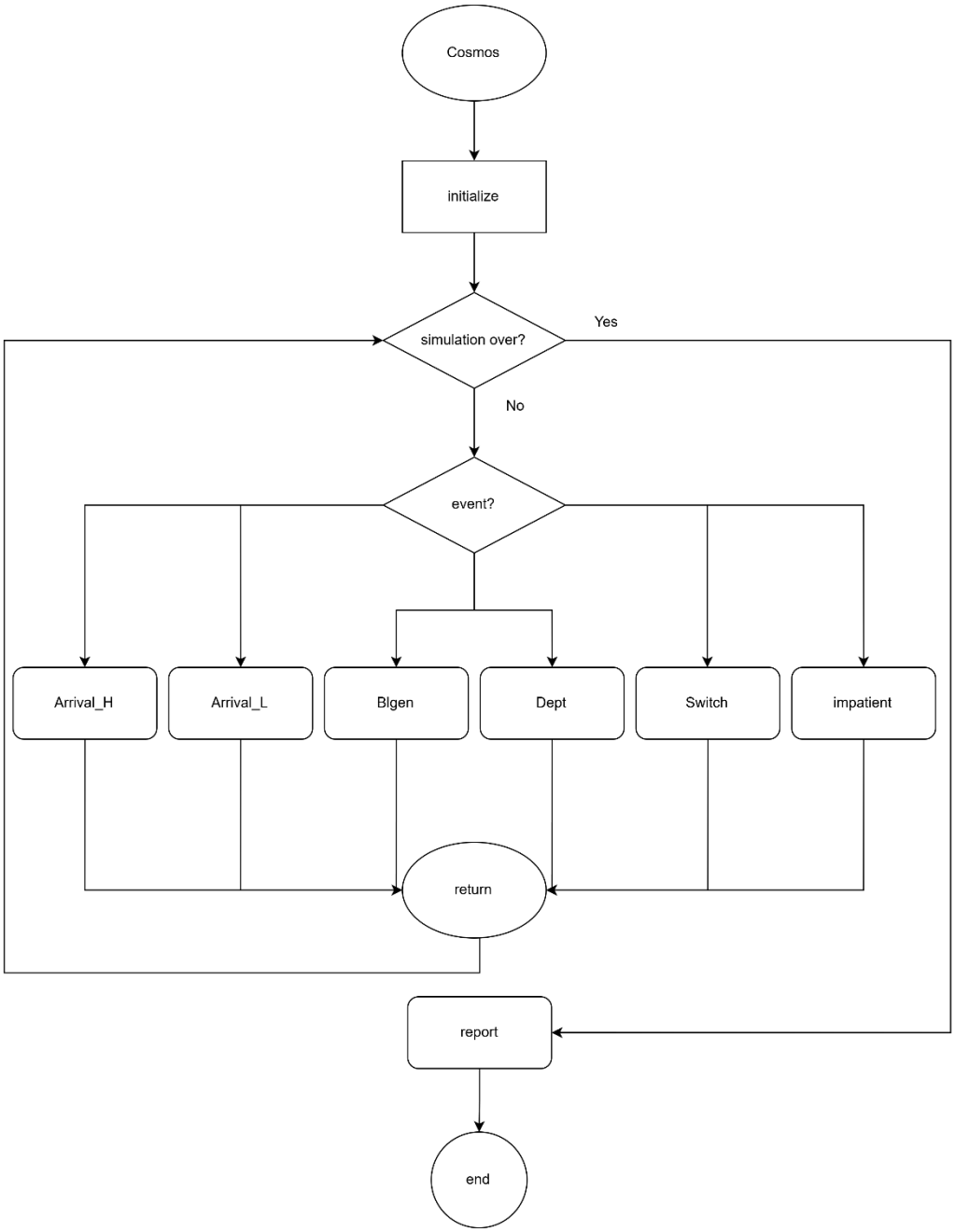


Figure 4-18: Flow chart of main program

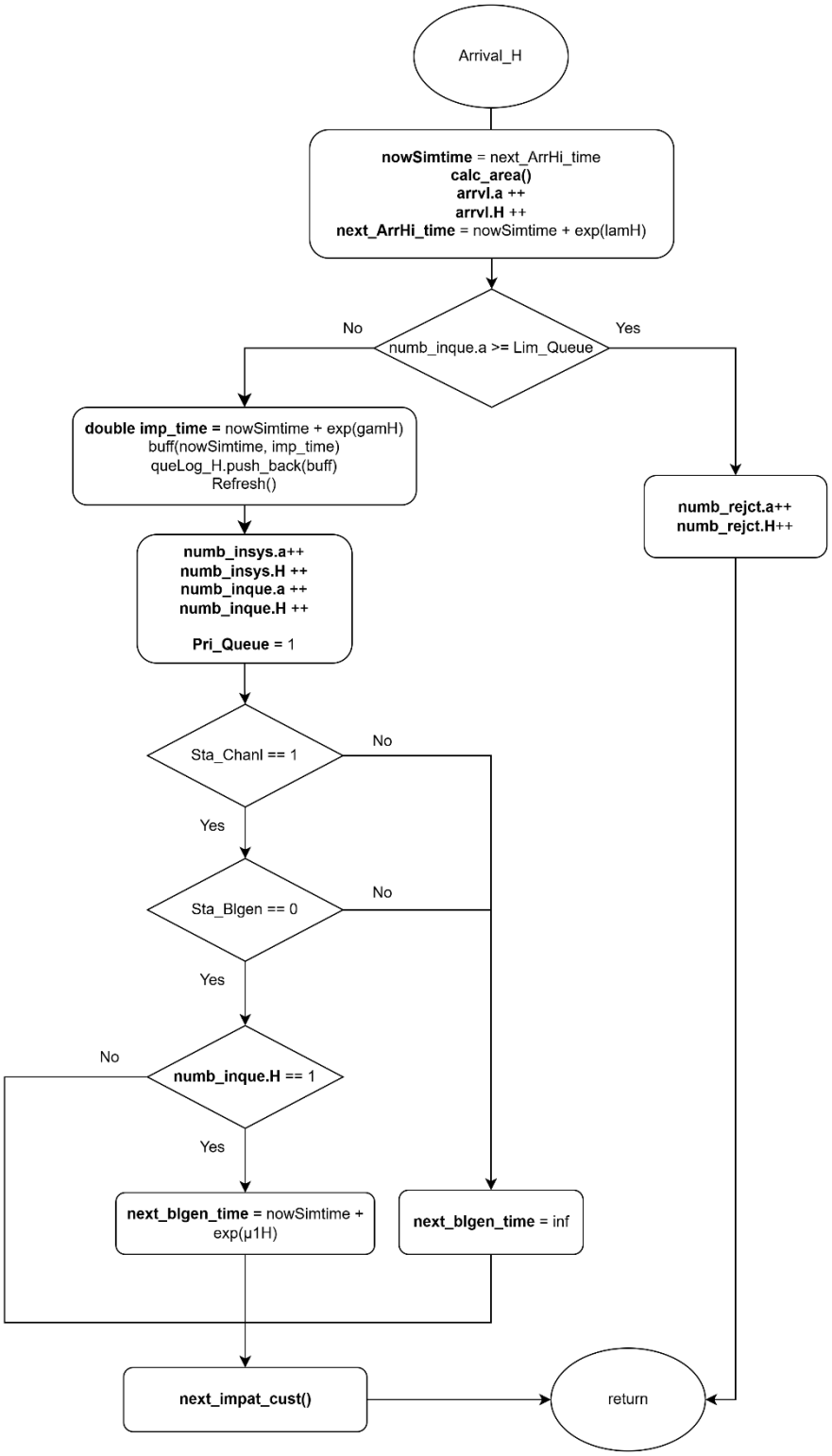


Figure 4-19: Flow chart of high-priority arrival subprogram

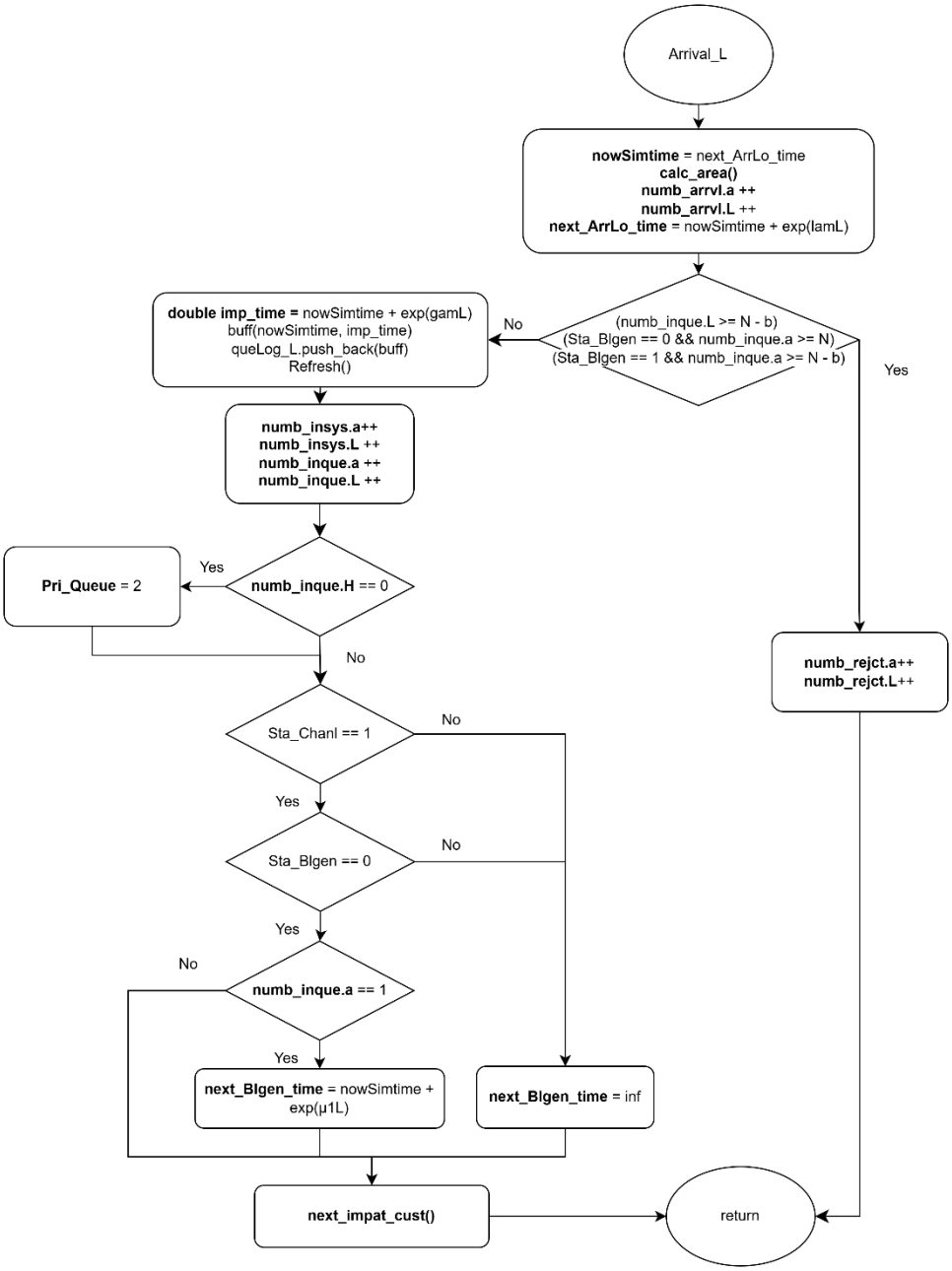


Figure 4-20: Flow chart of low-priority arrival subprogram

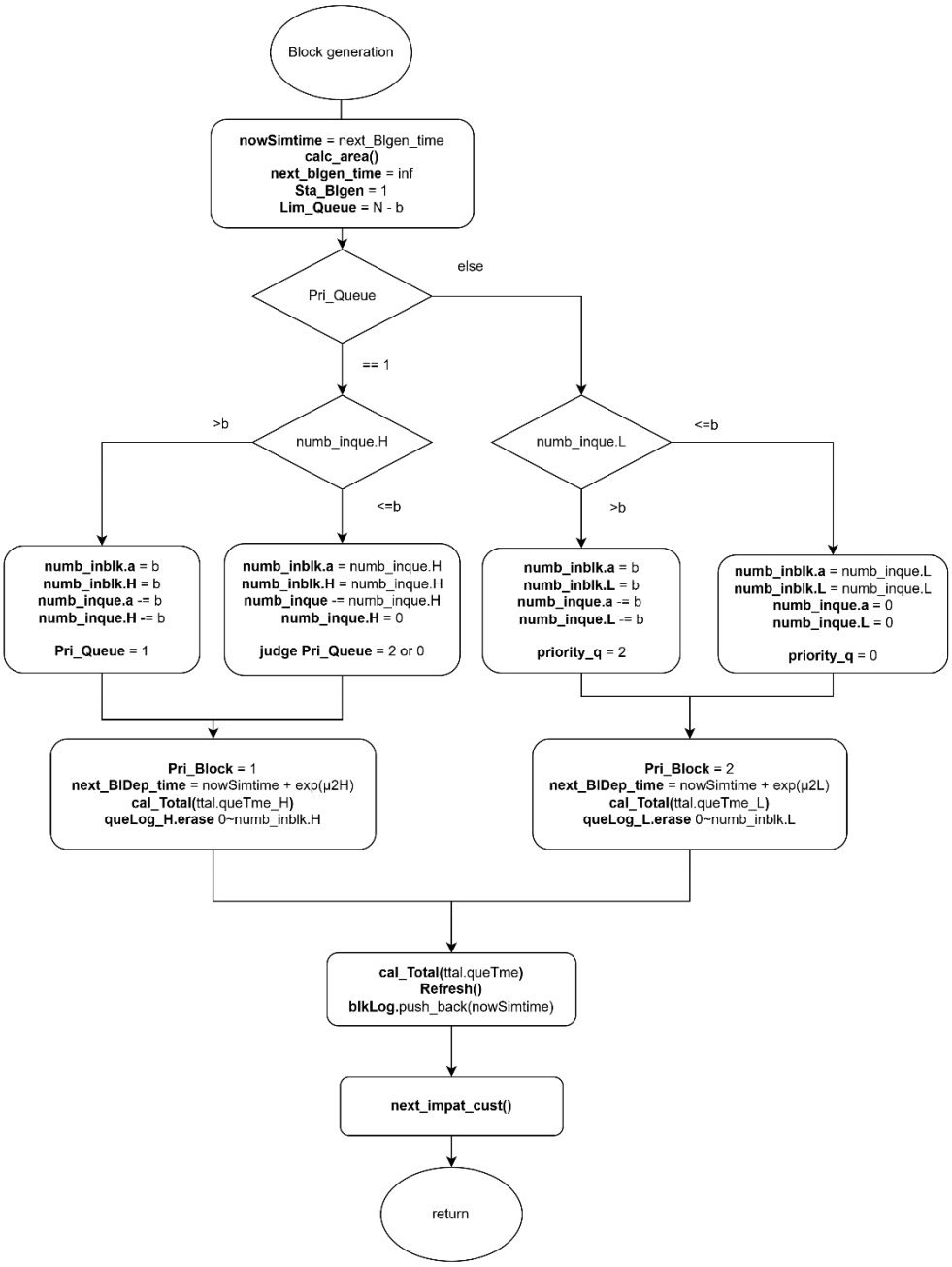


Figure 4-21: Flow chart of block generation subprogram

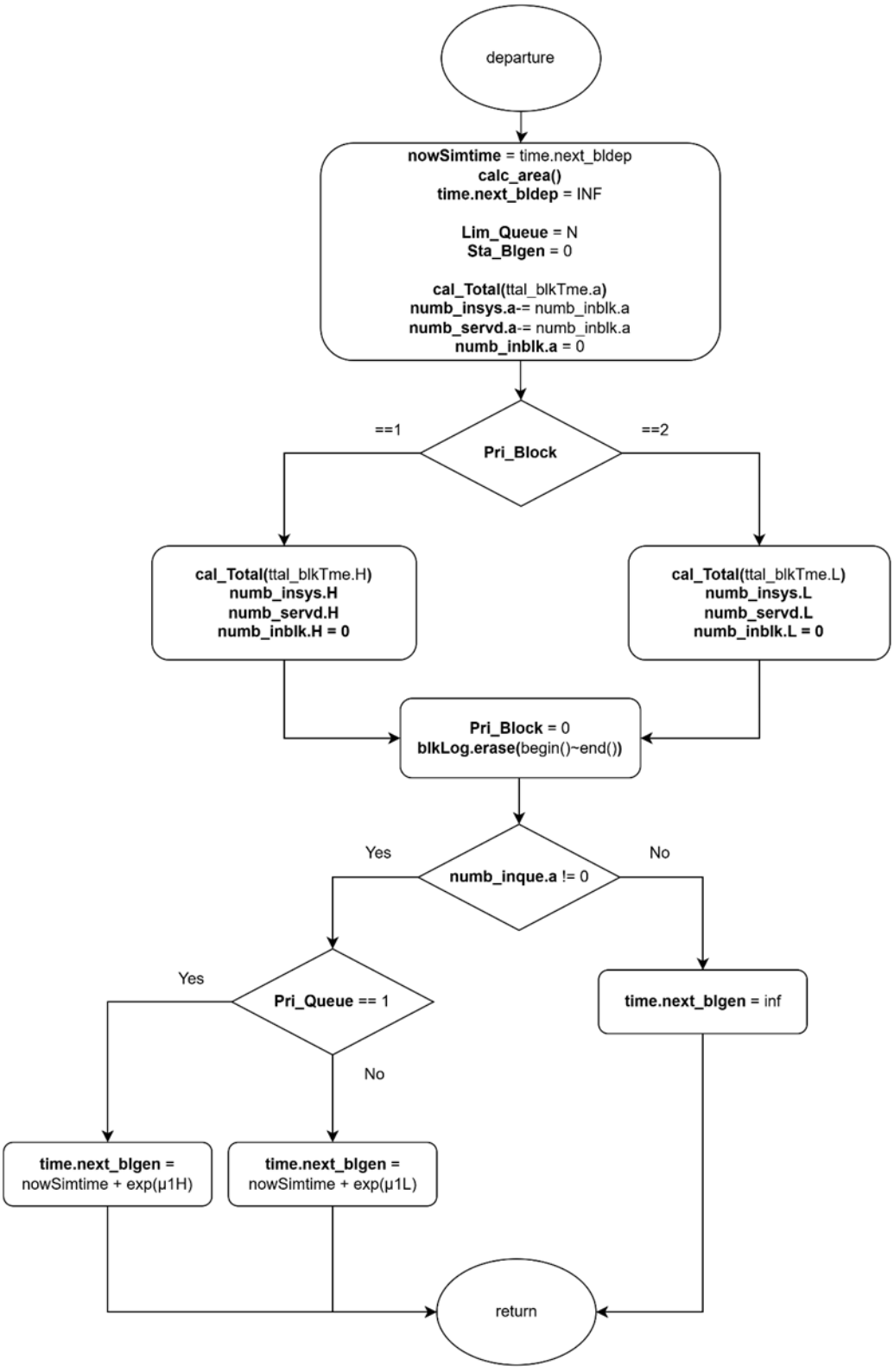


Figure 4-22: Flow chart of block departure subprogram

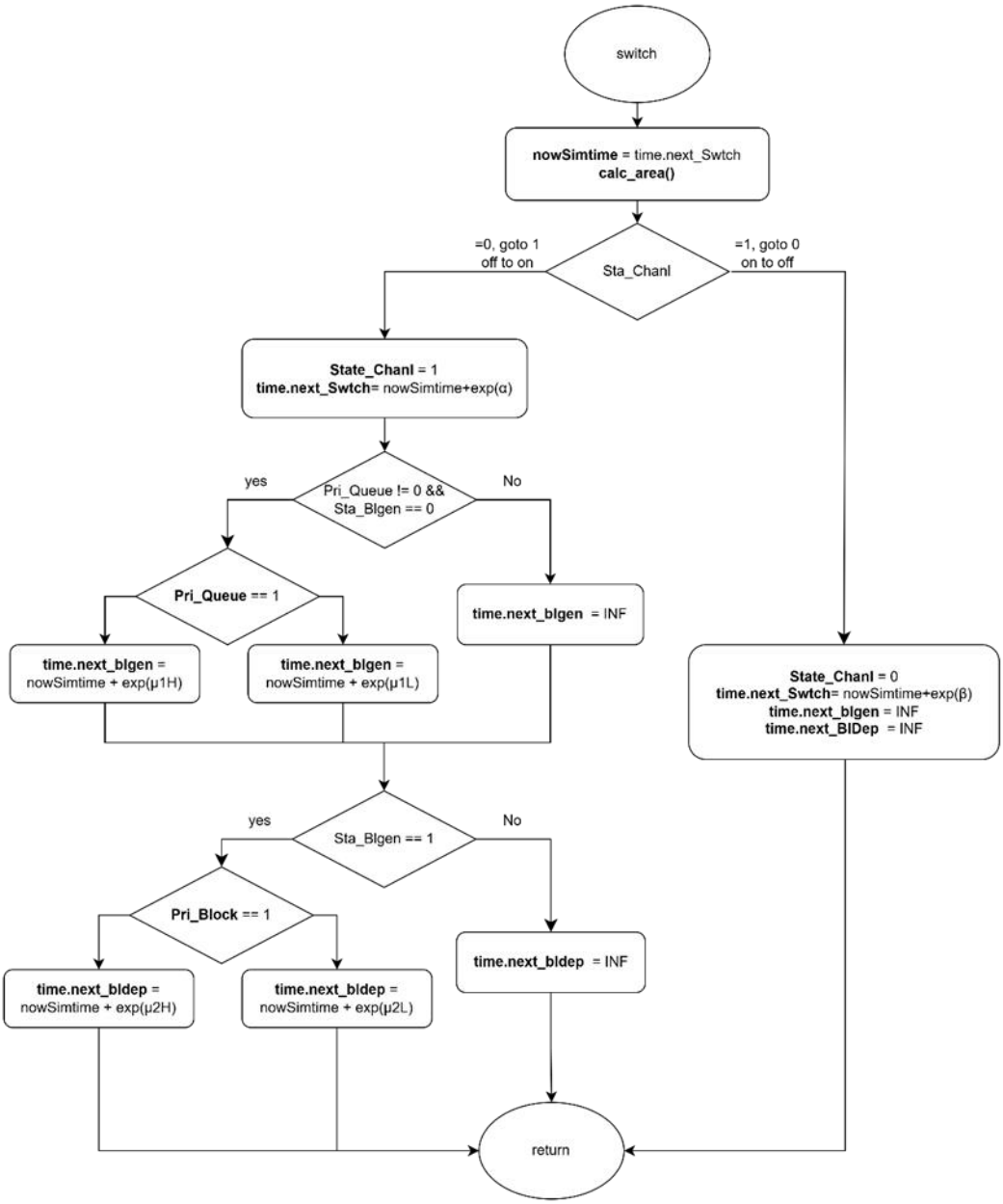


Figure 4-23: Flow chart of switch subprogram

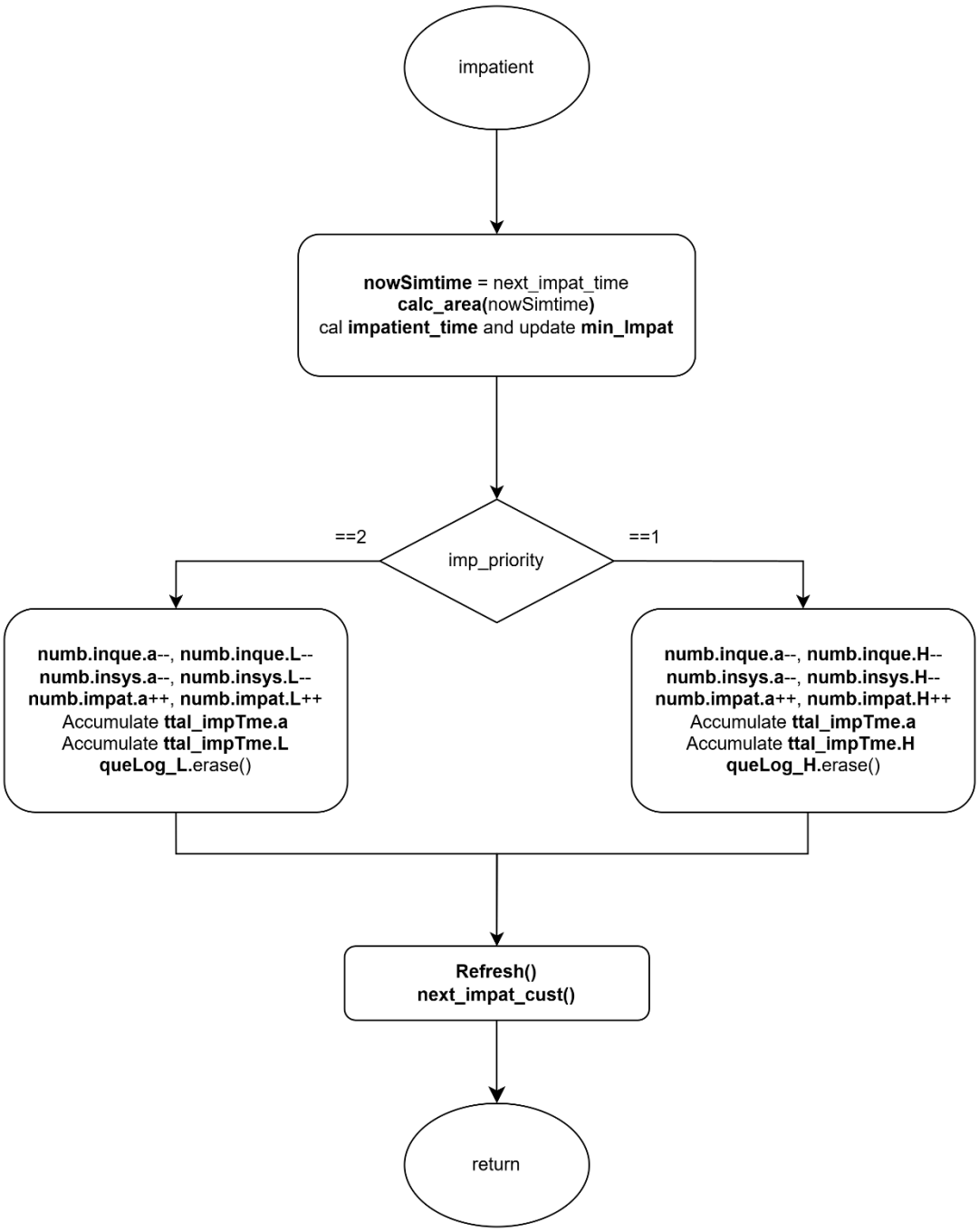


Figure 4-24: Flow chart of impatient subprogram

### 4.4.9. Performance Index

First of all, the average number of high-priority and low-priority customers in the whole system, denoted by $L_H$ and $L_L$, respectively, is given by:

$$L_H = \frac{Area\_total\_cust\_in\_system\_H}{sim\_time} \tag{4-47}$$

$$L_L = \frac{Area\_total\_cust\_in\_system\_L}{sim\_time} \quad (4\text{-}48)$$

The average number of customers in the whole system, denoted by $L$, is given by:

$$L = \frac{Area\_total\_cust\_in\_system}{sim\_time} \quad (4\text{-}49)$$

Second, the average number of high-priority and low-priority customers in customer queue, denoted by $L_{c_H}$ and $L_{c_L}$, respectively, is given by:

$$L_{c_H} = \frac{Area\_total\_cust\_in\_queue\_H}{sim\_time} \quad (4\text{-}50)$$

$$L_{c_L} = \frac{Area\_total\_cust\_in\_queue\_L}{sim\_time} \quad (4\text{-}51)$$

The average number of customers in customer queue, denoted by $L_c$, is given by:

$$L_c = \frac{Area\_total\_cust\_in\_queue}{sim\_time} \quad (4\text{-}52)$$

Third, the average number of high-priority and low-priority customers in block queue, denoted by $L_{b_H}$ and $L_{b_L}$, respectively, is given by:

$$L_{b_H} = \frac{Area\_total\_cust\_in\_block\_H}{sim\_time} \quad (4\text{-}53)$$

$$L_{b_L} = \frac{Area\_total\_cust\_in\_block\_L}{sim\_time} \quad (4\text{-}54)$$

The average number of customers in block queue, denoted by $L_b$, is given by:

$$L_b = \frac{Area\_total\_cust\_in\_block}{sim\_time} \quad (4\text{-}55)$$

Fourth, the blocking probability of high-priority and low-priority customers in the system, denoted by $P_{b_H}$ and $P_{b_L}$, respectively, is given by:

$$P_{b_H} = \frac{total\_rejected\_cust\_H}{total\_num\_of\_arrival\_H} \quad (4\text{-}56)$$

$$P_{b_L} = \frac{total\_rejected\_cust\_L}{total\_num\_of\_arrival\_L} \quad (4\text{-}57)$$

The blocking probability of the system, denoted by $P_b$, is given by:

$$P_b = \frac{total\_rejected\_cust}{total\_num\_of\_arrival} \tag{4-58}$$

Fifth, the impatient probability of high-priority and low-priority customers in the system, denoted by $P_{im_H}$ and $P_{im_L}$, respectively, is given by:

$$P_{im_H} = \frac{total\_impat\_cust\_H}{total\_num\_of\_arrival\_H - total\_rejected\_cust\_H} \tag{4-59}$$

$$P_{im_L} = \frac{total\_impat\_cust\_L}{total\_num\_of\_arrival\_L - total\_rejected\_cust\_L} \tag{4-60}$$

The impatient probability of the system, denoted by $P_{im}$, is given by:

$$P_{im} = \frac{total\_impat\_cust}{total\_num\_of\_arrival - total\_rejected\_cust} \tag{4-61}$$

Sixth, the throughput of high-priority and low-priority customers in the system, denoted by $T_{h_H}$ and $T_{h_L}$, respectively, is given by:

$$T_{h_H} = \frac{total\_served\_cust\_H}{sim\_time} \tag{4-62}$$

$$T_{h_L} = \frac{total\_served\_cust\_L}{sim\_time} \tag{4-63}$$

The throughput of the system, denoted by $T_h$, is given by:

$$T_h = \frac{total\_served\_cust}{sim\_time} \tag{4-64}$$

Seventh, the average waiting time of the high-priority and low-priority customers in the customer queue, denoted by $W_{c_H}$ and $W_{c_L}$, respectively, is given by:

$$W_{c_H} = \frac{total\_queue\_time\_H + total\_impat\_time\_H}{total\_served\_cust\_H + total\_impat\_cust\_H} \tag{4-65}$$

$$W_{c_L} = \frac{total\_queue\_time\_L + total\_impat\_time\_L}{total\_served\_cust\_L + total\_impat\_cust\_L} \tag{4-66}$$

The average waiting time in the customer queue, denoted by $W_c$, is given by:

$$W_c = \frac{total\_queue\_time + total\_impat\_time}{total\_served\_cust + total\_impat\_cust} \quad (4\text{-}67)$$

Eighth, the average waiting time of the high-priority and low-priority customers in the block queue, denoted by $W_{b_H}$ and $W_{b_L}$, respectively, is given by:

$$W_{b_H} = \frac{total\_block\_time\_H}{total\_served\_cust\_H} \quad (4\text{-}68)$$

$$W_{b_L} = \frac{total\_block\_time\_L}{total\_served\_cust\_L} \quad (4\text{-}69)$$

The average waiting time in the block queue, denoted by $W_b$, is given by:

$$W_b = \frac{total\_block\_time}{total\_served\_cust} \quad (4\text{-}70)$$

Ninth, the average waiting time of the high-priority and low-priority customers in the system, denoted by $W_H$ and $W_L$, respectively, is given by:

$$W_H = W_{c_H} + W_{b_H} \quad (4\text{-}71)$$

$$W_L = W_{c_L} + W_{b_L} \quad (4\text{-}72)$$

The average waiting time in the block queue, denoted by $W$ is given by:

$$W = W_H + W_L \quad (4\text{-}73)$$

Finally, the average number of high-priority and low-priority blocks participating in the consensus process per unit of time, denoted by $B_{n_H}$ and $B_{n_L}$, respectively, is given by:

$$B_{n_H} = \frac{Area\_total\_num\_of\_block\_H}{sim\_time} \quad (4\text{-}74)$$

$$B_{n_L} = \frac{Area\_total\_num\_of\_block\_L}{sim\_time} \quad (4\text{-}75)$$

The average number of blocks participating in the consensus process per unit of time within a block, denoted by $B_n$, is given by:

$$B_n = \frac{Area\_total\_num\_of\_block}{sim\_time} \quad (4\text{-}76)$$

# 5. Numerical Results

In this chapter, we present the numerical results derived from both the analytical model and the simulation model, covering four distinct blockchain queuing scenarios. Each scenario has two queues, which are customer queue and the block queue, operating under varying combinations of customer class structures and impatience behavior.

The first scenario models a single-class customer system without impatience, serving as a baseline configuration with First-Come-First-Served (FCFS) discipline and no abandonment. The second scenario introduces two classes of customers with non-preemptive priority, distinguishing high-priority and low-priority customers in both queuing and service procedures. In the third scenario, we revisit the single-class setting while incorporating impatience behavior, where customers may abandon the system after a random impatience threshold. Finally, the fourth scenario combines both priority and impatience, modeling a two-class customer system with distinct abandonment rates and non-preemptive priority rules.

Across all scenarios, the blockchain service is influenced by ON/OFF operational states and employs a partial batch policy during block generation. Performance metrics such as throughput, blocking probability, waiting time, and impatient rate (where applicable) are computed and compared under varying parameter configurations. These results provide a comprehensive view of how priority, impatience, and system reliability jointly impact the overall performance of blockchain-based queuing systems.

## 5.1. Scenario 1: Single-Class Customer without Impatience

The default values are as provided as below: $\lambda = 20,\ \mu_1 = 20,\ \mu_2 = 20,\ \alpha = \beta = 15$. The maximum capacity of the system is $N = 20$, and the maximum block size is $b = 5$.

### 5.1.1. Block size

Figure 5-1 to Figure 5-6 show the relationship between various performance metrics and the block size $b$. Both simulation results and analytical results are shown for comparison.

Figure 5-1 illustrates the impact of the block size $b$ on the average waiting time in the customer queue ($W_c$). As $b$ increases, $W_c$ decreases significantly, particularly

at smaller block size. This is because larger blocks allow more customers to be served per service cycle, thereby reducing the time customers spend waiting in the queue. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-2 illustrates the impact of the block size $b$ on the average waiting time in the block queue ($W_b$). As $b$ increases, $W_b$ remains nearly constant. This indicates that the time each block spends in the consensus queue is determined by the consensus rate and system state transition rate, and is independent of block size. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-3 illustrates the impact of the block size $b$ on the average waiting time in the system ($W$). As $b$ increases, $W$ decreases significantly, particularly at smaller block size. This is because larger blocks allow more customers to be served per service cycle, thereby reducing the time customers spend waiting in the queue. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-4 illustrates the impact of the block size $b$ on the average number of customers in the block queue ($L_b$). As $b$ increases, $L_b$ initially grows and then stabilizes around constant value. This indicates that although larger blocks permit more customers per batch, the average block occupancy tends to saturate when the customer arrival rate is equal to the effective service rate of the customer queue, where the effective service rate of the customer queue is dependent on the block generation rate, the system transition rate, and the average block size. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-5 illustrates the impact of the block size $b$ on the blocking probability ($P_b$). As $b$ increases, $P_b$ steadily decreases. This is because larger blocks allow more customers to be served per block generation cycle, thereby reducing the chance of the customer queue reaching its capacity limit. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-6 illustrates the impact of the block size $b$ on the system throughput ($T_h$). As $b$ increases, $T_h$ rises rapidly at first and then gradually saturates. This is because larger blocks enable more customers to be processed per consensus cycle, but the throughput eventually approaches a limit determined by the customer arrival rate, which is less than the system processing capacity. The system processing capacity is decided by the maximum block size, the block generation rate, consensus rate, and system state transition rate. Lastly, the analytical results are in good agreement with the simulation results.

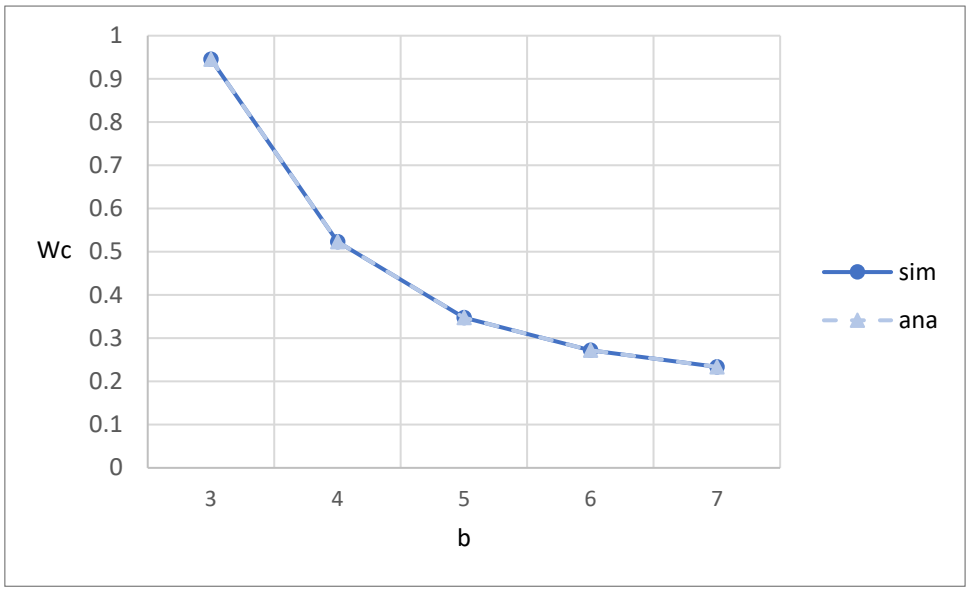


Figure 5-1: Effect of block size on average waiting time in the customer queue

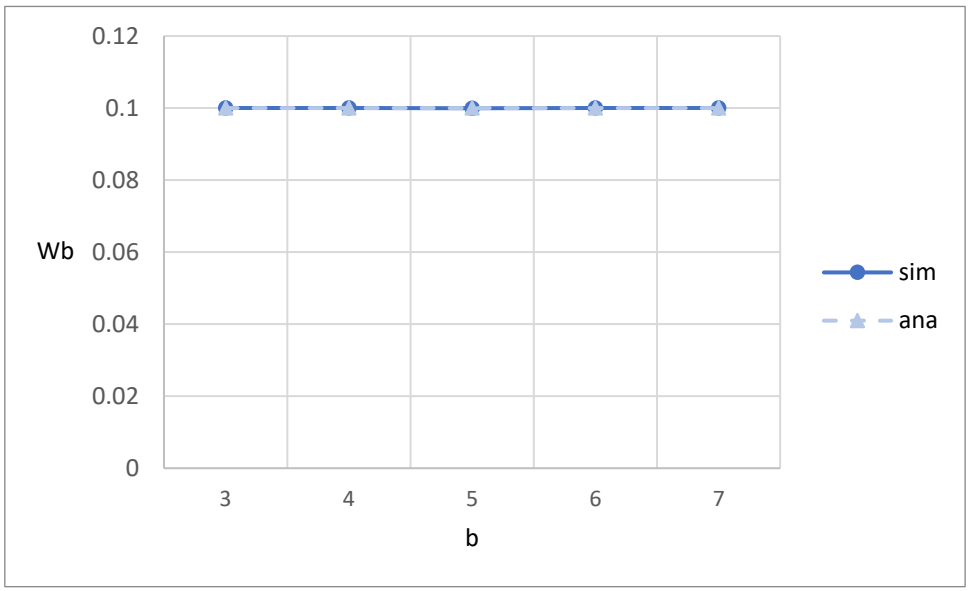


Figure 5-2: Effect of block size on average waiting time in the block queue

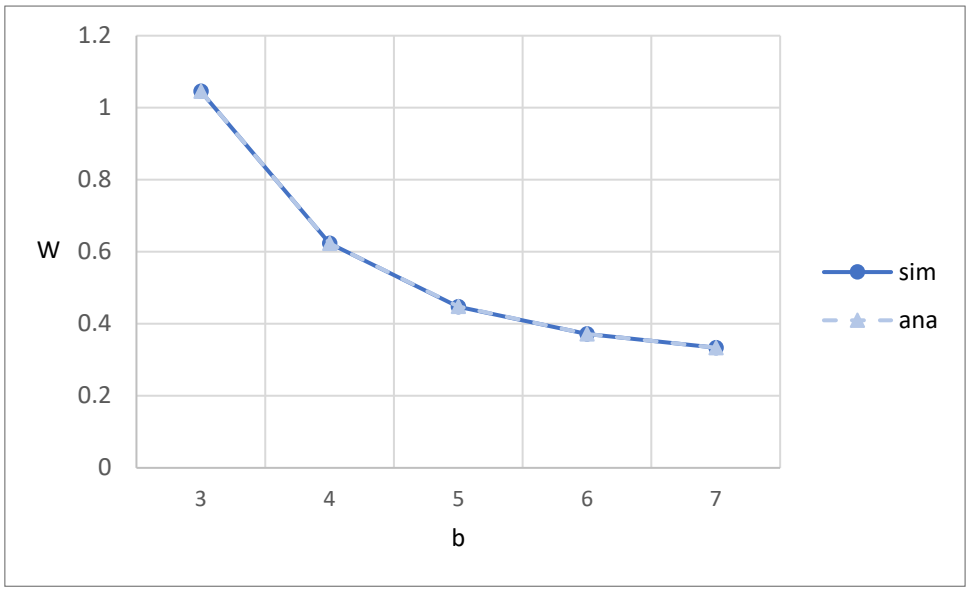


Figure 5-3: Effect of block size on average waiting time in the system

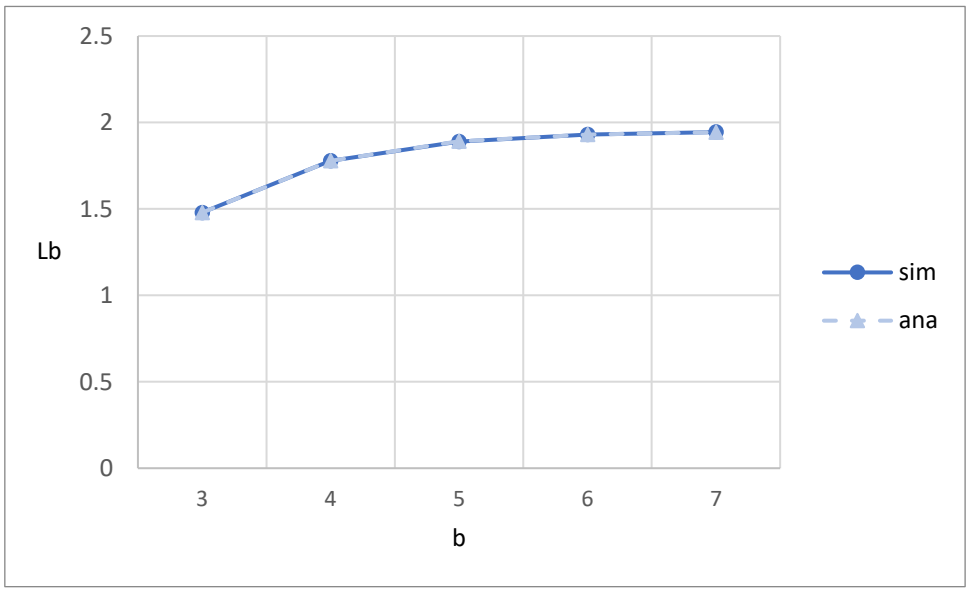


Figure 5-4: Effect of block size on average number of customers in block queue

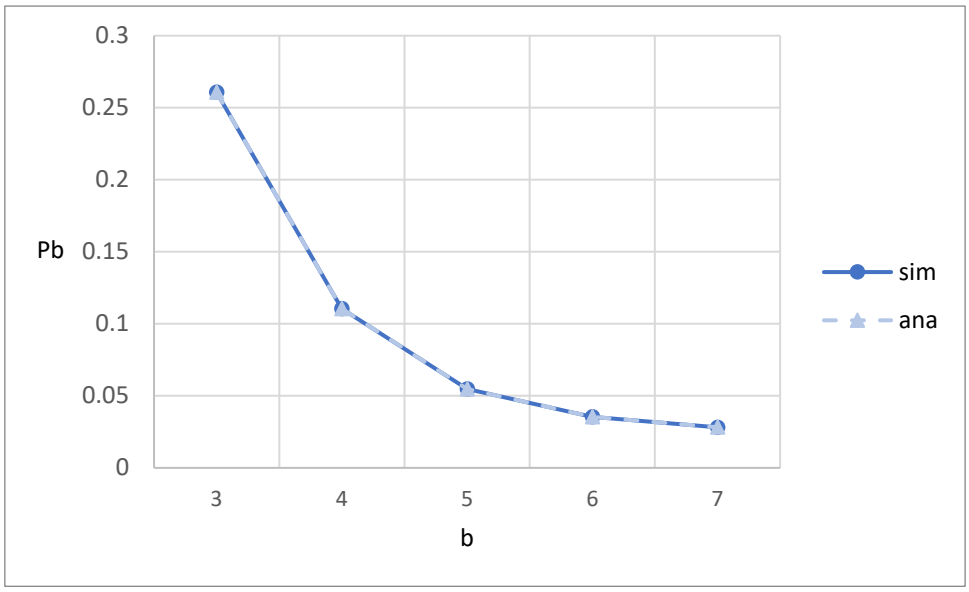


Figure 5-5: Effect of block size on blocking probability

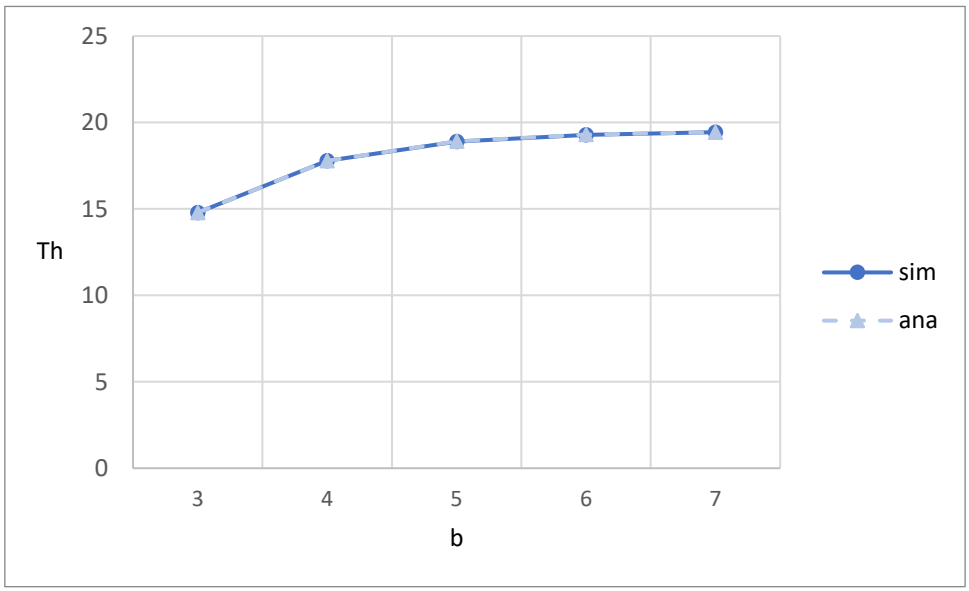


Figure 5-6: Effect of block size on system throughput

### 5.1.2. Arrival rate

Figure 5-7 to Figure 5-12 show the relationship between various performance metrics and the arrival rate $\lambda$. Both simulation results and analytical results are shown for comparison.

Figure 5-7 illustrates the impact of the arrival rate $\lambda$ on the average waiting time in the customer queue ($W_c$). As $\lambda$ increases, $W_c$ increases steadily. This is because higher arrival rate leads to more customers entering the system, causing increased congestion and longer queuing delays before customers can be batched into blocks. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-8 illustrates the impact of the arrival rate $\lambda$ on the average waiting time in the block queue ($W_b$). As $\lambda$ increases, $W_b$ remains nearly constant. This indicates that the time each block spends in the consensus queue is determined by the consensus rate and system transition rate, and is independent of the arrival rate. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-9 illustrates the impact of the arrival rate $\lambda$ on the average waiting time in the system ($W$). As $\lambda$ increases, $W$ increases steadily. This is because higher arrival rate leads to more customers entering the system, causing increased congestion and longer queuing delays before customers depart from the system. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-10 illustrates the impact of the arrival rate $\lambda$ on the average number of customers in the block queue ($L_b$). As $\lambda$ increases, $L_b$ increases steadily. This is because a higher arrival rate leads to more frequent block formation and more customers being accumulated in each block, increasing the average number of customers waiting for consensus. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-11 illustrates the impact of the arrival rate $\lambda$ on the blocking probability ($P_b$). As $\lambda$ increases, $P_b$ rises sharply, specially beyond $\lambda = 20$. This is because a higher arrival rate leads to more customers being accumulated in the customer queue, increasing the chance that incoming customers are blocked due to limited queue capacity. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-12 illustrates the impact of the arrival rate $\lambda$ on the system throughput ($T_h$). As $\lambda$ increases, $T_h$ rises rapidly at first and then gradually saturates. This is because higher arrival rates supply more customers into the system, but throughput

eventually becomes limited by the system service capacity, which is decided by the maximum block size, the block generation rate, the consensus rate, and system transition rate. Lastly, the analytical results are in good agreement with the simulation results.

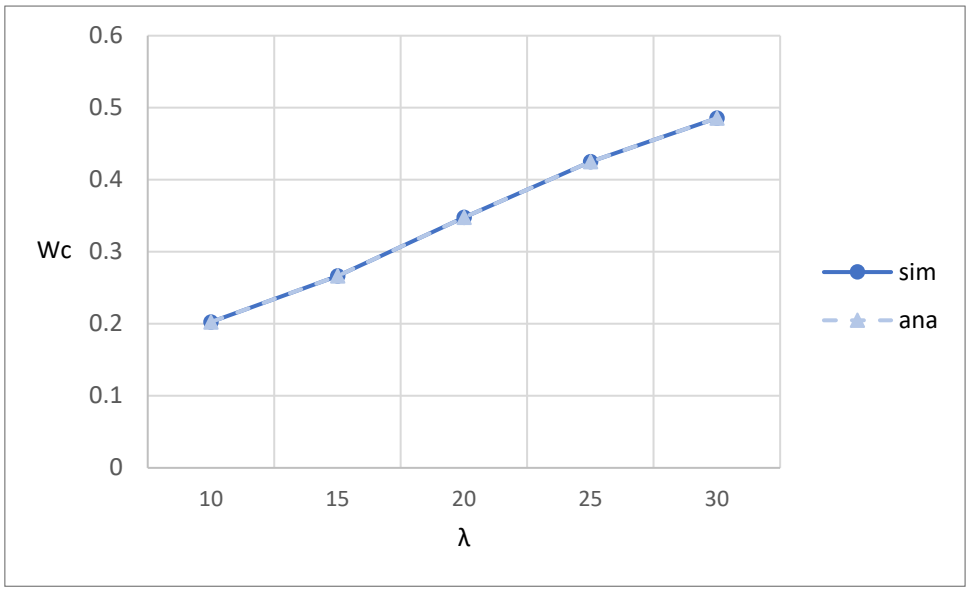


Figure 5-7: Effect of arrival rate on average waiting time in the customer queue

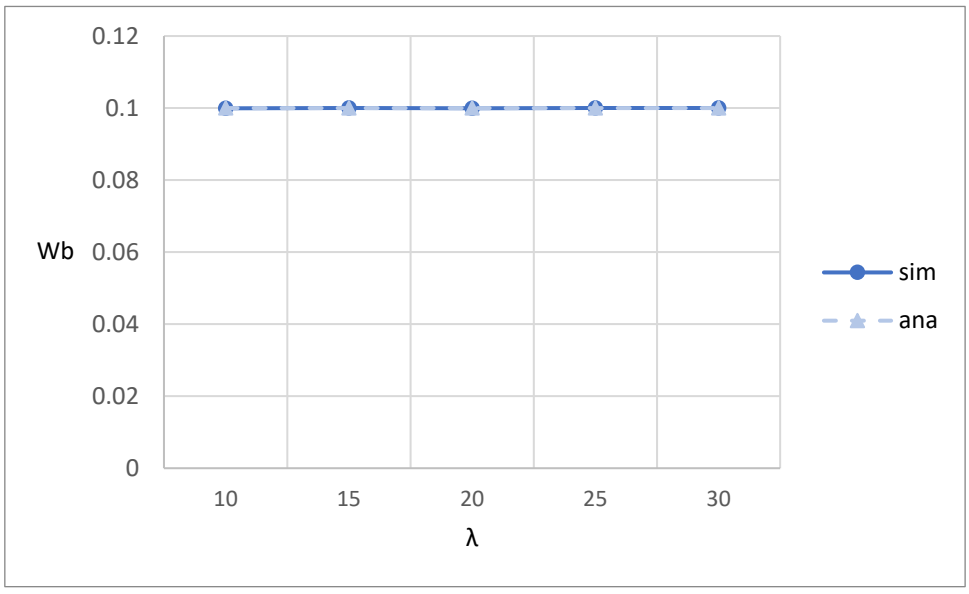


Figure 5-8: Effect of arrival rate on average waiting time in the block queue

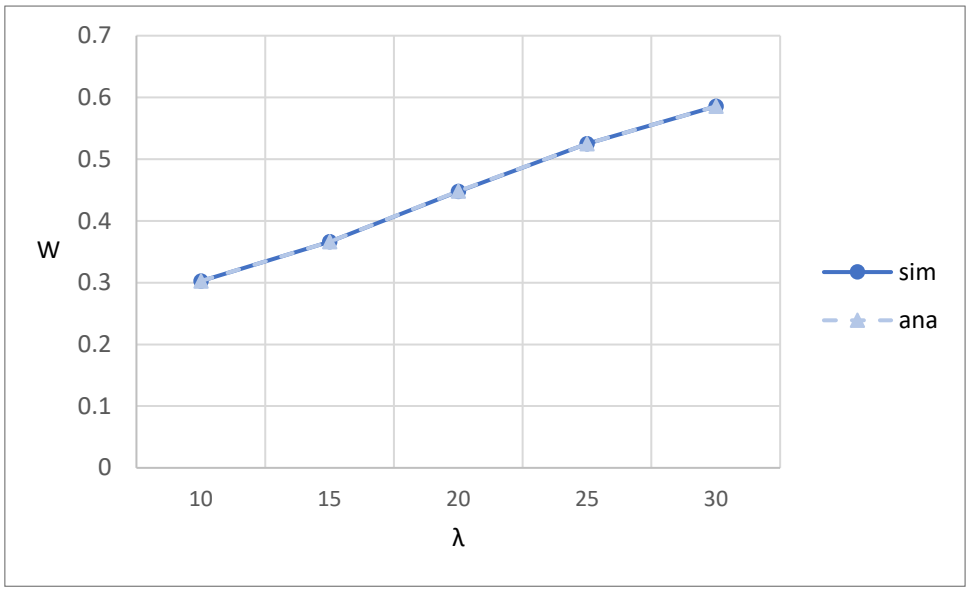


Figure 5-9: Effect of arrival rate on average waiting time in the system

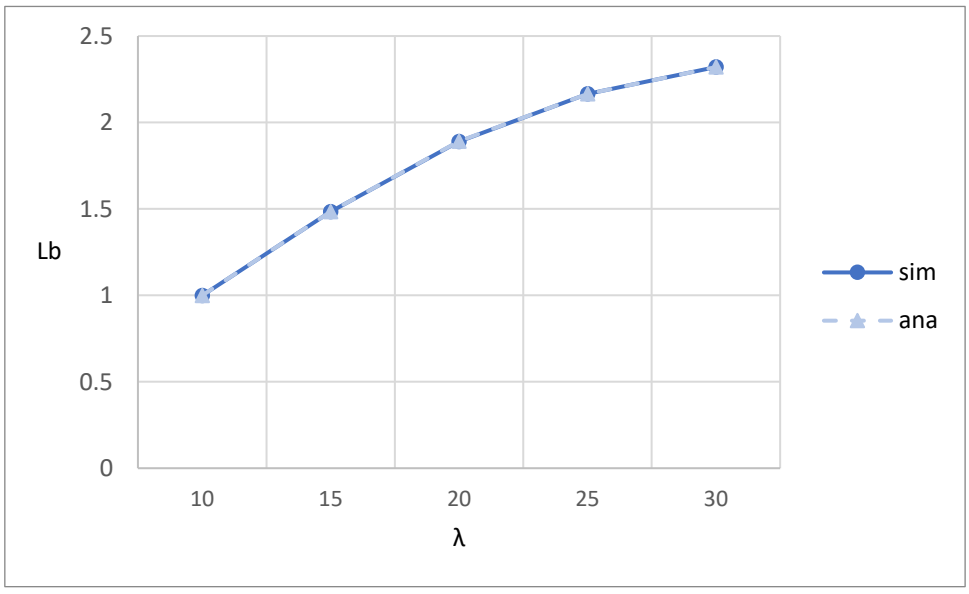


Figure 5-10: Effect of arrival rate on average number of customers in block queue

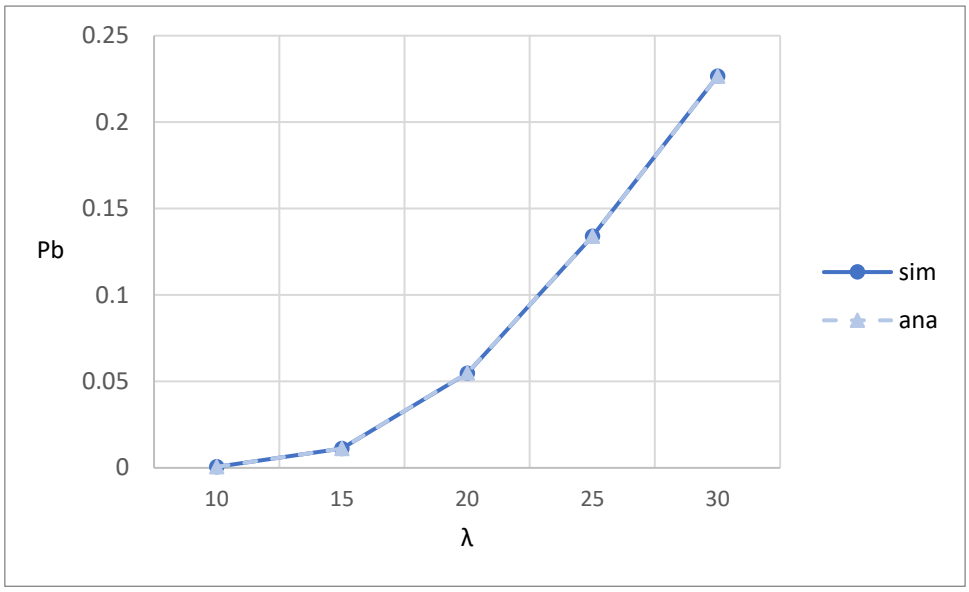


Figure 5-11: Effect of arrival rate on blocking probability

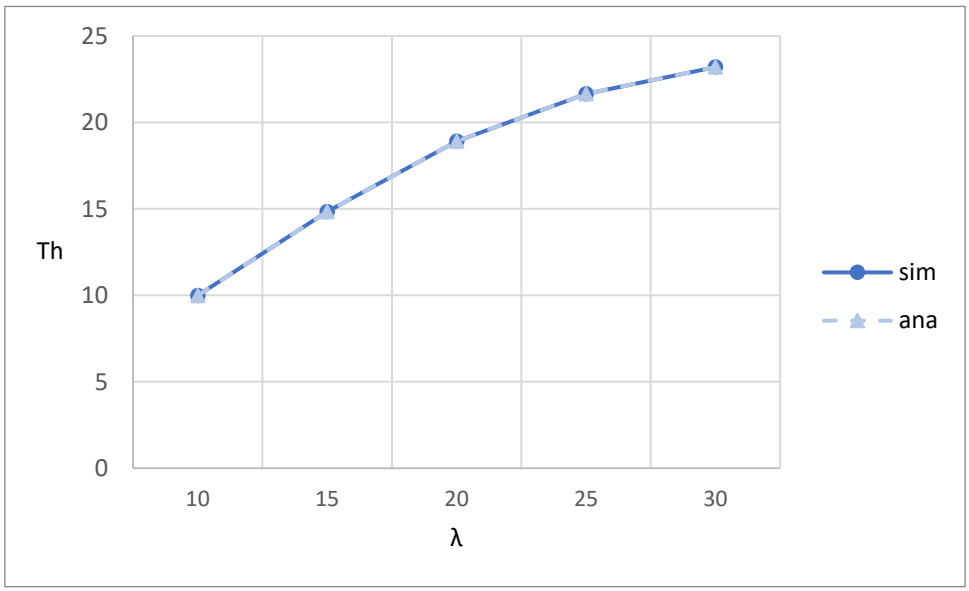


Figure 5-12: Effect of arrival rate on throughput

### 5.1.3. Block generation rate

Figure 5-13 to Figure 5-18 show the relationship between various performance metrics and the block generation rate $\mu_1$. Both simulation results and analytical results are shown for comparison.

Figure 5-13 illustrates the impact of the block generation rate $\mu_1$ on the average waiting time in the customer queue ($W_c$). As $\mu_1$ increases, $W_c$ decreases significantly. This is because higher block generation rates allow customers to be grouped and processed more frequently, which reduces congestion in the customer queue. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-14 illustrates the impact of the block generation rate $\mu_1$ on the average waiting time in the block queue ($W_b$). As $\mu_1$ increases, $W_b$ remains nearly constant. This indicates that the time each block spends in the consensus queue is determined by the consensus rate and system transition rate, and is independent of the block generation rate. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-15 illustrates the impact of the block generation rate $\mu_1$ on the average waiting time in the system ($W$). As $\mu_1$ increases, $W$ decreases steadily. This is because higher block generation rates allow customers to be grouped and processed more frequently, which reduces congestion in the customer queue. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-16 illustrates the impact of the block generation rate $\mu_1$ on the average number of customers in the block queue ($L_b$). As $\mu_1$ increases, $L_b$ initially grows and then stabilizes around constant value. This indicates that although more frequent block generation can expedite customers service, the average block occupancy tends to saturate because of the arrival rate and the effective service rate reached a state of equilibrium. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-17 illustrates the impact of the block generation rate $\mu_1$ on the blocking probability ($P_b$). As $\mu_1$ increases, $P_b$ decreases rapidly. This is because higher block generation rates allow customers to be served more frequently, reducing the possibility that the customer queue reaches its capacity and causes incoming arrivals to be blocked. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-18 illustrates the impact of the block generation rate $\mu_1$ on the system throughput ($T_h$). As $\mu_1$ increases, $T_h$ rises rapidly at first and then gradually saturates.

This is because higher block generation rates enable more customers to be processed per consensus cycle, but the throughput eventually approaches a limit determined by the customer arrival rate, which is less than the system processing capacity. Lastly, the analytical results are in good agreement with the simulation results.

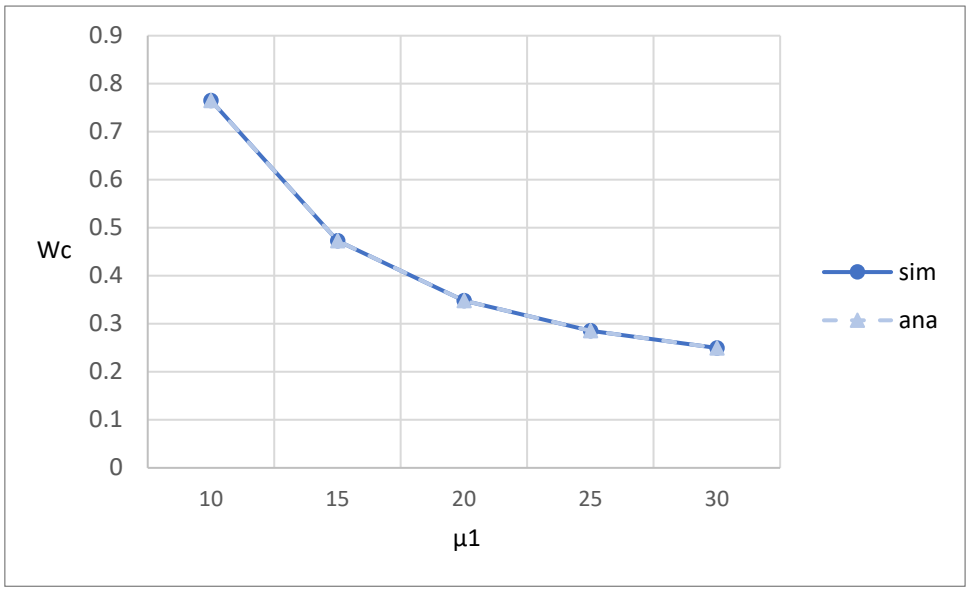


Figure 5-13: Effect of block generation rate on average waiting time in the customer queue

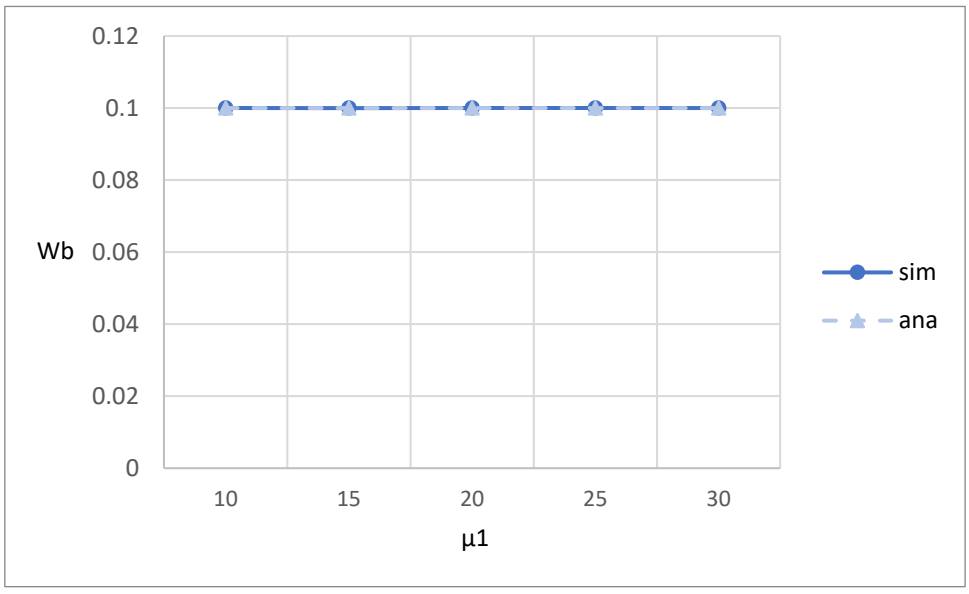


Figure 5-14: Effect of block generation rate on average waiting time in the block queue

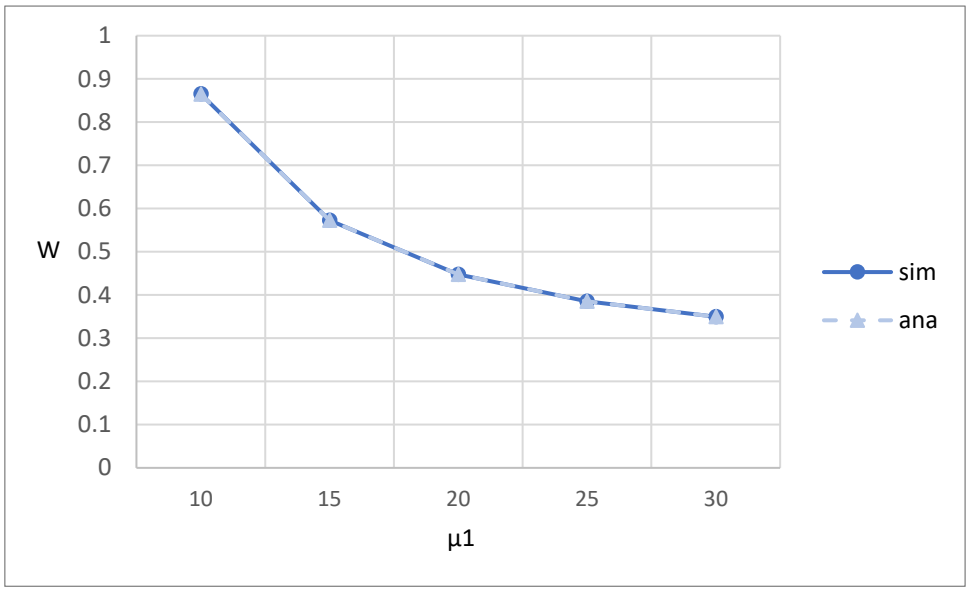


Figure 5-15: Effect of block generation rate on average waiting time in the system

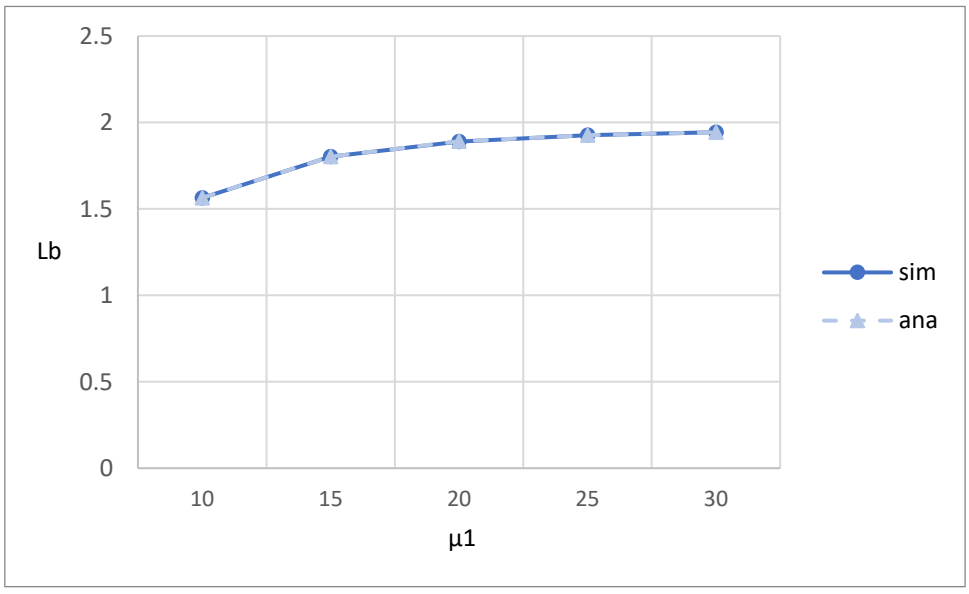


Figure 5-16: Effect of block generation rate on average number of customers in block queue

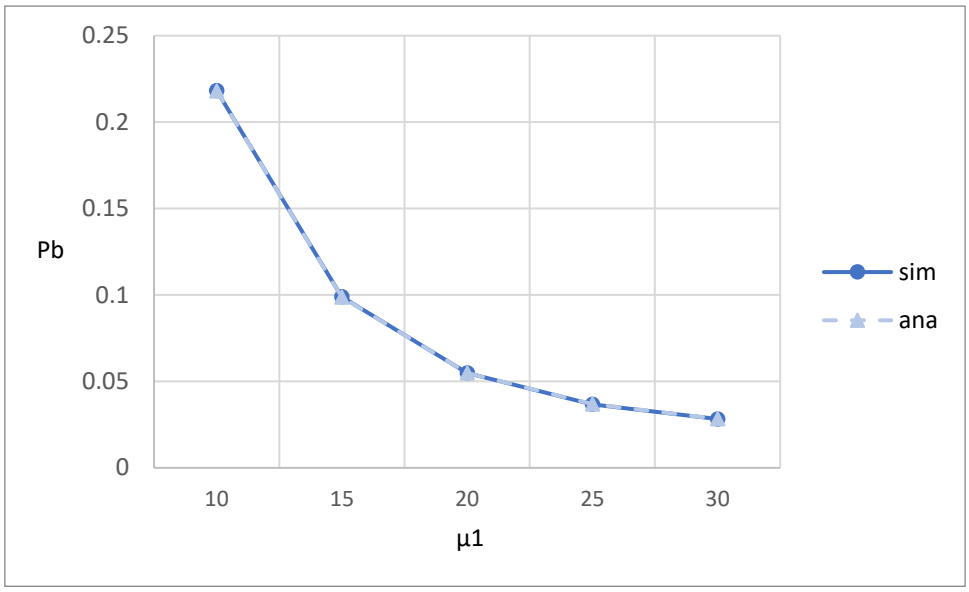


Figure 5-17: Effect of block generation rate on blocking probability

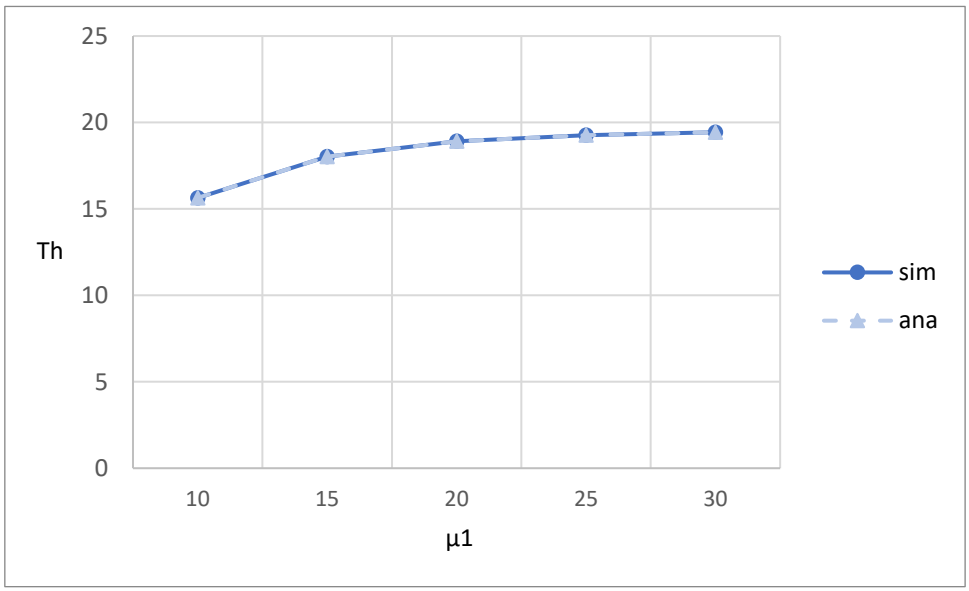


Figure 5-18: Effect of block generation rate on system throughput

### 5.1.4. Consensus rate

Figure 5-19 to Figure 5-24 show the relationship between various performance metrics and the consensus rate $\mu_2$. Both simulation results and analytical results are shown for comparison.

Figure 5-19 illustrates the impact of the consensus rate $\mu_2$ on the average waiting time in the customer queue ($W_c$). As $\mu_2$ increases, $W_c$ decrease steadily. This is because a higher consensus rate shortens the time block spend in the consensus queue, enabling faster turnover and more frequent admission of waiting customers from the customer queue. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-20 illustrates the impact of the consensus rate $\mu_2$ on the average waiting time in the block queue ($W_b$). As $\mu_2$ increases, $W_b$ decrease significantly. This is because a faster consensus rate allows blocks to be processed more quickly, thereby reducing the time that blocks spend waiting in the queue. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-21 illustrates the impact of the consensus rate $\mu_2$ on the average waiting time in the system ($W$). As $\mu_2$ increases, $W$ decrease significantly. This is because faster consensus processing reduces delays in the block queue, which in turn accelerates the overall service flow and shortens the total time customers spend in the system. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-22 illustrates the impact of the consensus rate $\mu_2$ on the average number of customers in the block queue ($L_b$). As $\mu_2$ increases, $L_b$ decrease significantly. This is because a higher consensus rate enables faster processing of blocks, which reduces queue accumulation and lowers the $L_b$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-23 illustrates the impact of the consensus rate $\mu_2$ on the blocking probability ($P_b$). As $\mu_2$ increases, $P_b$ decrease rapidly. This is because faster consensus rate enables faster processing of blocks, and lowers the probability of reaching the queue capacity that triggers blocking. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-24 illustrates the impact of the consensus rate $\mu_2$ on the system throughput ($T_h$). As $\mu_2$ increases, $T_h$ increase rapidly at first and then gradually saturates. This is because higher consensus rate allows blocks to be processed more

efficiently, but the throughput eventually approaches a limit determined by the customer arrival rate, which is less than the system processing capacity. Lastly, the analytical results are in good agreement with the simulation results.

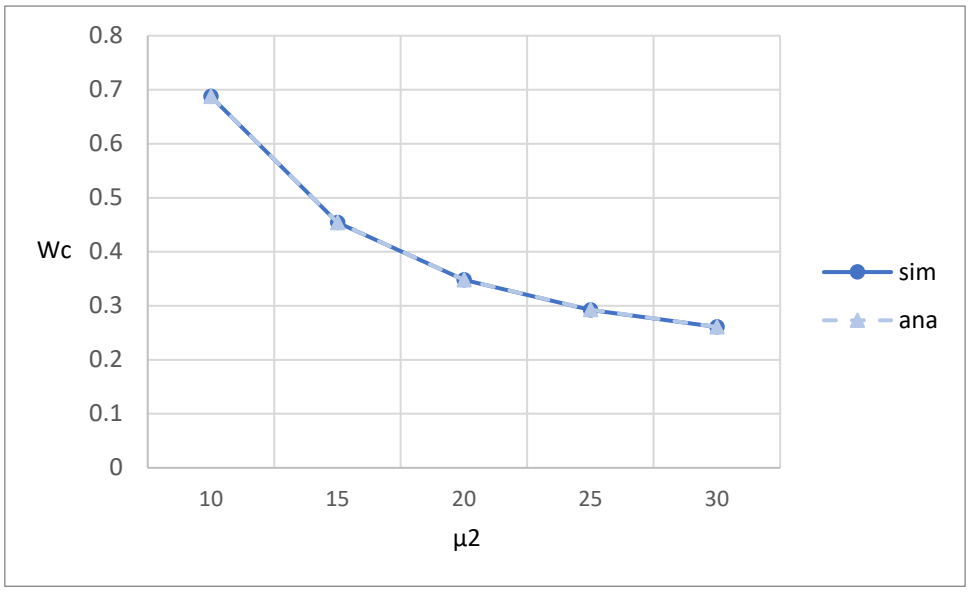


Figure 5-19: Effect of consensus rate on average waiting time in the customer queue

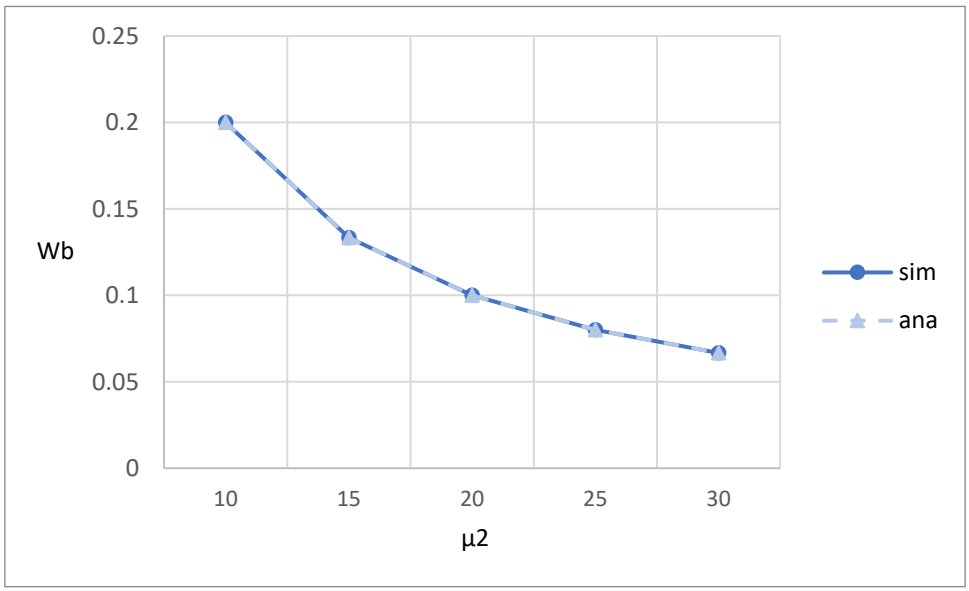


Figure 5-20: Effect of consensus rate on average waiting time in the block queue

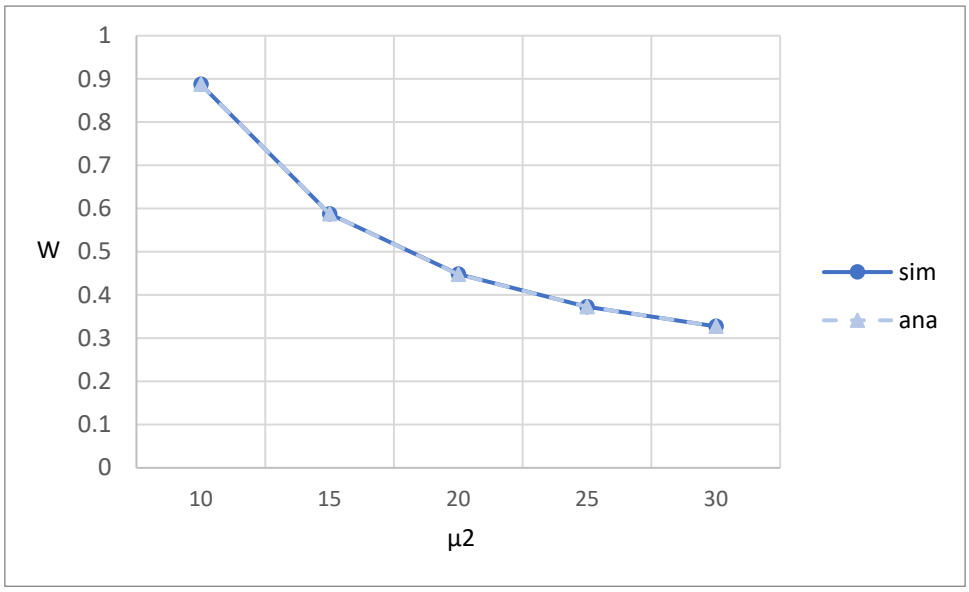


Figure 5-21: Effect of consensus rate on average waiting time in the system

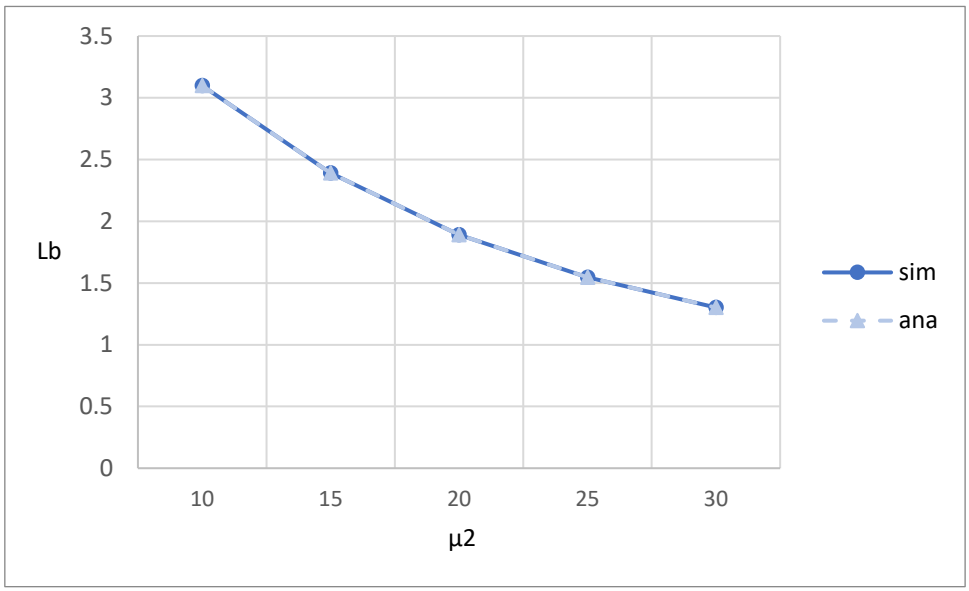


Figure 5-22: Effect of consensus rate on average number of customers in block queue

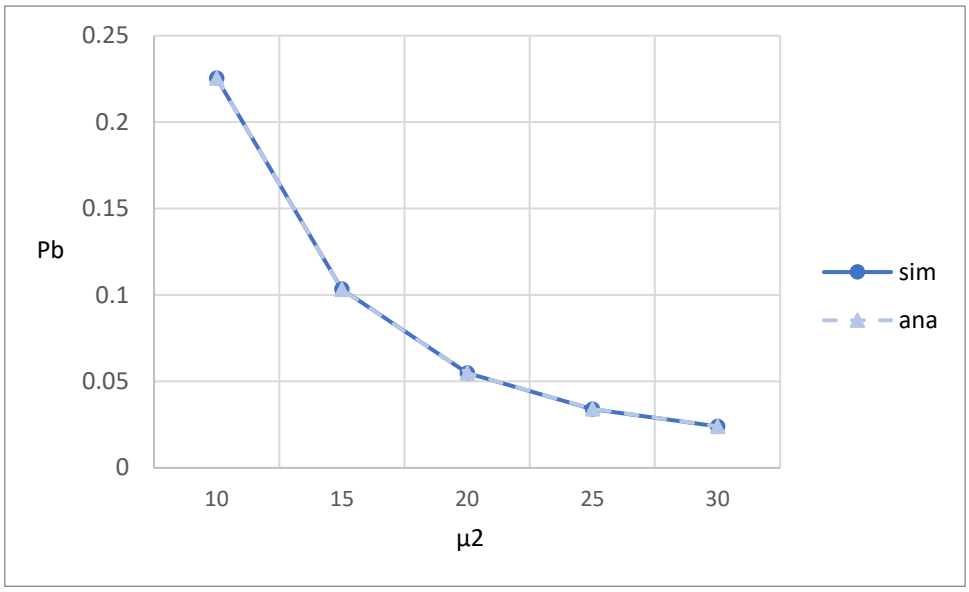


Figure 5-23: Effect of consensus rate on blocking probability

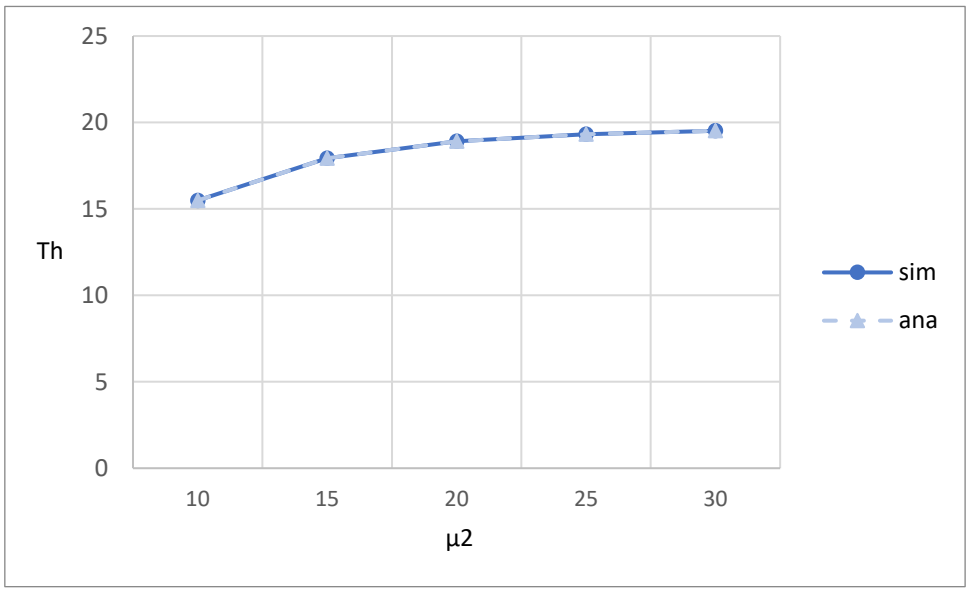


Figure 5-24: Effect of consensus rate on system throughput

### 5.1.5. Transition rate (from ON to OFF)

Figure 5-25 to Figure 5-30 show the relationship between various performance metrics and the transition rate from ON to OFF $\alpha$. Both simulation results and analytical results are shown for comparison.

Figure 5-25 illustrates the impact of the transition rate $\alpha$ on the average waiting time in the customer queue ($W_c$). As $\alpha$ increases, $W_c$ increases steadily. This is because more frequent transitions from ON to OFF reduce the availability of block generation service, causing longer queueing delays for arriving customers. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-26 illustrates the impact of the transition rate $\alpha$ on the average waiting time in the block queue ($W_b$). As $\alpha$ increases, $W_b$ increase steadily. This is because more frequent service interruptions delay consensus processing, resulting in longer waiting times for customer(s) in block queue. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-27 illustrates the impact of the transition rate $\alpha$ on the average waiting time in the system ($W$). As $\alpha$ increases, $W$ increase steadily. This is because more frequent service interruptions caused by transitions to the OFF state reduce overall availability of block generation and consensus service, leading to longer queueing delays for customers in the system. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-28 illustrates the impact of the transition rate $\alpha$ on the average number of customers in the block queue ($L_b$). As $\alpha$ increases, $L_b$ increases steadily. This is because more frequent service interruptions of block generation processing cause more customers to wait in the customer queue while forming batches from the customer queue. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-29 illustrates the impact of the transition rate $\alpha$ on the blocking probability ($P_b$). As $\alpha$ increases, $P_b$ increases steadily. This is because more frequent service interruptions reduce the system's capacity to process customers, which increases the probability that the customer queue reaches its capacity and causes incoming arrivals to be blocked. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-30 illustrates the impact of the transition rate $\alpha$ on the system throughput ($T_h$). As $\alpha$ increases, $T_h$ decreases gradually. This is because more

frequent service interruptions reduce the chance for block generation and consensus processing, thereby limiting the rate at which customers are served and ultimately lowering the overall system throughput. Lastly, the analytical results are in good agreement with the simulation results.

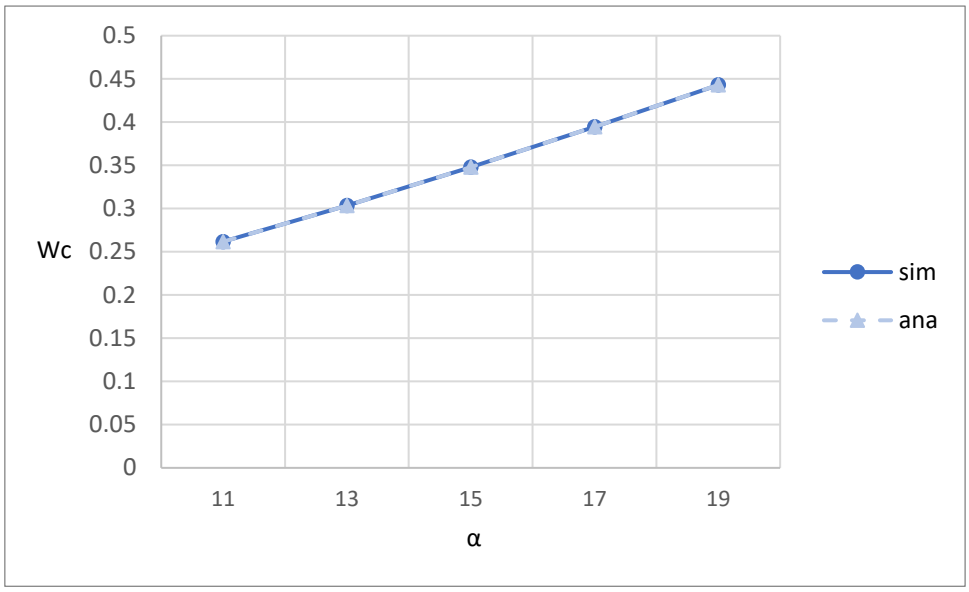


Figure 5-25: Effect of transition rate on average waiting time in the customer queue in the system

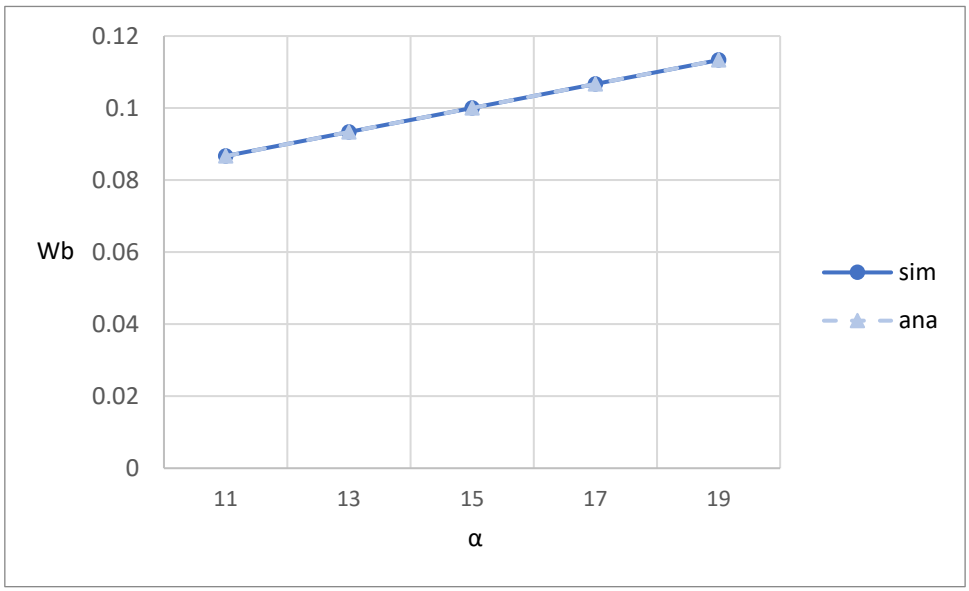


Figure 5-26: Effect of transition rate on average waiting time in the block queue

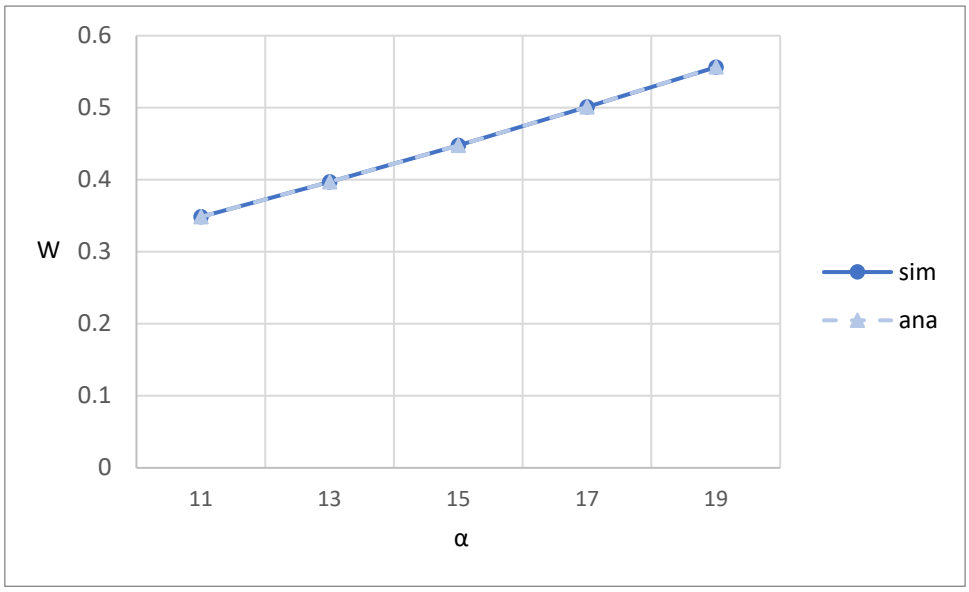


Figure 5-27: Effect of transition rate on average waiting time in the system

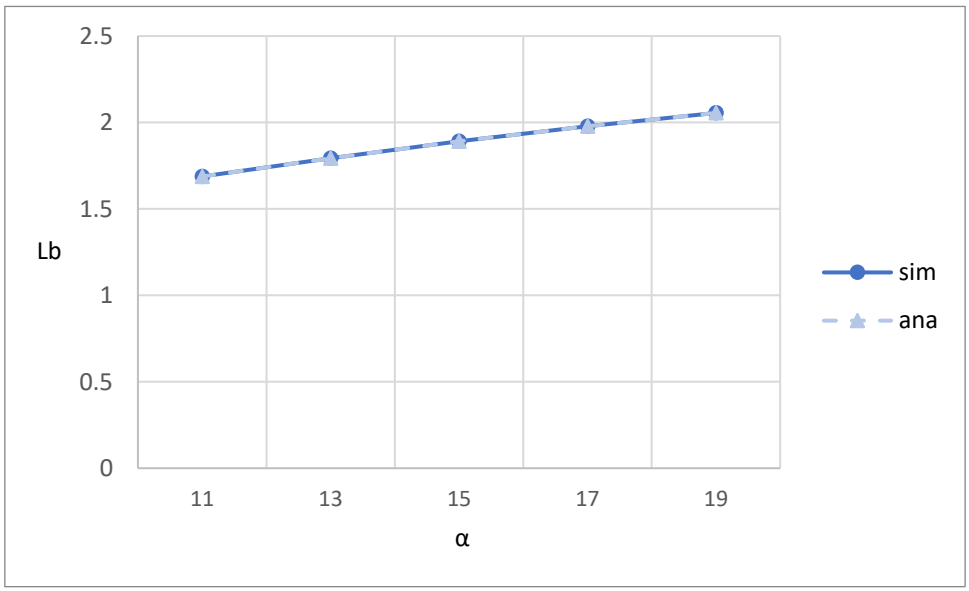


Figure 5-28: Effect of transition rate on average number of customers in block queue

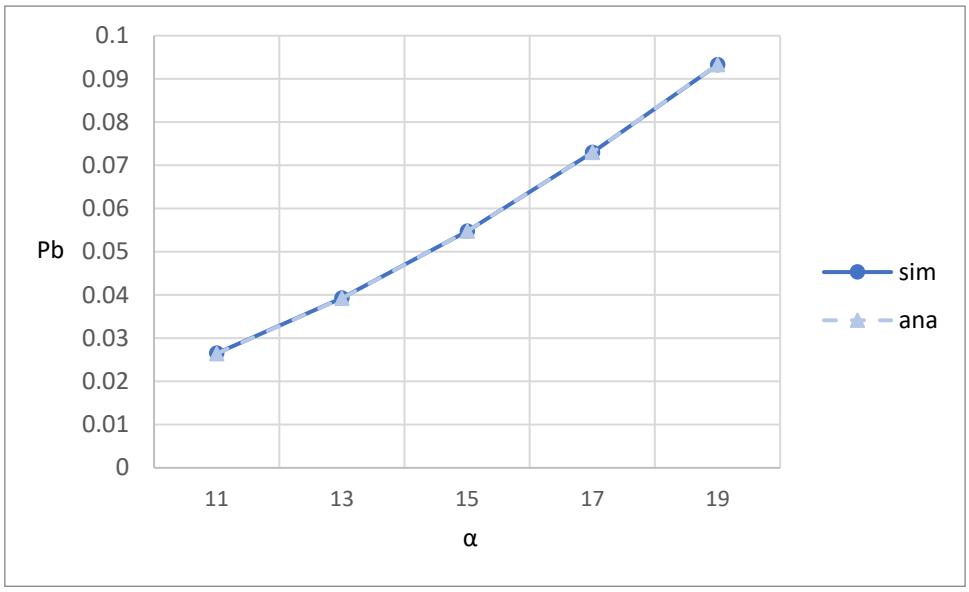


Figure 5-29: Effect of transition rate on blocking probability

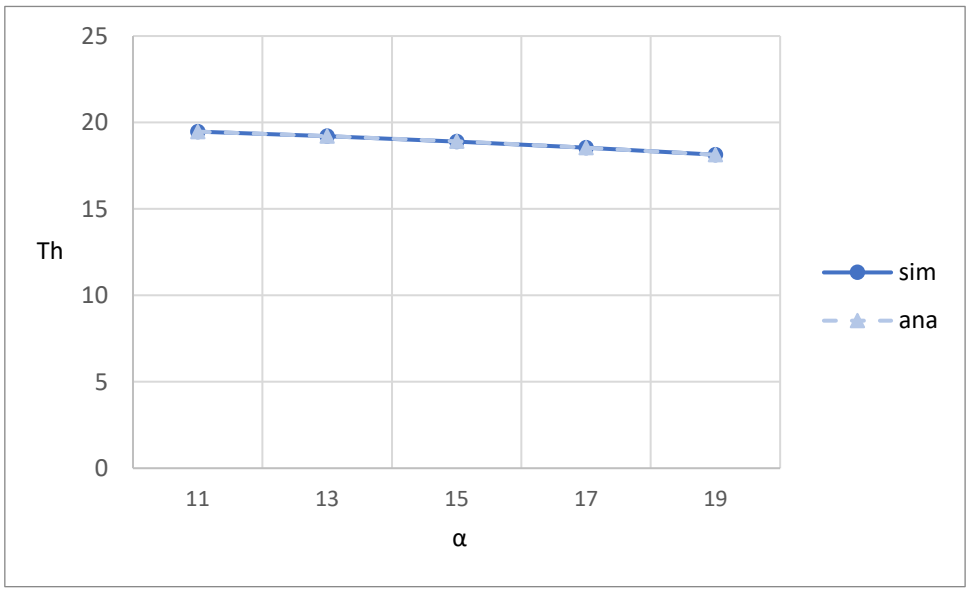


Figure 5-30: Effect of transition rate on system throughput

## 5.2. Scenario 2: Two-Class Customer without Impatience

The default values are as provided as below: $\lambda_H = 5$, $\lambda_L = 15$, $\mu_{1_H} = 20$, $\mu_{1_L} = 20$, $\mu_{2_H} = 25$, $\mu_{2_L} = 20$, $\alpha = \beta = 15$. The maximum capacity of the system is $N = 20$, and the maximum block size is $b = 5$.

### 5.2.1. Block size

Figure 5-31 to Figure 5-36 show the relationship between various performance metrics and the block size $b$. Both simulation results and analytical results are shown for comparison.

Figure 5-31 illustrates the impact of the block size $b$ on the average waiting times in the customer queue for high-priority, low-priority, and overall customers. As $b$ increases, the $W_c$ decreases. The reduction is more pronounced for $W_{c_L}$, while $W_{c_H}$ remain consistently low. This is because larger blocks allow more customers to be served per service cycle, thereby reducing the time, especially for low-priority customers who tend to experience longer delays when $b$ is small. In addition, the $W_{c_H}$ is much smaller than $W_{c_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-32 illustrates the impact of the block size $b$ on the average waiting times in the block queue for high-priority, low-priority, and overall customers. As $b$ increases, the average waiting time in the block queue remains nearly constant for all priority levels. This indicates that the time each block spends in the consensus queue is determined by the consensus rate and system state transition rate, and is independent of block size. In addition, the $W_{b_H}$ is smaller than $W_{b_L}$. This is because $\mu_{2_H}$ is larger than $\mu_{2_L}$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-33 illustrates the impact of the block size $b$ on the average waiting times in the system for high-priority, low-priority, and overall customers. As $b$ increases, the $W$ decreases. The decline is especially significant for low-priority customers, while the $W_H$ remains relatively constant. This is because larger blocks allow more customers to be served per service cycle, which benefits low-priority customers who are otherwise delayed by the non-preemptive priority mechanism. In addition, the $W_H$ is much smaller than $W_L$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue and $\mu_{2_H}$ is larger than $\mu_{2_L}$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-34 illustrates the impact of the block size $b$ on the average numbers of customers in the block queue for high-priority, low- priority, and overall customers. As $b$ increases, the average number of customers in the block queue rises gradually across all priority levels. The increase is most noticeable for $L_b$ and $L_{b_L}$, while the $L_{b_H}$ remains relatively low and stable. This indicates that although larger blocks permit more customers per batch, the average block occupancy tends to saturate when the customer arrival rate is equal to the effective service rate of the customer queue. In addition, $L_{b_H}$ is smaller than $L_{b_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue and the high-priority customers have faster consensus rate than low-priority customers in the block queue, and therefore more low-priority customers remain waiting in the customer queue before being batched. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-35 illustrates the impact of the block size $b$ on the blocking probabilities for high-priority, low- priority, and overall customers. As $b$ increases, the blocking probability decreases across all priority levels. The decline is more pronounced for $P_{b_L}$, which is initially much higher and drops significantly with increasing $b$. This is because larger blocks allow more customers to be served per block generation cycle, thereby reducing the chance of the customer queue reaching its capacity limit, especially for low-priority customers who are more likely to be blocked under limited queue capacity. In addition, $P_{b_H}$ is smaller than $P_{b_L}$. This is because the capacity limit of the customer queue for the high-priority customers is no smaller than that for low-priority customers. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-36 illustrates the impact of the block size $b$ on the system throughputs for high-priority, low-priority, and overall customers. As $b$ increases, the system throughput increases across all priority levels and then gradually saturates. Both $T_{h_H}$ and $T_{h_L}$ increase with $b$, with the growth being more significant for low-priority customers. This is because larger blocks enable more customers to be processed per consensus cycle. However, the throughput eventually approaches a limit determined by the customer arrival rate, which is less than the system processing capacity. In addition, $T_{h_H}$ is smaller than $T_{h_L}$. This is because the customer arrival rate of the high-priority customers is smaller than that of low-priority customers. Lastly, the analytical results are in good agreement with the simulation results.

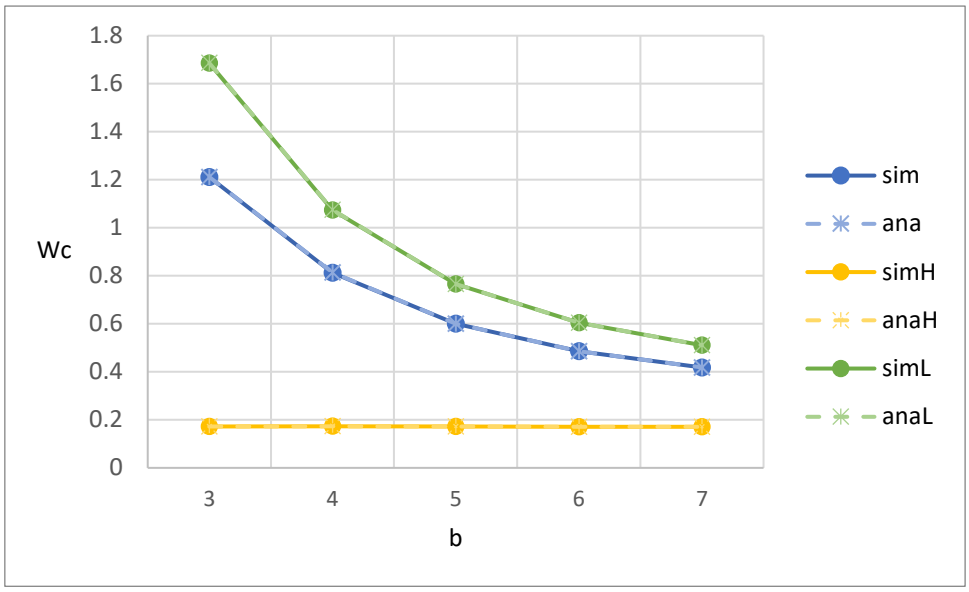


Figure 5-31: Effect of block size on average waiting time in the customer queue

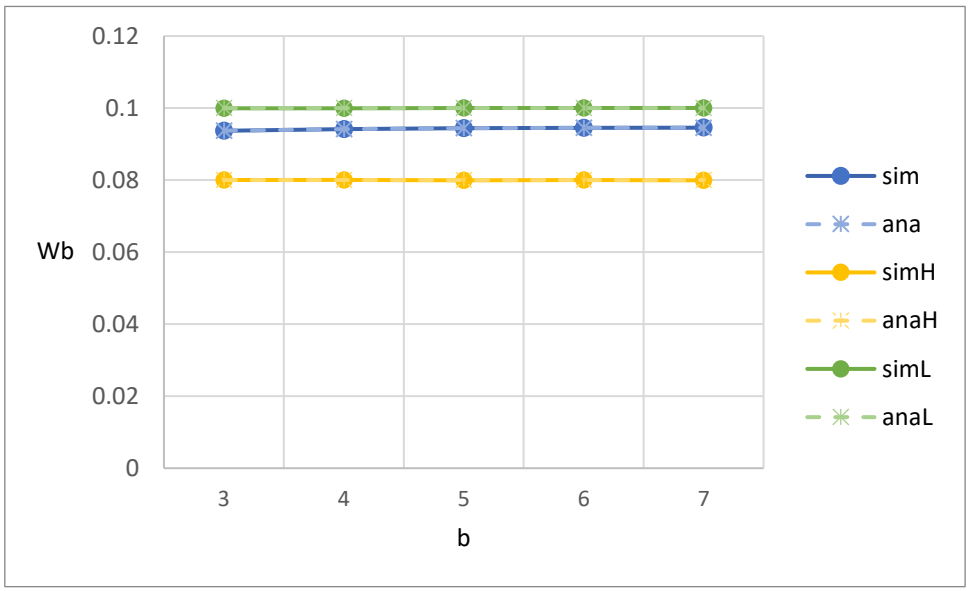


Figure 5-32: Effect of block size on average waiting time in the block queue

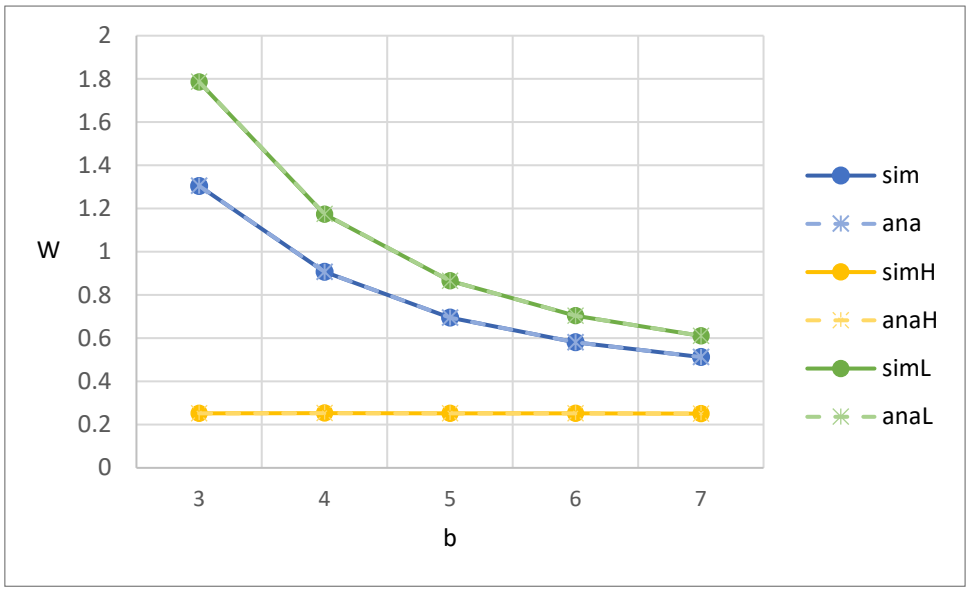


Figure 5-33: Effect of block size on average waiting time in the system

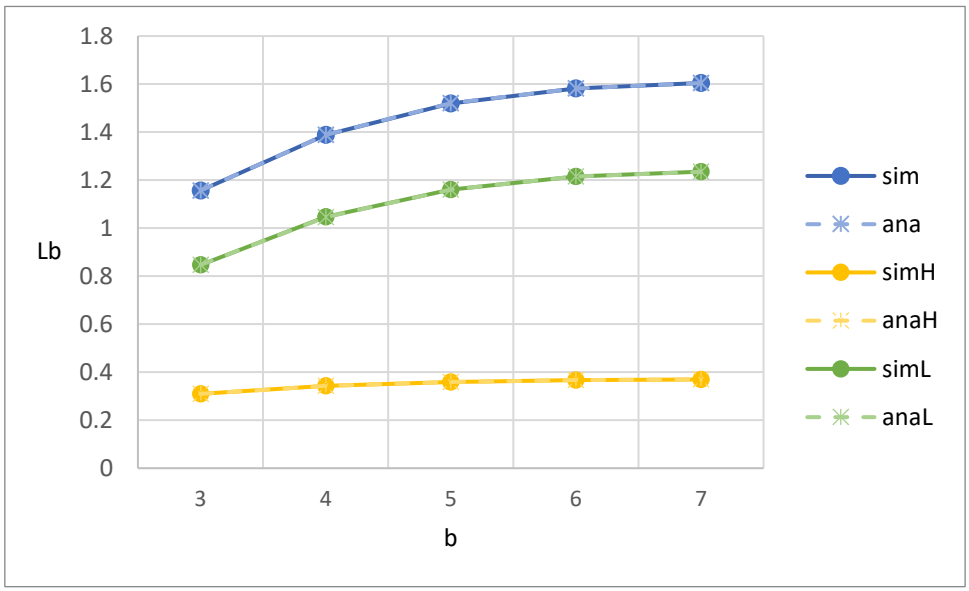


Figure 5-34: Effect of block size on average number of customers in block queue

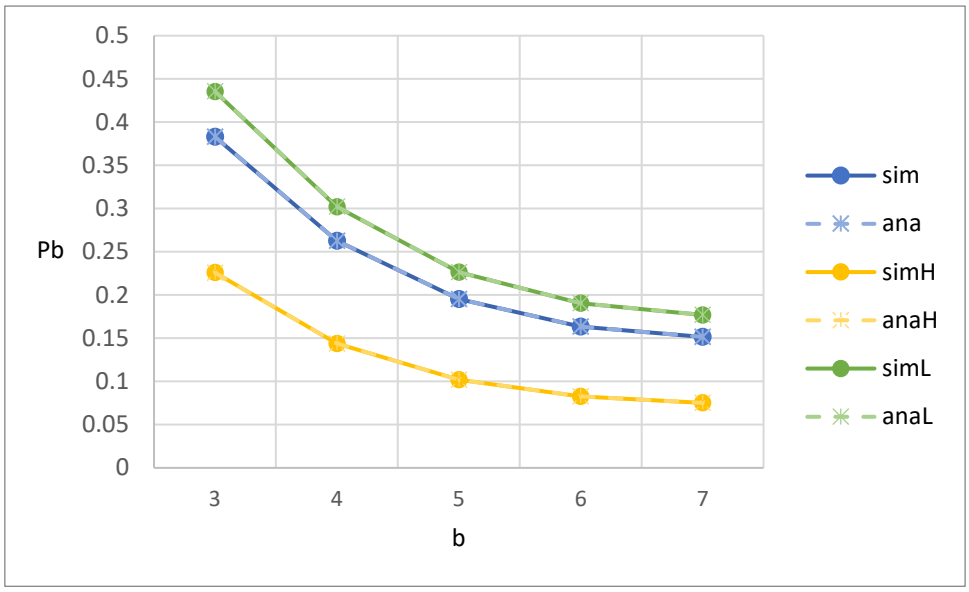


Figure 5-35: Effect of block size on blocking probability

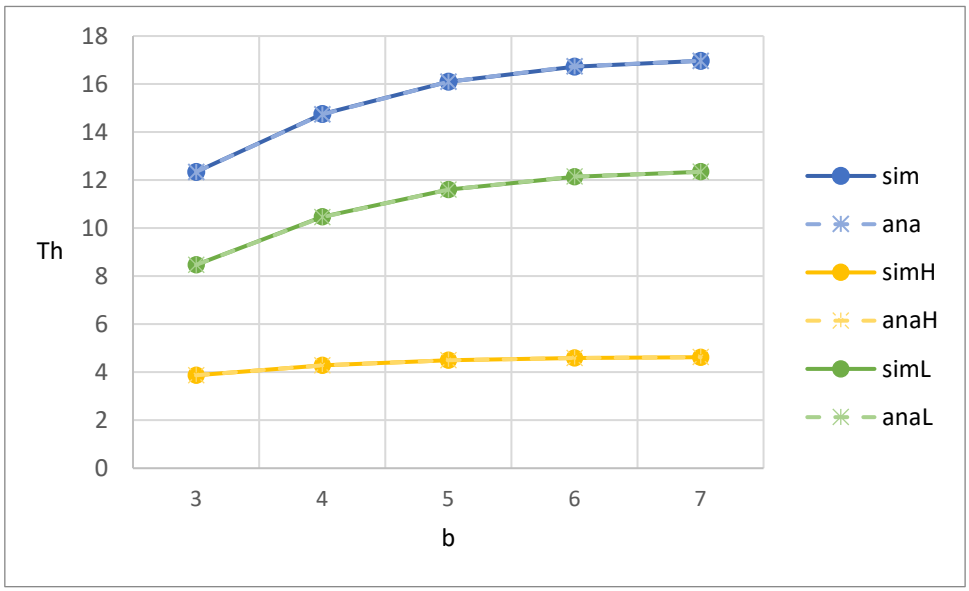


Figure 5-36: Effect of block size on system throughput

### 5.2.2. Arrival rate

Figure 5-37 to Figure 5-42 show the relationship between various performance metrics and the arrival rate of high-priority customers $\lambda_H$. Both simulation results and analytical results are shown for comparison.

Figure 5-37 illustrates the impact of the arrival rate of high-priority customers $\lambda_H$ on the average waiting times in the customer queue for high-priority, low-priority, and overall customers. As $\lambda_H$ increases, the $W_c$ increases steadily. The rise is mainly due to the significant increase in $W_{c_L}$, while $W_{c_H}$ remains nearly constant. This is because more high-priority arrivals dominate the queue under the non-preemptive priority mechanism, causing low-priority customers to wait longer in the customer queue. In addition, the $W_{c_H}$ is much shorter than $W_{c_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-38 illustrates the impact of the arrival rate of high-priority customers $\lambda_H$ on the average waiting times in the block queue for high-priority, low-priority, and overall customers. As $\lambda_H$ increases, $W_{b_H}$ and $W_{b_L}$ remains nearly constant. This indicates that the time each block spends in the consensus queue is determined by the associated consensus rate and system transition rate, and is independent of $\lambda_H$. In addition, $W_{b_H}$ is smaller than $W_{b_L}$. This is because $\mu_{2_H}$ is larger than $\mu_{2_L}$. Furthermore, as $\lambda_H$ increases, $W_b$ decreases. This is because as $\lambda_H$ increases, more high-priority blocks are formed and the consensus rate of high-priority customers is larger than that of low-priority customers. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-39 illustrates the impact of the arrival rate of high-priority customers $\lambda_H$on the average waiting times in the system for high-priority, low-priority, and overall customers. As $\lambda_H$ increases, the $W$ increases steadily. The rise is mainly due to the significant increase in $W_L$, while $W_H$ remains nearly constant. This is because more $\lambda_H$ dominate the queue under the non-preemptive priority mechanism, causing low-priority customers to spend more time in the system. In addition, $W_H$ is much smaller than $W_L$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue and $\mu_{2_H}$ is larger than $\mu_{2_L}$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-40 illustrates the impact of the arrival rate of high-priority customers $\lambda_H$ on the average numbers of customers in the block queue for high-priority, low-priority,

and overall customers. As $\lambda_H$ increases, the $L_b$ remains relatively stable, but with diverging trends across priority class. Specifically, $L_{b_H}$ increases and $L_{b_L}$ decreases. This behavior reflects the shift in queue composition under the non-preemptive priority mechanism, where more high-priority customers are admitted into the system while low-priority customers are blocked earlier or delayed at the customer queue. Thus, more high-priority blocks and less low-priority blocks are formed. In addition, $L_{b_H}$ is smaller than $L_{b_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue and the high-priority customers have faster consensus rate than low-priority customers in the block queue, and therefore more low-priority customers remain waiting in the customer queue before being batched. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-41 illustrates the impact of the arrival rate of high-priority customers $\lambda_H$ on the blocking probabilities for high-priority, low-priority, and overall customers. As $\lambda_H$ increases, the blocking probability increases across all priority levels. The rise is most significant for $P_{b_L}$, who face greater difficulty being admitted into the system due to the increased presence of high-priority arrivals. This trend reflects the effect of the non-preemptive priority mechanism, where high-priority customers dominate the queue and are less probability to be blocked, while low-priority customers experience higher blocking rates as system congestion intensifies. In addition, $P_{b_H}$ is smaller than $P_{b_L}$. This is because the capacity limit of the customer queue for the high-priority customers is no smaller than that for low-priority customers. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-42 illustrates the impact of the arrival rate of high-priority customers $\lambda_H$ on the system throughputs for high-priority, low-priority, and overall customers. As $\lambda_H$ increases, $T_{h_H}$ increases and $T_{h_L}$ decreases. The $T_h$ remains relatively stable, as the gain in $T_{h_H}$ compensates for the loss in $T_{h_L}$. This behavior reflects the shift in resource allocation under the non-preemptive priority mechanism, where increasing $\lambda_H$ leads to more system capacity being devoted to high-priority customers at the expense of low-priority ones. In addition, $T_{h_H}$ is smaller than $T_{h_L}$. This is because the customer arrival rate of the high-priority customers is smaller than that of low-priority customers. Lastly, the analytical results are in good agreement with the simulation results.

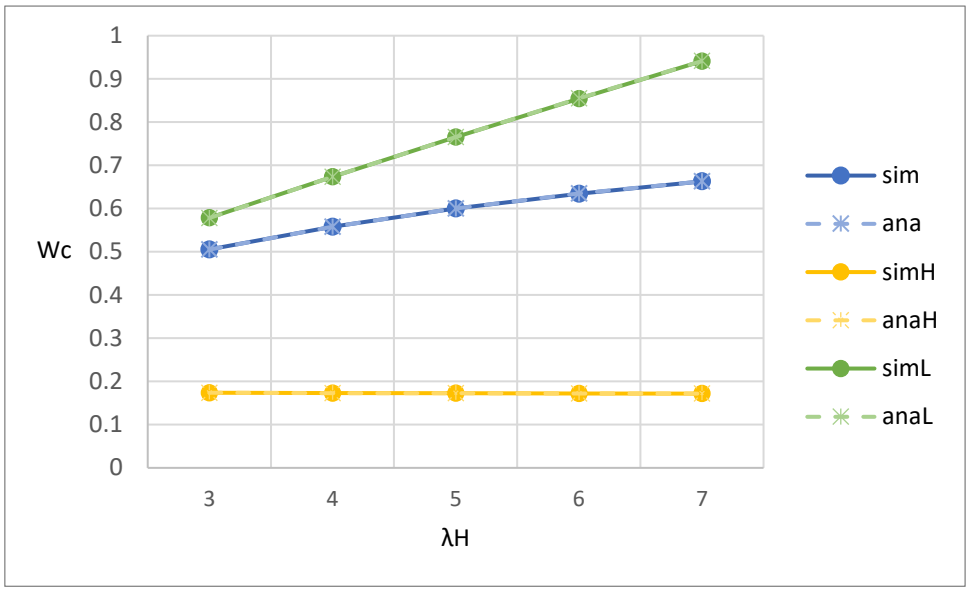


Figure 5-37: Effect of arrival rate on average waiting time in the customer queue

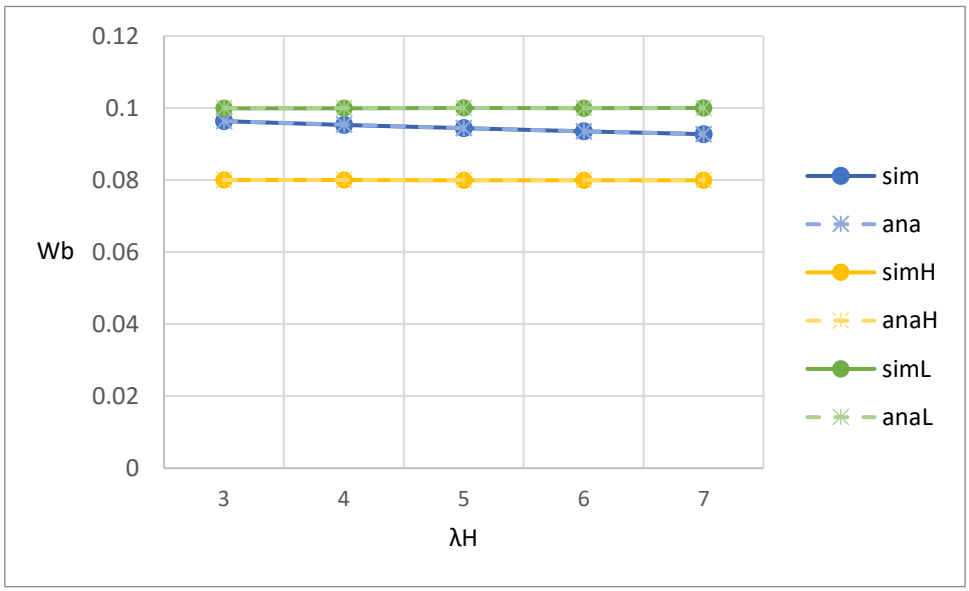


Figure 5-38: Effect of arrival rate on average waiting time in the block queue

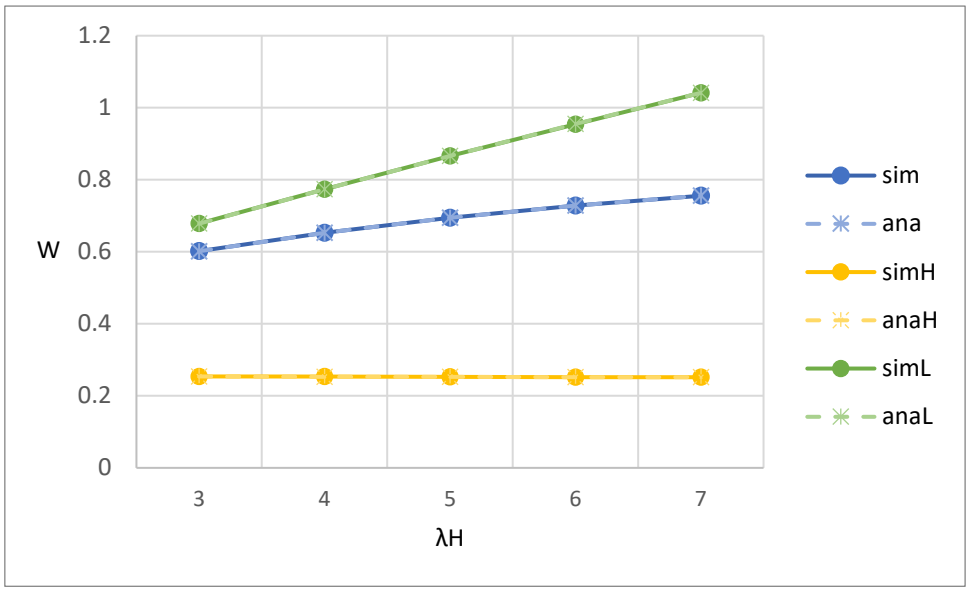


Figure 5-39: Effect of arrival rate on average waiting time in the system

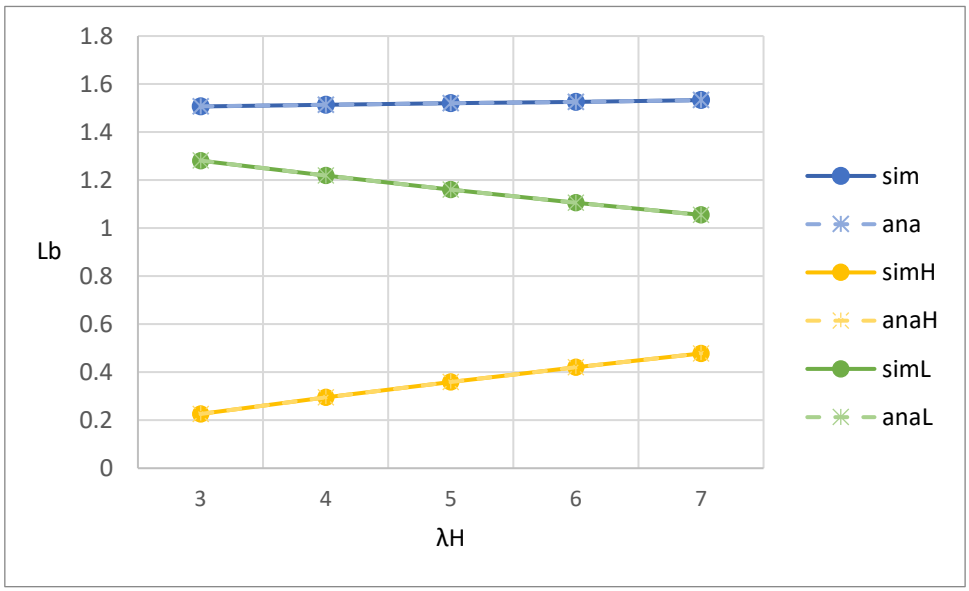


Figure 5-40: Effect of arrival rate on average number of customers in block queue

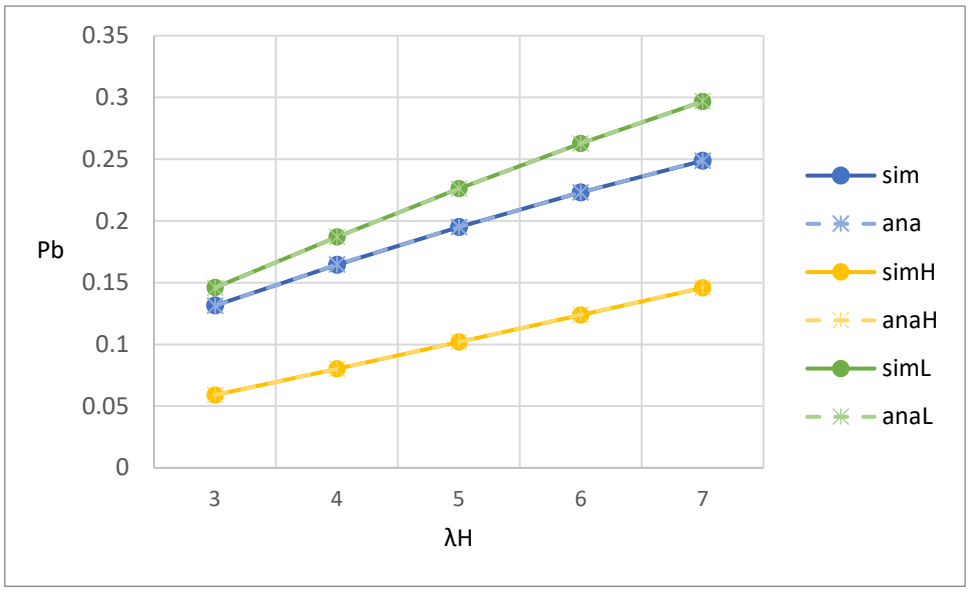


Figure 5-41: Effect of arrival rate on blocking probability

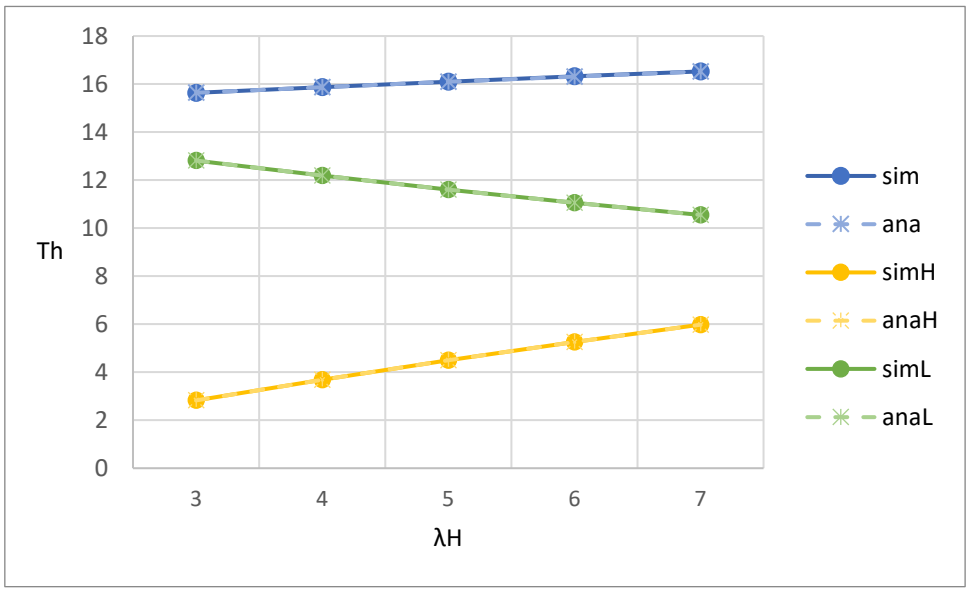


Figure 5-42: Effect of arrival rate on throughput

### 5.2.3. Block generation rate

Figure 5-43 to Figure 5-48 show the relationship between various performance metrics and the block generation rate of high priority customers $\mu_{1_H}$. Both simulation results and analytical results are shown for comparison.

Figure 5-43 illustrates the impact of the block generation rate of high-priority customers $\mu_{1_H}$ on the average waiting times in the customer queue for high-priority, low-priority, and overall customers. As $\mu_{1_H}$ increases, the average waiting time in the customer queue decreases across all priority levels. The decline is more substantial for low-priority customers, who benefit from the increased service opportunities enabled by faster block generation. Although high-priority customers also experience shorter waiting times, their improvement is less pronounced since their queuing delay is already low. In addition, the $W_{c_H}$ is much smaller than $W_{c_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-44 illustrates the impact of the block generation rate of high-priority customers $\mu_{1_H}$ on the average waiting times in the block queue for high-priority, low-priority, and overall customers. As $\mu_{1_H}$ increases, the average waiting time in the block queue remains nearly constant across all priority levels. This indicates that the time each block spends in the consensus queue is determined by the associated consensus rate and system transition rate, and is independent of the block generation rate. In addition, the $W_{b_H}$ is smaller than $W_{b_L}$. This is because $\mu_{2_H}$ is larger than $\mu_{2_L}$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-45 illustrates the impact of the block generation rate of high-priority customers $\mu_{1_H}$ on the average waiting times in the system for high-priority, low-priority, and overall customers. As $\mu_{1_H}$ increases, average waiting time in the system decreases across all priority levels. The rise is primarily contributed by $W_L$, while high-priority customers also benefit from more frequent block formation. This is because higher block generation rates allow customers to be grouped and processed more frequently, which reduces congestion in the customer queue. In addition, the $W_H$ is much smaller than $W_L$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue and $\mu_{2_H}$ is larger than $\mu_{2_L}$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-46 illustrates the impact of the block generation rate of high-priority

customers $\mu_{1_H}$ on the average numbers of customers in the block queue for high-priority, low-priority, and overall customers. As $\mu_{1_H}$ increases, the $L_b$ gradually increases. The rise is primarily contributed by $L_{b_L}$, while $L_{b_H}$ increases only slightly and remains at a relatively low level. This is because a higher $\mu_{1_H}$ allows high-priority customers to be processed more quickly, which indirectly leads to more low-priority customers forming batches. In addition, $L_{b_H}$ is smaller than $L_{b_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue and therefore more low-priority customers remain waiting in the customer queue before being batched. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-47 illustrates the impact of the block generation rate of high-priority customers $\mu_{1_H}$ on the blocking probabilities for high-priority, low-priority, and overall customers. As $\mu_{1_H}$ increases, the blocking probability decreases across all priority levels. This is because the higher block generation rate allows high-priority customers to be served more frequently, which in turn release the capacity in the customer queue for low-priority customers. As a result, the probability that low-priority customers are blocked is reduced. In addition, $P_{b_H}$ is smaller than $P_{b_L}$. This is because the capacity limit of the customer queue for the high-priority customers is no smaller than that for low-priority customers. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-48 illustrates the impact of the block generation rate of high-priority customers $\mu_{1_H}$ on the system throughputs for high-priority, low-priority, and overall customers. As $\mu_{1_H}$ increases, the system throughput increases across all priority levels and then gradually saturates. The rise is primarily contributed by $T_{h_L}$, which increases more significantly due to the release of queue capacity made possible by faster processing of high-priority blocks. In contrast, $T_{h_H}$ eventually approaches a limit determined by $\lambda_H$. In addition, $T_{h_H}$ is smaller than $T_{h_L}$. This is because the customer arrival rate of the high-priority customers is smaller than that of low-priority customers. Lastly, the analytical results are in good agreement with the simulation results.

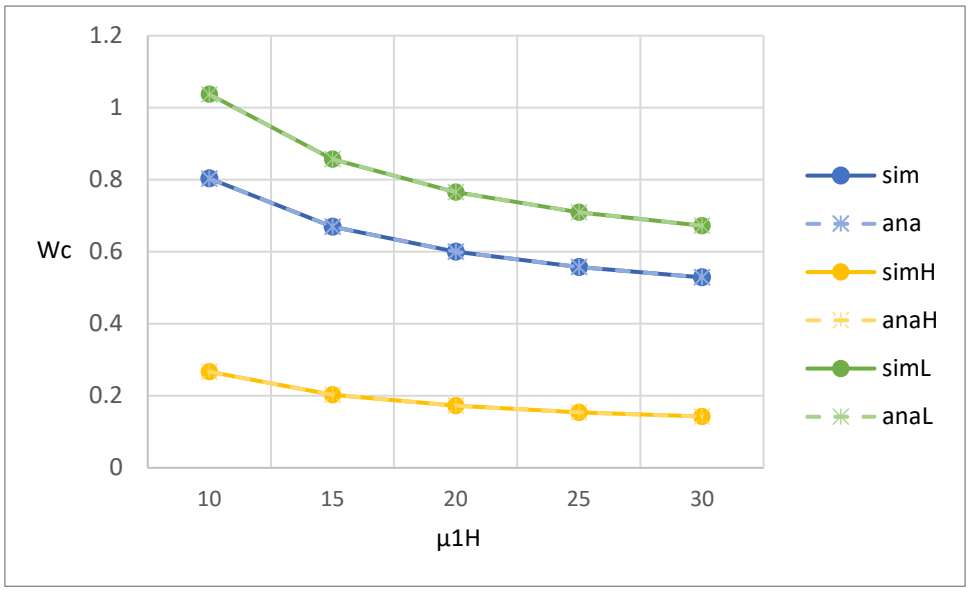


Figure 5-43: Effect of block generation rate on average waiting time in the customer queue

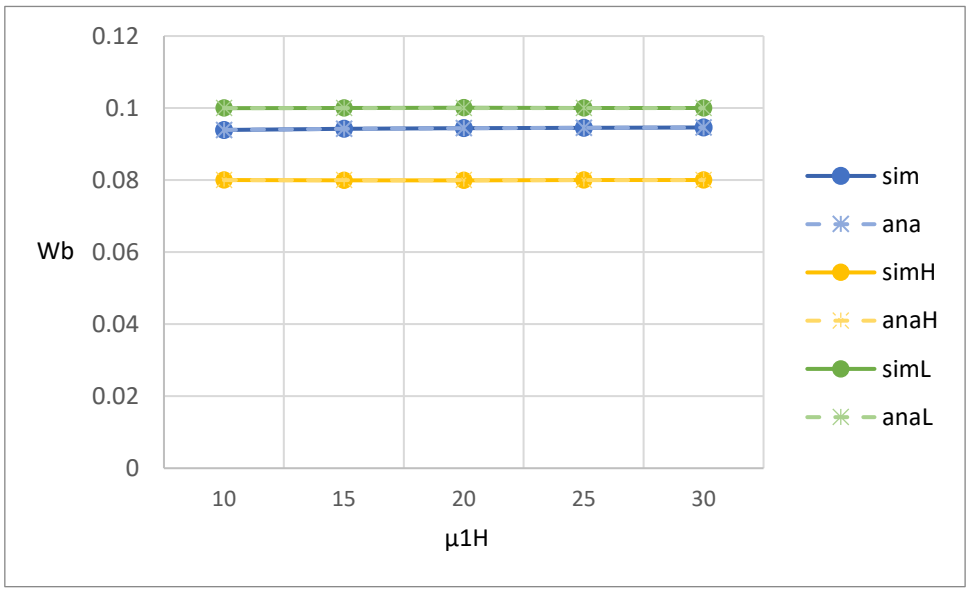


Figure 5-44: Effect of block generation rate on average waiting time in the block queue

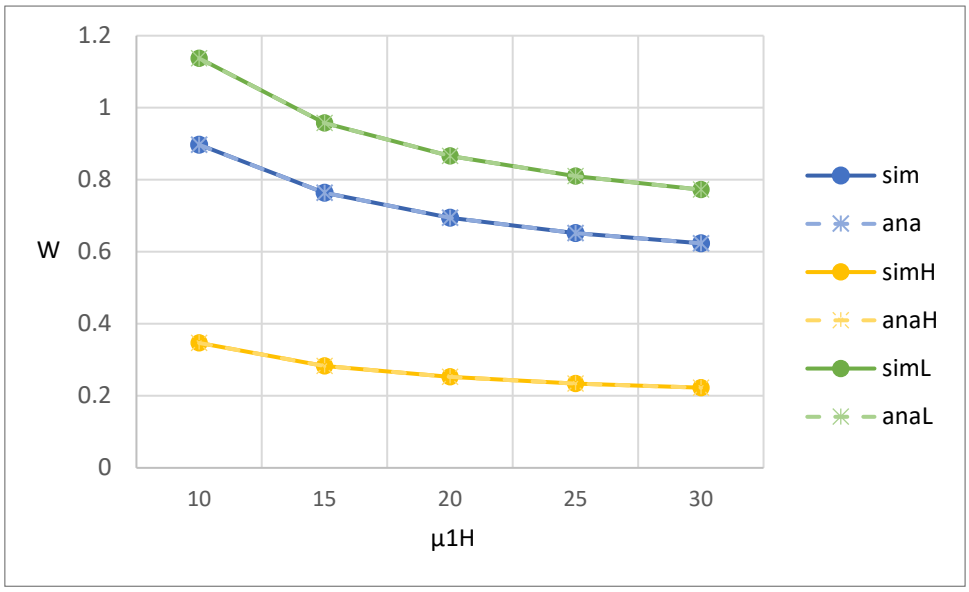


Figure 5-45: Effect of block generation rate on average waiting time in the system

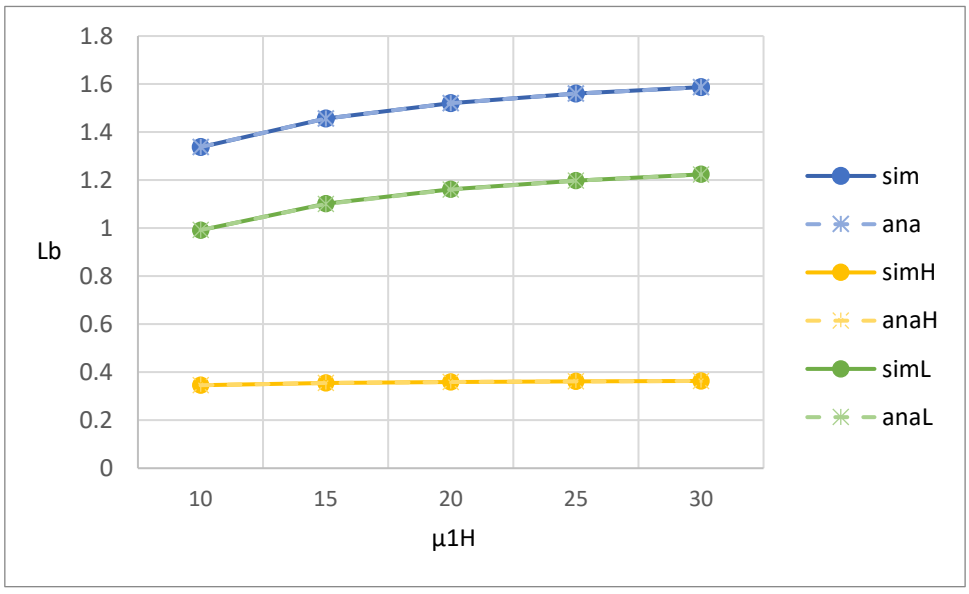


Figure 5-46: Effect of block generation rate on average number of customers in block queue

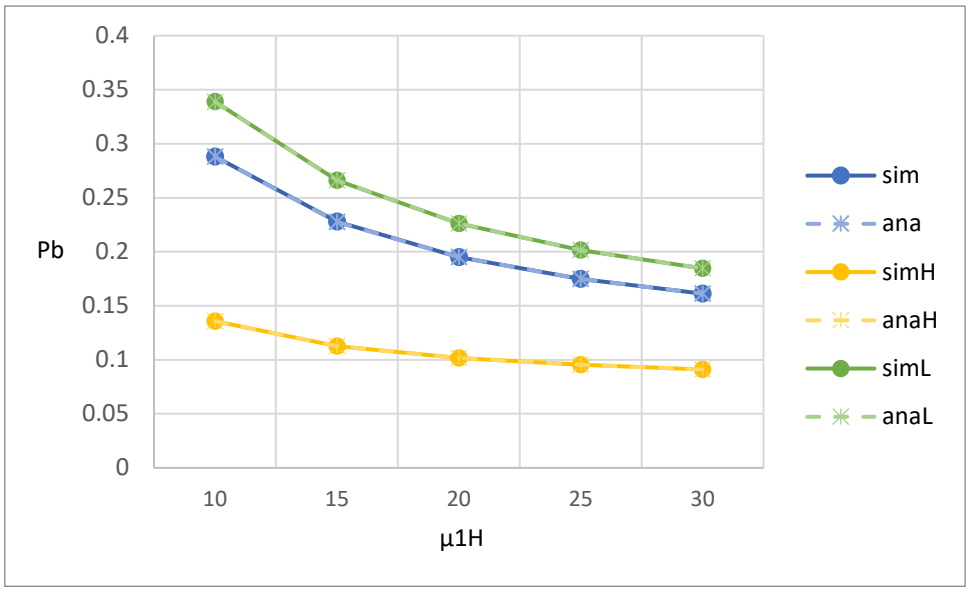


Figure 5-47: Effect of block generation rate on blocking probability

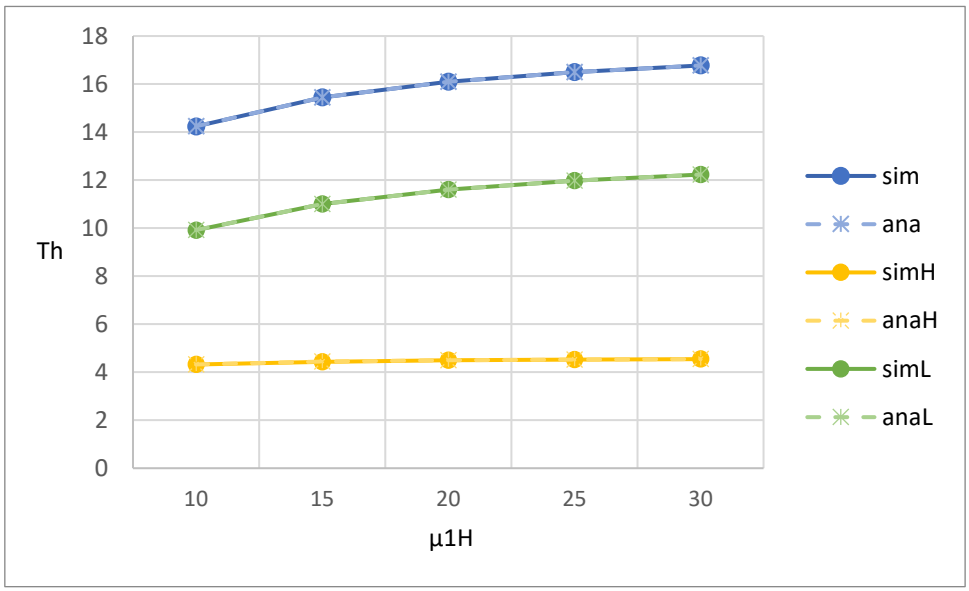


Figure 5-48: Effect of block generation rate on system throughput

### 5.2.4. Consensus rate

Figure 5-49 to Figure 5-54 show the relationship between various performance metrics and the consensus rate of high-priority customers $\mu_{2_H}$. Both simulation results and analytical results are shown for comparison.

Figure 5-49 illustrates the impact of the consensus rate of high-priority customers $\mu_{2_H}$ on the average waiting times in the customer queue for high-priority, low-priority, and overall customers. As $\mu_{2_H}$ increases, the $W_c$ decreases. The reduction is more pronounced for $W_{c_L}$, while $W_{c_H}$ remains relatively stable. This is because higher $\mu_{2_H}$ allows high-priority blocks to complete consensus more quickly, thereby shortening the duration that high-priority customers occupy system capacity. As a result, more capacity becomes available for low-priority customers, so that reducing $W_{c_L}$. In addition, the $W_{c_H}$ is much smaller than $W_{c_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-50 illustrates the impact of the consensus rate of high-priority customers $\mu_{2_H}$ on the average waiting times in the block queue for high-priority, low-priority, and overall customers. As $\mu_{2_H}$ increases, the $W_b$ decreases slightly. The decline is primarily contributed by $W_{b_H}$, while $W_{b_L}$ remains nearly constant. This is because a higher $\mu_{2_H}$ allows high-priority blocks to complete consensus more quickly, reducing the amount of time the high-priority customers spend waiting in the block queue. Since the consensus rate of low-priority customers is not affected, their waiting time remains unchanged. In addition, at $\mu_{2_H} = 15$, $W_{b_H}$ is noticeably higher than $W_{b_L}$, but the two curves intersect around $\mu_{2_H} = 20$, after which $W_{b_H}$ becomes lower. This crossover occurs because $\mu_{2_H}$ starts lower than the fixed $\mu_{2_L} = 20$, and increasing $\mu_{2_H}$ improves high-priority processing speed. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-51 illustrates the impact of the consensus rate of high-priority customers $\mu_{2_H}$ on the average waiting times in the system for high-priority, low-priority, and overall customers. As $\mu_{2_H}$ increases, the average waiting time in the system decreases steadily for all priority levels. This is because faster $\mu_{2_H}$ reduces delays in the block queue, thereby improving system efficiency. While $W_L$ also decreases slightly as system capacity is released from high-priority customers. This trend highlights how improving consensus efficiency for high-priority customers can enhance overall system flow and reduce $W_H$ and $W_L$. In addition, the $W_H$ is much smaller than $W_L$. This is because the high-priority customers have non-preemptive priority over low-priority

customers in the customer queue. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-52 illustrates the impact of the consensus rate of high-priority customers $\mu_{2_H}$ on the average numbers of customers in the block queue for high-priority, low-priority, and overall customers. As $\mu_{2_H}$ increases, the $L_b$ slightly decreases. This change is mainly driven by a noticeable reduction in $L_{b_H}$, while $L_{b_L}$ gradually increases due to the admission of more low-priority customers as system capacity is released by faster $\mu_{2_H}$. This reflects how increasing $\mu_{2_H}$ can reduce queue capacity for high-priority customers while indirectly increasing $L_{b_L}$. In addition, $L_{b_H}$ is smaller than $L_{b_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue and $\mu_{2_H}$ is larger than $\mu_{2_L}$, and therefore more low-priority customers remain waiting in the customer queue before being batched. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-53 illustrates the impact of the consensus rate of high-priority customers $\mu_{2_H}$ on the blocking probabilities for high-priority, low-priority, and overall customers. As $\mu_{2_H}$ increases, the blocking probability decreases across all priority levels. This decline is more significant for high-priority customers, whose blocks are processed more rapidly with higher $\mu_{2_H}$, resulting in fewer delays and fewer blocked arrivals. Also, faster processing of high-priority customers indirectly frees up capacity in the customer queue, reducing the $P_{b_L}$ as well. In addition, $P_{b_H}$ is smaller than $P_{b_L}$. This is because the capacity limit of the customer queue for the high-priority customers is no smaller than that for low-priority customers. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-54 illustrates the impact of the consensus rate of high-priority customers $\mu_{2_H}$ on the system throughputs for high-priority, low-priority, and overall customers. As $\mu_{2_H}$ increases, the system throughput increases across all priority levels and then gradually saturates. The rise is primarily contributed by $T_{h_L}$, which increases more significantly due to the release of queue capacity made possible by faster processing of high-priority blocks. In contrast, $T_{h_H}$ eventually approaches a limit determined by $\lambda_H$. In addition, $T_{h_H}$ is smaller than $T_{h_L}$. This is because the customer arrival rate of the high-priority customers is smaller than that of low-priority customers. Lastly, the analytical results are in good agreement with the simulation results.

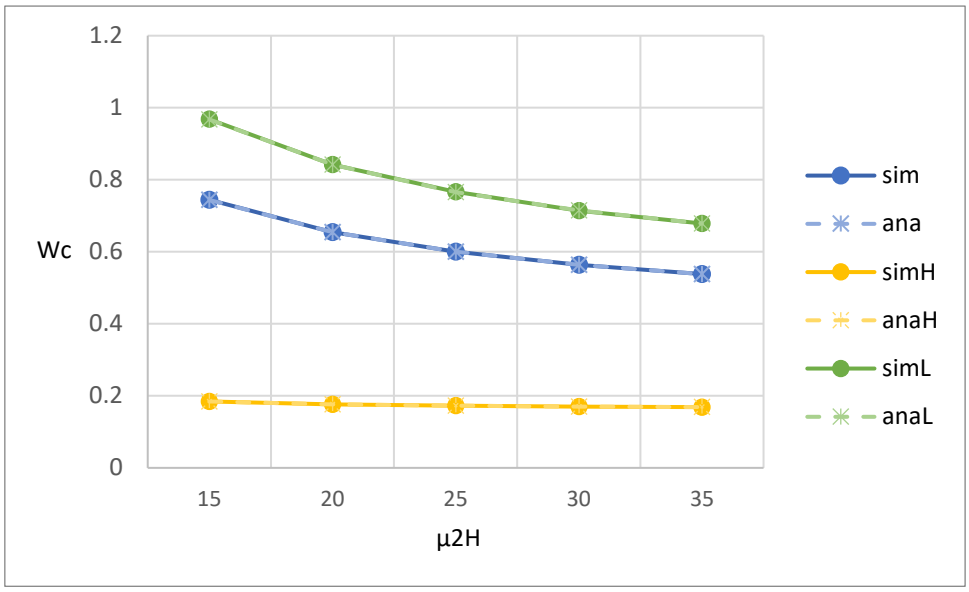


Figure 5-49: Effect of consensus rate on average waiting time in the customer queue

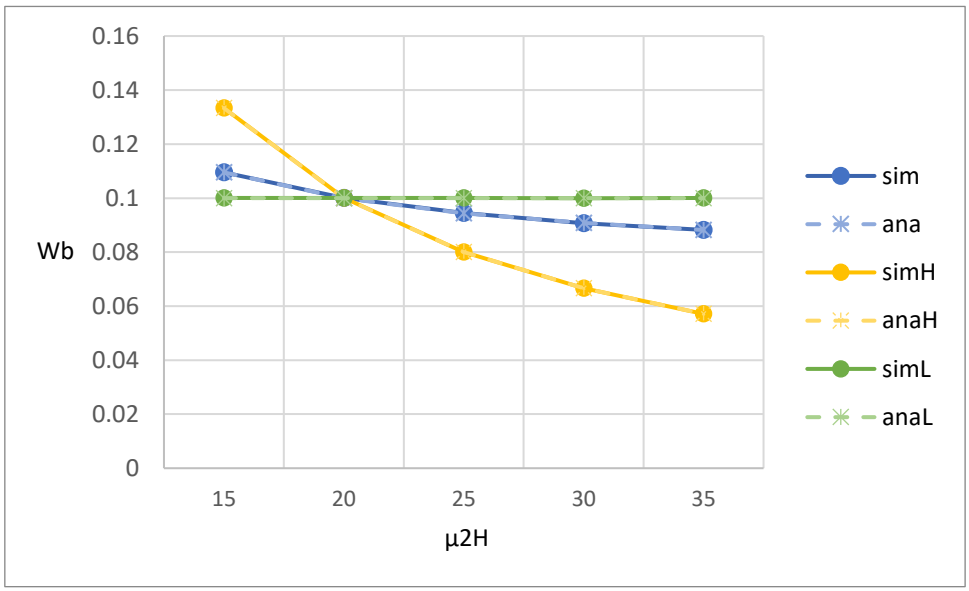


Figure 5-50: Effect of consensus rate on average waiting time in the block queue

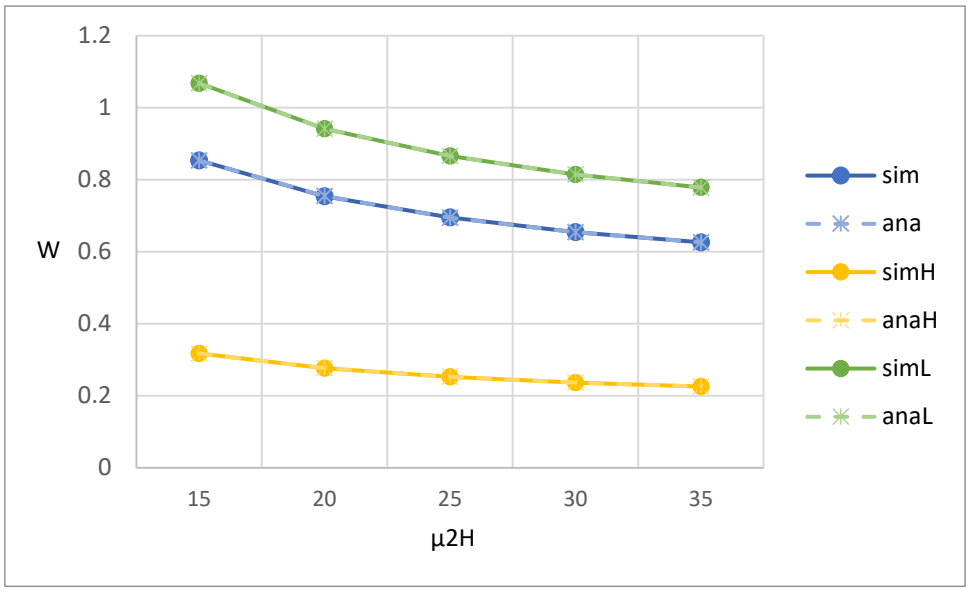


Figure 5-51: Effect of consensus rate on average waiting time in the system

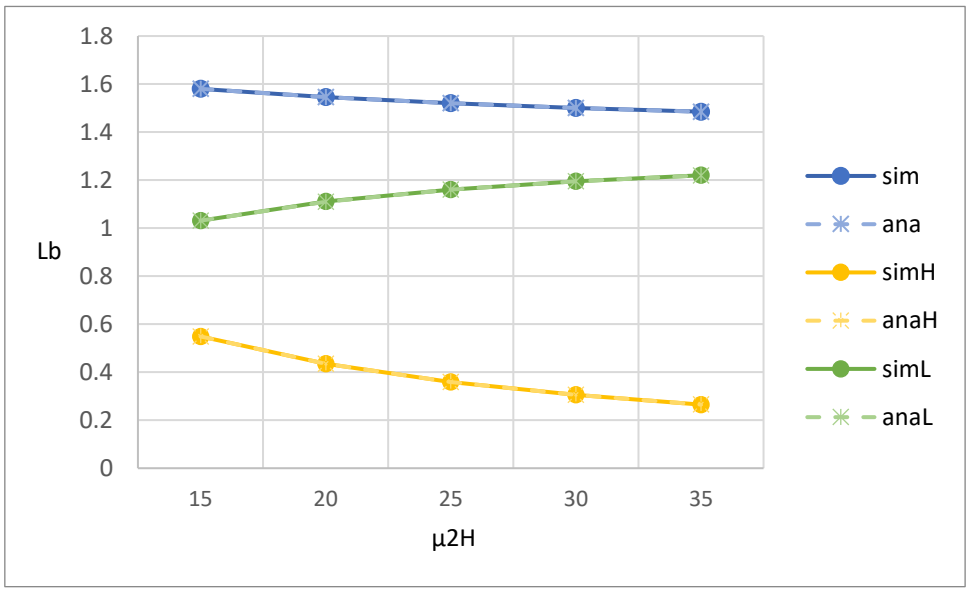


Figure 5-52: Effect of consensus rate on average number of customers in block queue

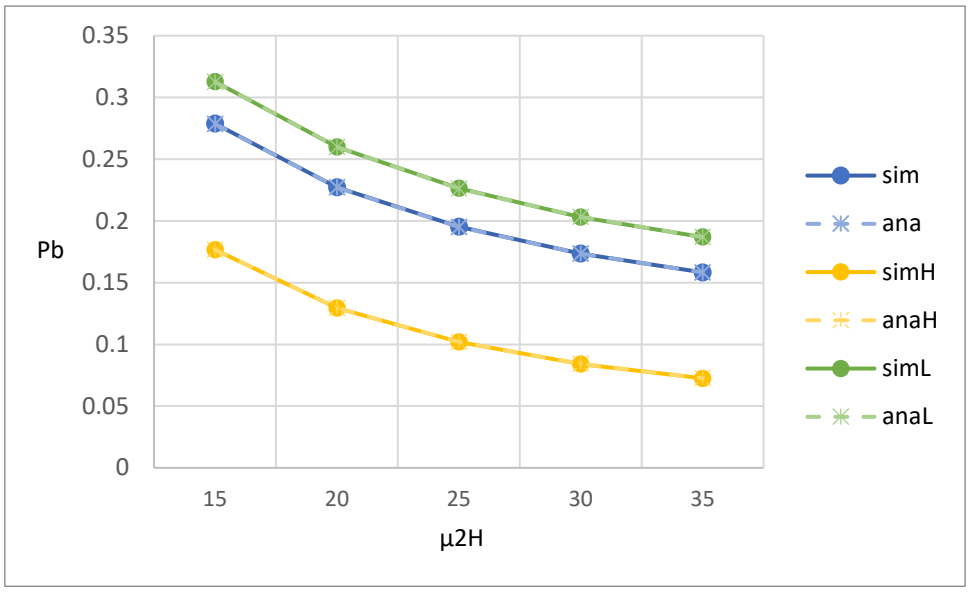


Figure 5-53: Effect of consensus rate on blocking probability

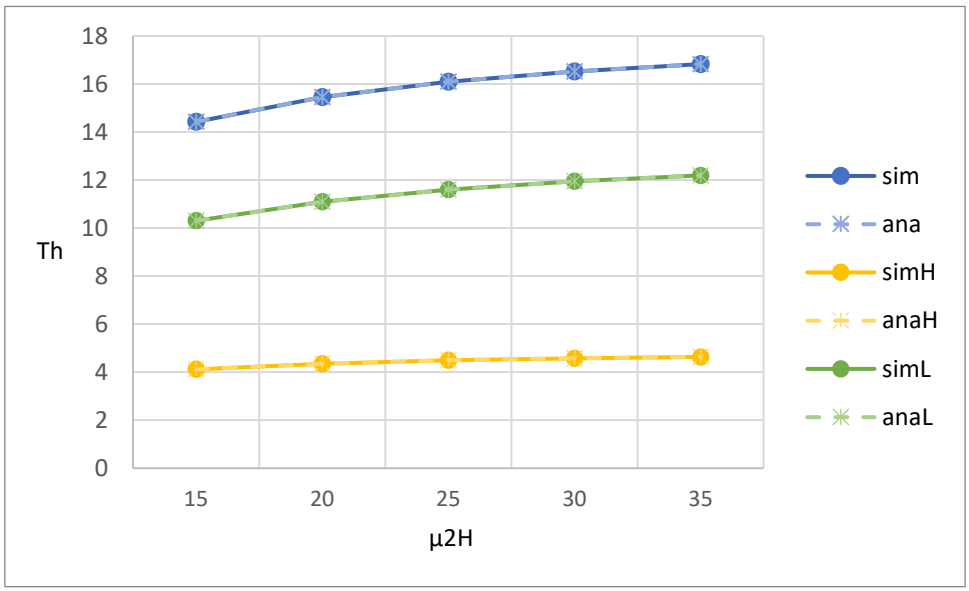


Figure 5-54: Effect of consensus rate on system throughput

### 5.2.5. Transition rate (ON to OFF)

Figure 5-55 to Figure 5-60 show the relationship between various performance metrics and the transition rate $\alpha$. Both simulation results and analytical results are shown for comparison.

Figure 5-55 illustrates the impact of the transition rate $\alpha$ on the average waiting times in the customer queue for high-priority, low-priority, and overall customers. As $\alpha$ increases, the average waiting time in the customer queue rises steadily across all priority levels. The increase is more pronounced for low-priority customers, who are more affected by interruptions in service. This is because more frequent transitions from ON to OFF reduce the availability of block generation service, causing longer queueing delays for arriving customers, especially for customers with lower priority. In addition, the $W_{c_H}$ is much smaller than $W_{c_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-56 illustrates the impact of the transition rate $\alpha$ on the average waiting times in the block queue for high-priority, low-priority, and overall customers. As $\alpha$ increases, the average waiting time in the block queue increase steadily across all priority levels. This is because more frequent service interruptions delay consensus processing, resulting in longer waiting times for customer(s) in block queue. This effect is observed for both high- and low-priority customers, with low-priority customers experiencing slightly longer delay. In addition, the $W_{b_H}$ is smaller than $W_{b_L}$. This is because $\mu_{2_H}$ is larger than $\mu_{2_L}$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-57 illustrates the impact of the transition rate $\alpha$ on the average waiting times in the system for high-priority, low-priority, and overall customers. As $\alpha$ increases, the average waiting time in the system increase steadily across all priority levels. The increase is more significant for $W_L$, while $W_H$ increases slightly. This is because more frequent service interruptions caused by transitions to the OFF state reduce overall availability of block generation and consensus service, leading to longer queueing delays for customers in the system. In addition, the $W_H$ is much smaller than $W_L$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue and $\mu_{2_H}$ is larger than $\mu_{2_L}$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-58 illustrates the impact of the transition rate $\alpha$ on the average numbers of customers in the block queue for high-priority, low-priority, and overall customers. As $\alpha$ increases, the average number of customers in the block queue increase steadily across all priority levels. This is because more frequent service interruptions of block generation processing cause more customers to wait in the customer queue while forming batches from the customer queue. In addition, $L_{b_H}$ is smaller than $L_{b_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue and $\mu_{2_H}$ is larger than $\mu_{2_L}$, and therefore more low-priority customers remain waiting in the customer queue before being batched. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-59 illustrates the impact of the transition rate $\alpha$ on the blocking probabilities for high-priority, low-priority, and overall customers. As $\alpha$ increases, the blocking probability increase steadily across all priority levels. The rise is more pronounced for $P_{b_L}$, while $P_{b_H}$ increases at a slower rate. This is because more frequent service interruptions reduce the system's capacity to process customers, which increases the probability that the customer queue reaches its capacity and causes incoming arrivals to be blocked. In addition, $P_{b_H}$ is smaller than $P_{b_L}$. This is because the capacity limit of the customer queue for the high-priority customers is no smaller than that for low-priority customers. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-60 illustrates the impact of the transition rate $\alpha$ on the system throughputs for high-priority, low-priority, and overall customers. As $\alpha$ increases, the system throughput decreases gradually across all priority levels. This is because more frequent service interruptions reduce the chance for block generation and consensus processing, thereby limiting the rate at which customers are served and ultimately lowering the $T_h$. The decline is more pronounced for $T_{h_L}$, whose service opportunities are more severely impacted by limited availability, while $T_{h_H}$ remains relatively stable. Lastly, the analytical results are in good agreement with the simulation results.

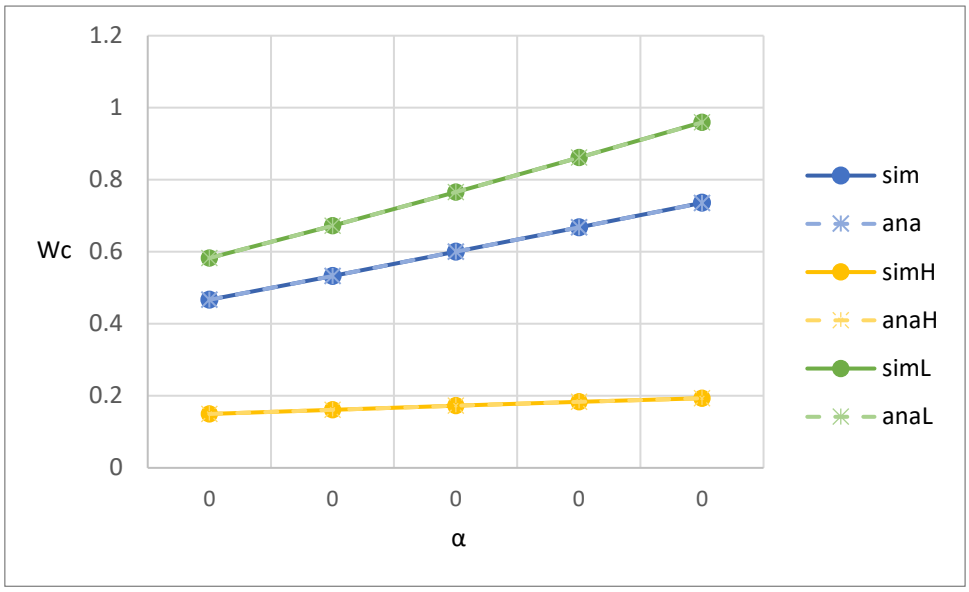


Figure 5-55: Effect of transition rate on average waiting time in the customer queue in the system

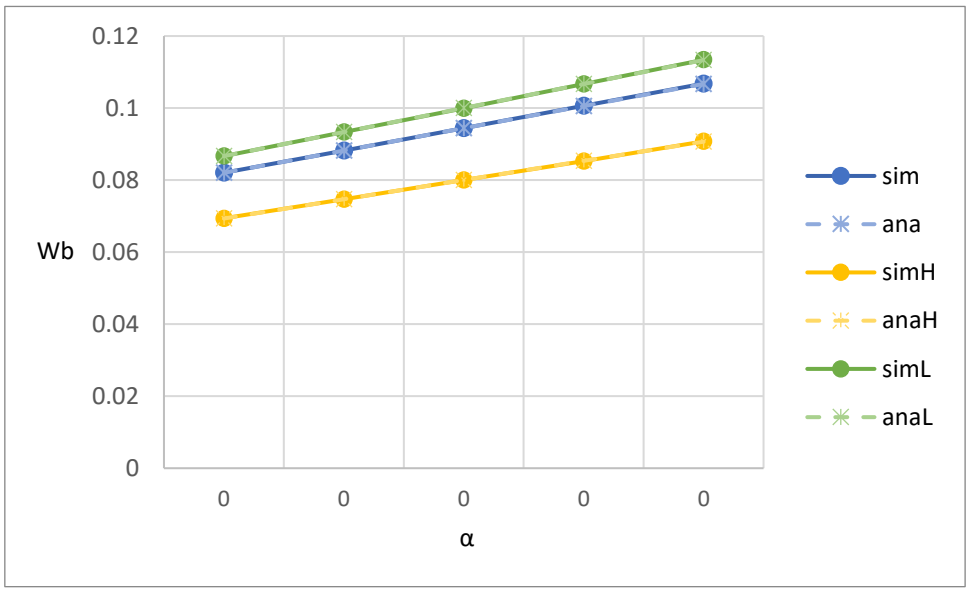


Figure 5-56: Effect of transition rate on average waiting time in the block queue

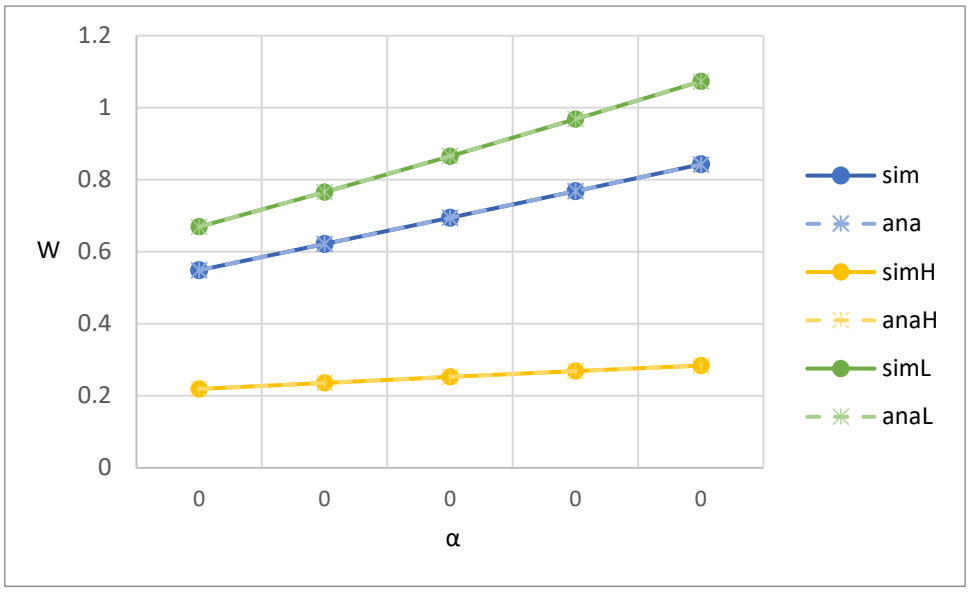


Figure 5-57: Effect of transition rate on average waiting time in the system

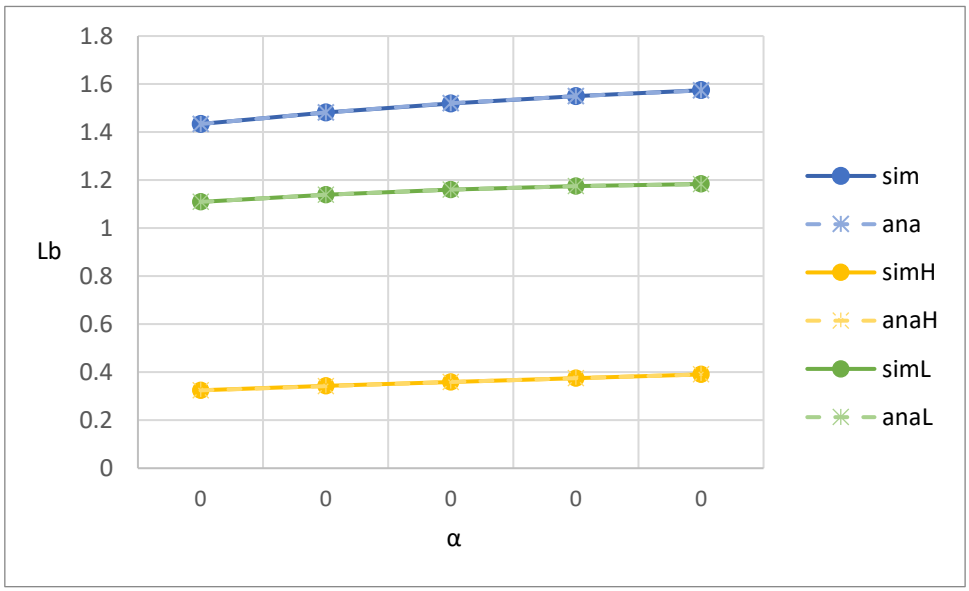


Figure 5-58: Effect of transition rate on average number of customers in block queue

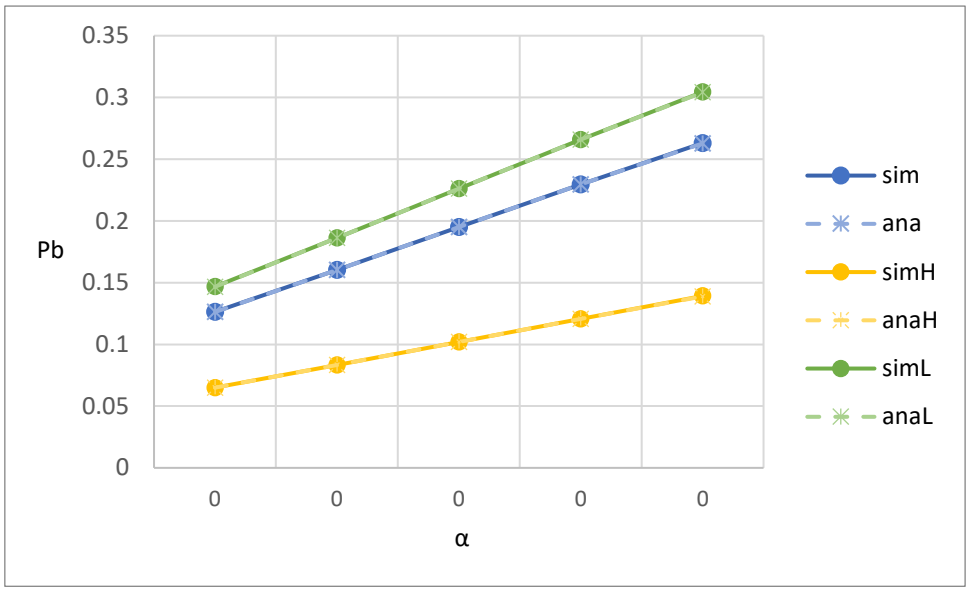


Figure 5-59: Effect of transition rate on blocking probability

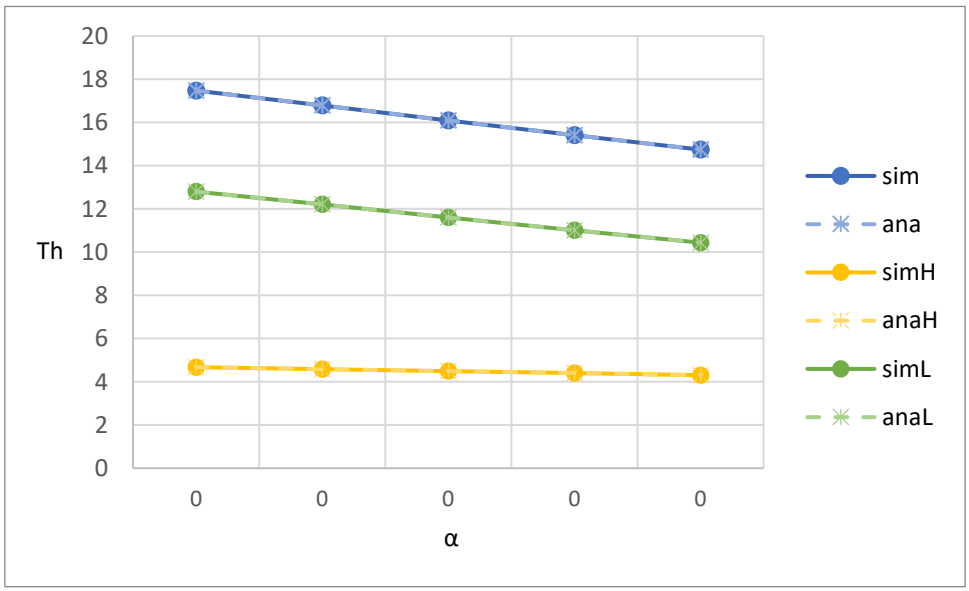


Figure 5-60: Effect of transition rate on system throughput

## 5.3. Scenario 3: Single-Class Customer with Impatience

The default values are as provided as below: $\lambda = 20,\ \mu_1 = 20,\ \mu_2 = 20,\ \alpha = \beta = 15,\ \gamma = 0.5$. The maximum capacity of the system is $N = 20$, and the maximum block size is $b = 5$.

### 5.3.1. Block size

Figure 5-61 to Figure 5-67 show the relationship between various performance metrics and the block size $b$. Both simulation results and analytical results are shown for comparison.

Figure 5-61 illustrates the impact of the block size $b$ on the average waiting time in the customer queue ($W_c$). As $b$ increases, $W_c$ decreases significantly, particularly at smaller block size. This is because larger blocks allow more customers to be served per service cycle, thereby reducing the time customers spend waiting in the queue. Additionally, $W_c$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which further suppresses queue buildup and contributes to the decline in $W_c$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-62 illustrates the impact of the block size $b$ on the average waiting time in the block queue ($W_b$). As $b$ increases, $W_b$ remains nearly constant. This indicates that the time each block spends in the consensus queue is determined by the consensus rate and system state transition rate, and is independent of block size. Furthermore, $W_b$ with impatience is the same as that without impatience. This is because once a block is formed, impatience mechanism has no effect on it. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-63 illustrates the impact of the block size $b$ on the average waiting time in the system ($W$). As $b$ increases, $W$ decreases significantly, particularly at smaller block size. This is because larger blocks allow more customers to be served per service cycle, thereby reducing the time customers spend waiting in the queue. Additionally, $W$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which further suppresses queue buildup and contributes to the decline in $W$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-64 illustrates the impact of the block size $b$ on the average number of customers in the block queue ($L_b$). As $b$ increases, $L_b$ initially grows and then stabilizes around constant value. This indicates that although larger blocks permit more customers per batch, the average block occupancy tends to saturate when the customer arrival rate is equal to the effective service rate of the customer queue, where the effective service rate of the customer queue is dependent on the block generation rate, the system transition rate, and the average block size. Additionally, $L_b$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which contributes to a reduction in $L_b$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-65 illustrates the impact of the block size $b$ on the blocking probability ($P_b$). As $b$ increases, $P_b$ steadily decreases. This is because larger blocks allow more customers to be served per block generation cycle, thereby reducing the chance of the customer queue reaching its capacity limit. Additionally, $P_b$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which contributes to a reduction in $P_b$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-66 illustrates the impact of the block size $b$ on the system throughput ($T_h$). As $b$ increases, $T_h$ rises rapidly at first and then gradually saturates. This is because larger blocks enable more customers to be processed per consensus cycle, but the throughput eventually approaches a limit determined by the customer arrival rate and the impatient rate, which is less than the system processing capacity. The system processing capacity is decided by the maximum block size, the block generation rate, consensus rate, and system state transition rate. As a result, $T_h$ with impatience is smaller than that without impatience. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-67 illustrates the impact of the block size $b$ on the impatience probability ($P_{im}$). As $b$ increases, $P_{im}$ steadily decreases. This is because larger blocks allow more customers to be served in each service cycle, which reduces the waiting time in the customer queue. As a result, the probability that customers reach their impatience threshold and leave the system becomes lower. As a result, $P_{im}$ with impatience is larger than without impatience. Lastly, the analytical results are in good agreement with the simulation results.

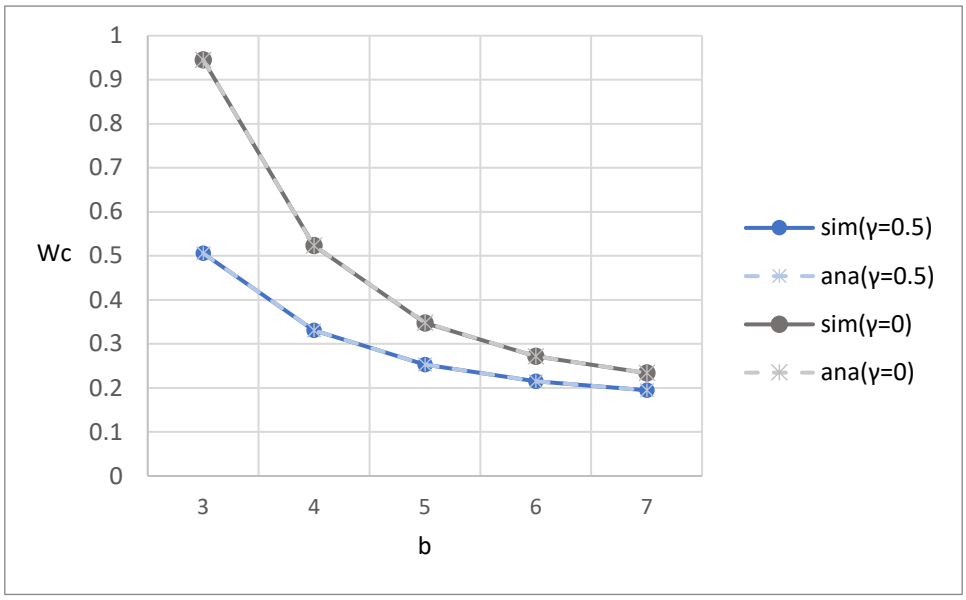


Figure 5-61: Effect of block size on average waiting time in the customer queue

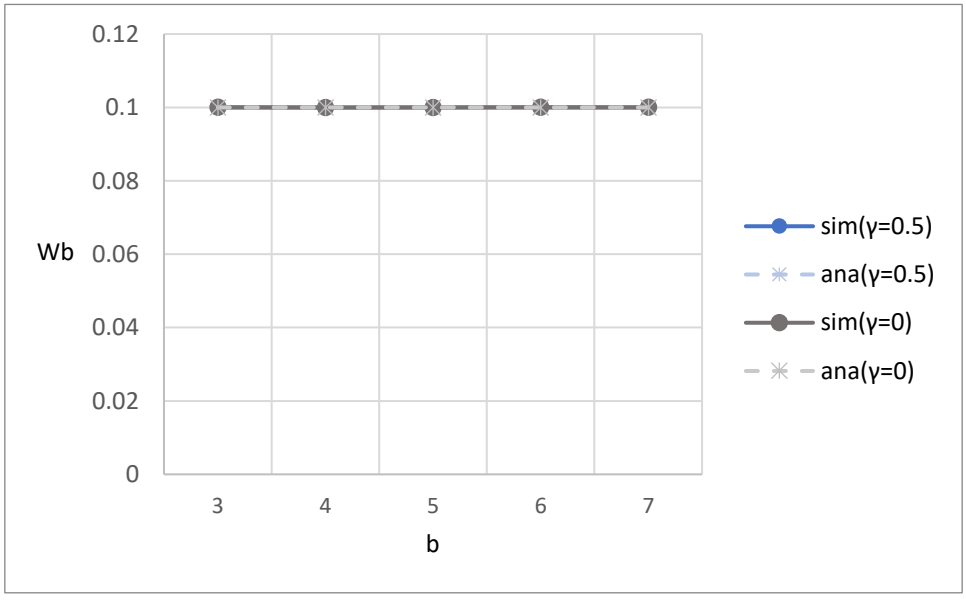


Figure 5-62: Effect of block size on average waiting time in the block queue

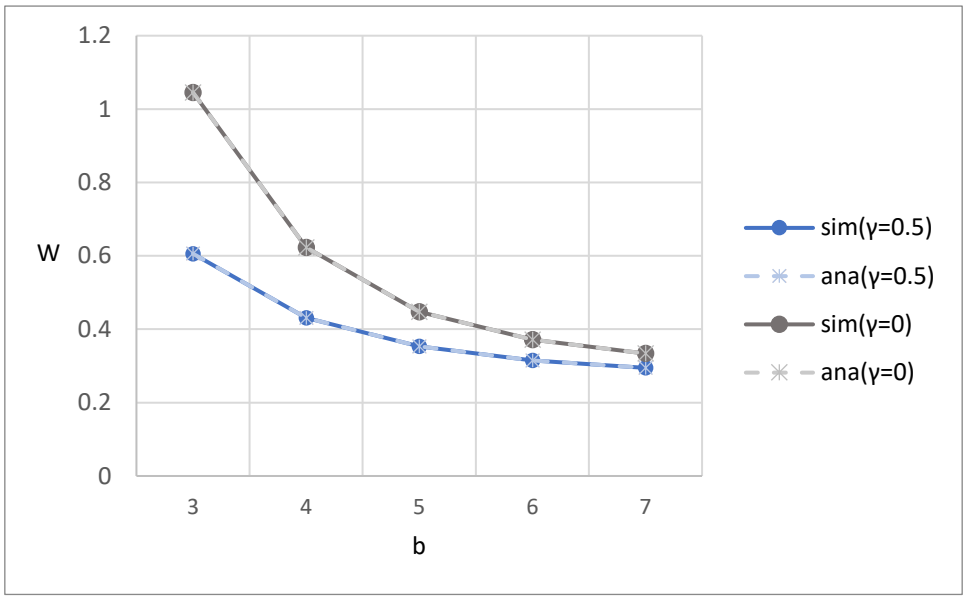


Figure 5-63: Effect of block size on average waiting time in the system

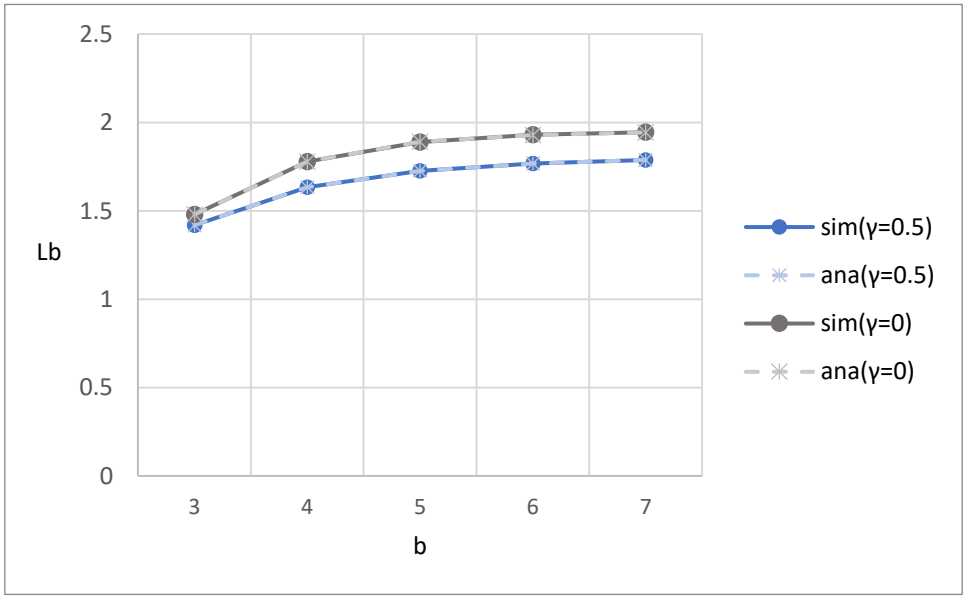


Figure 5-64: Effect of block size on average number of customers in block queue

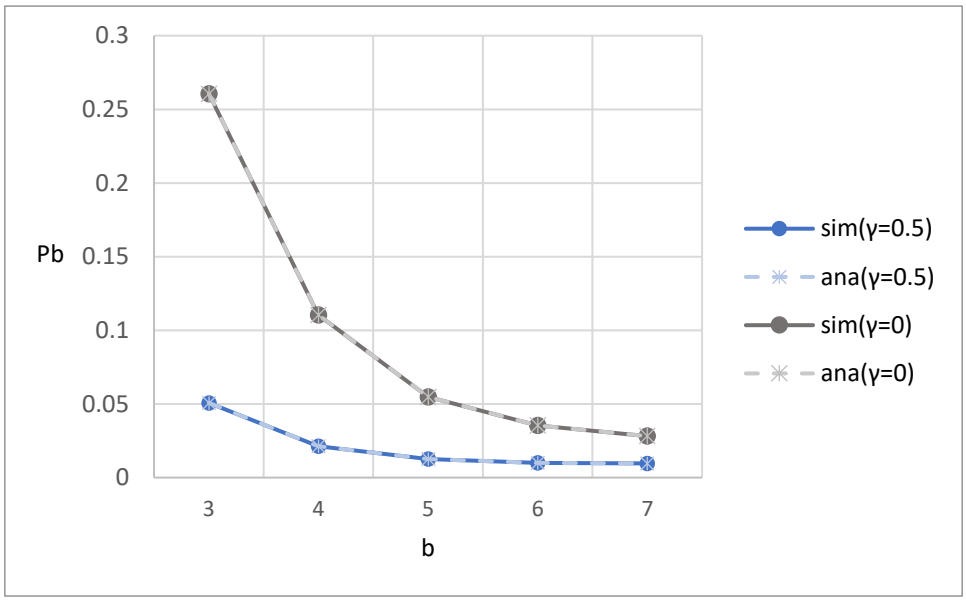


Figure 5-65: Effect of block size on blocking probability

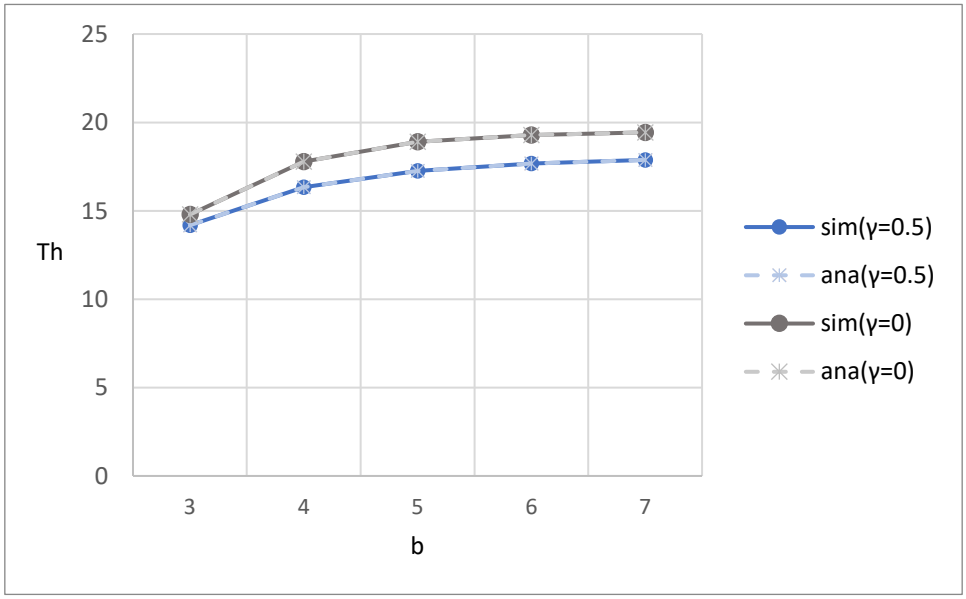


Figure 5-66: Effect of block size on system throughput

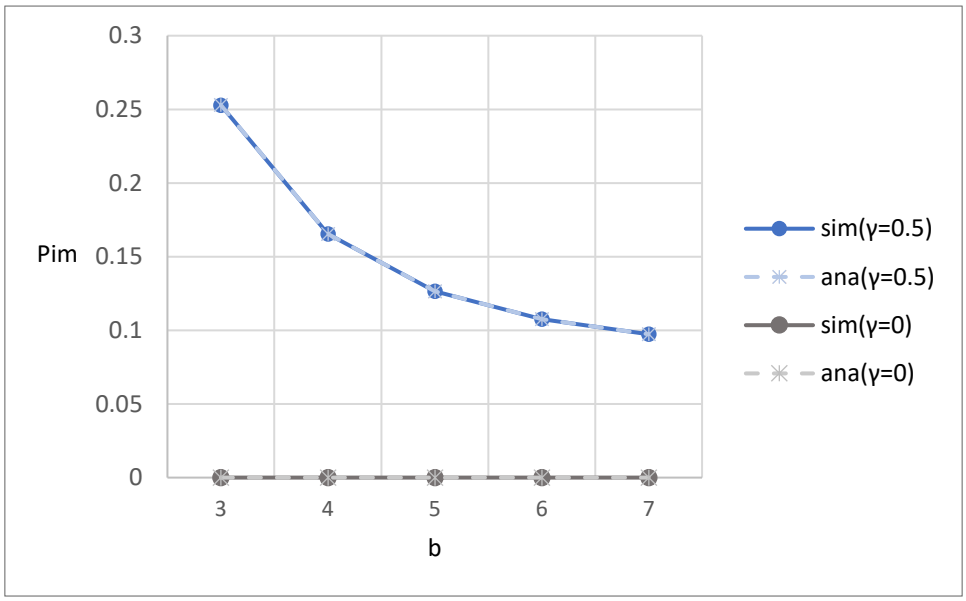


Figure 5-67: Effect of block size on the impatient probability

### 5.3.2. Arrival rate

Figure 5-68 to Figure 5-74 show the relationship between various performance metrics and the arrival rate $\lambda$. Both simulation results and analytical results are shown for comparison.

Figure 5-68 illustrates the impact of the arrival rate $\lambda$ on the average waiting time in the customer queue ($W_c$). As $\lambda$ increases, $W_c$ increases steadily. This is because higher arrival rate leads to more customers entering the system, causing increased congestion and longer queuing delays before customers can be batched into blocks. Additionally, $W_c$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which further suppresses queue buildup and contributes to the decline in $W_c$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-69 illustrates the impact of the arrival rate $\lambda$ on the average waiting time in the block queue ($W_b$). As $\lambda$ increases, $W_b$ remains nearly constant. This indicates that the time each block spends in the consensus queue is determined by the consensus rate and system transition rate, and is independent of the arrival rate. Furthermore, $W_b$ with impatience is the same as that without impatience. This is because once a block is formed, impatience mechanism has no effect on it. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-70 illustrates the impact of the arrival rate $\lambda$ on the average waiting time in the system ($W$). As $\lambda$ increases, $W$ increases steadily. This is because higher arrival rate leads to more customers entering the system, causing increased congestion and longer queuing delays before customers depart from the system. Additionally, $W$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which further suppresses queue buildup and contributes to the decline in $W$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-71 illustrates the impact of the arrival rate $\lambda$ on the average number of customers in the block queue ($L_b$). As $\lambda$ increases, $L_b$ increases steadily. This is because a higher arrival rate leads to more frequent block formation and more customers being accumulated in each block, increasing the average number of customers waiting for consensus. Additionally, $L_b$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to

leave the queue once their impatience threshold is reached, which contributes to a reduction in $L_b$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-72 illustrates the impact of the arrival rate $\lambda$ on the blocking probability ($P_b$). As $\lambda$ increases, $P_b$ rises sharply, specially beyond $\lambda = 20$. This is because a higher arrival rate leads to more customers being accumulated in the customer queue, increasing the chance that incoming customers are blocked due to limited queue capacity. Additionally, $P_b$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which contributes to a reduction in $P_b$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-73 illustrates the impact of the arrival rate $\lambda$ on the system throughput ($T_h$). As $\lambda$ increases, $T_h$ rises rapidly at first and then gradually saturates. This is because higher arrival rates supply more customers into the system, but throughput eventually becomes limited by the impatience rate and the system service capacity, which is decided by the maximum block size, the block generation rate, the consensus rate, and system transition rate. As a result, $T_h$ with impatience is smaller than that without impatience. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-74 illustrates the impact of the arrival rate $\lambda$ on the impatience probability ($P_{im}$). As $\lambda$ increases, $P_{im}$ increases steadily. This is because a higher arrival rate leads to increased queue congestion and longer waiting times in the customer queue. Consequently, more customers are likely to reach their impatience threshold and leave the system before being served. As a result, $P_{im}$ with impatience is larger than without impatience. Lastly, the analytical results are in good agreement with the simulation results.

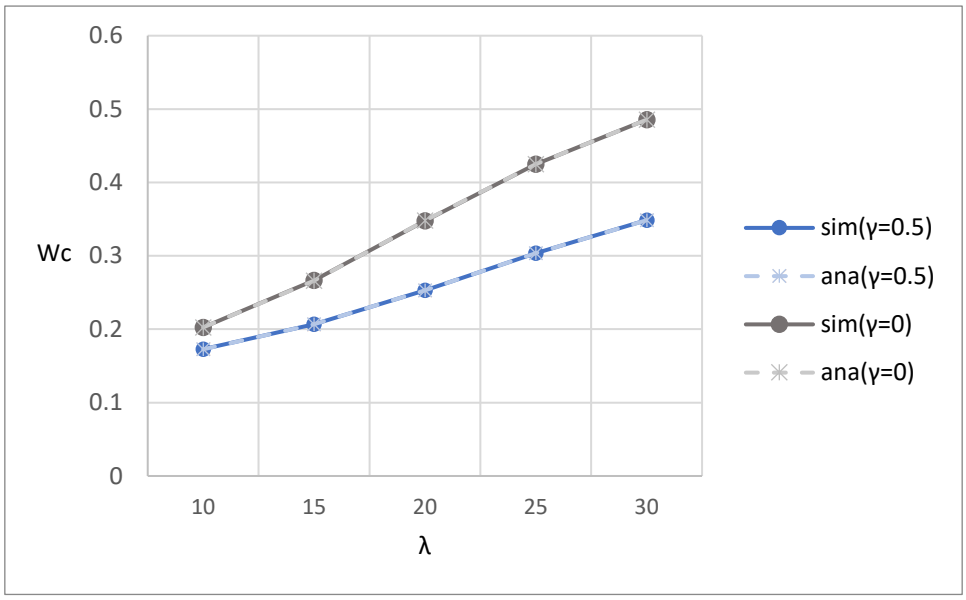


Figure 5-68: Effect of arrival rate on average waiting time in the customer queue

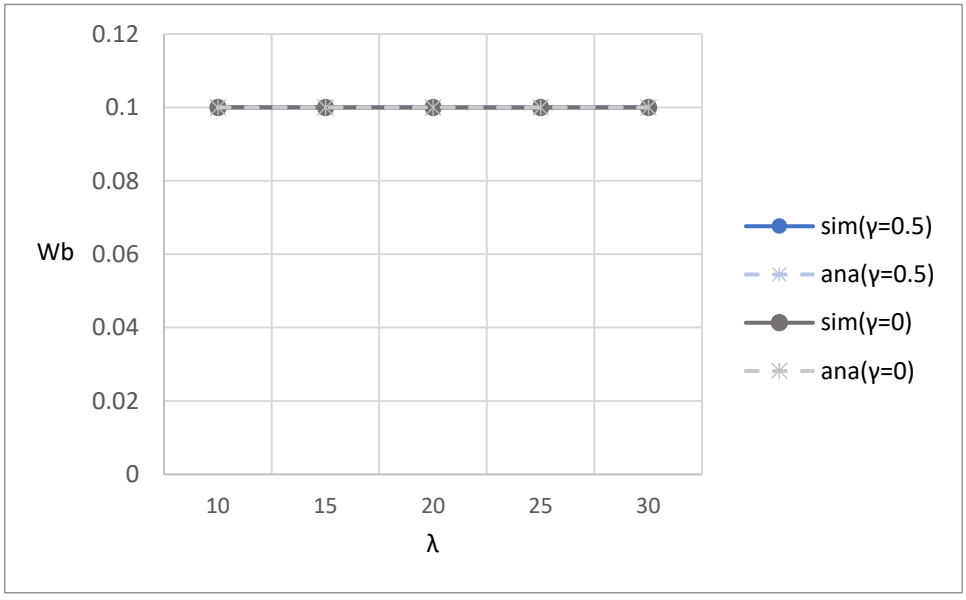


Figure 5-69: Effect of arrival rate on average waiting time in the block queue

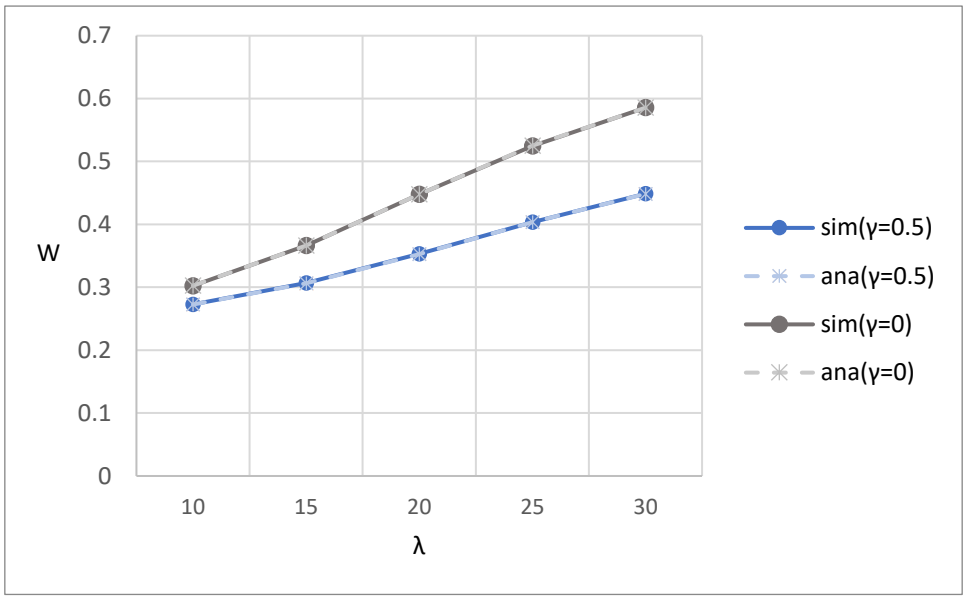


Figure 5-70: Effect of arrival rate on average waiting time in the system

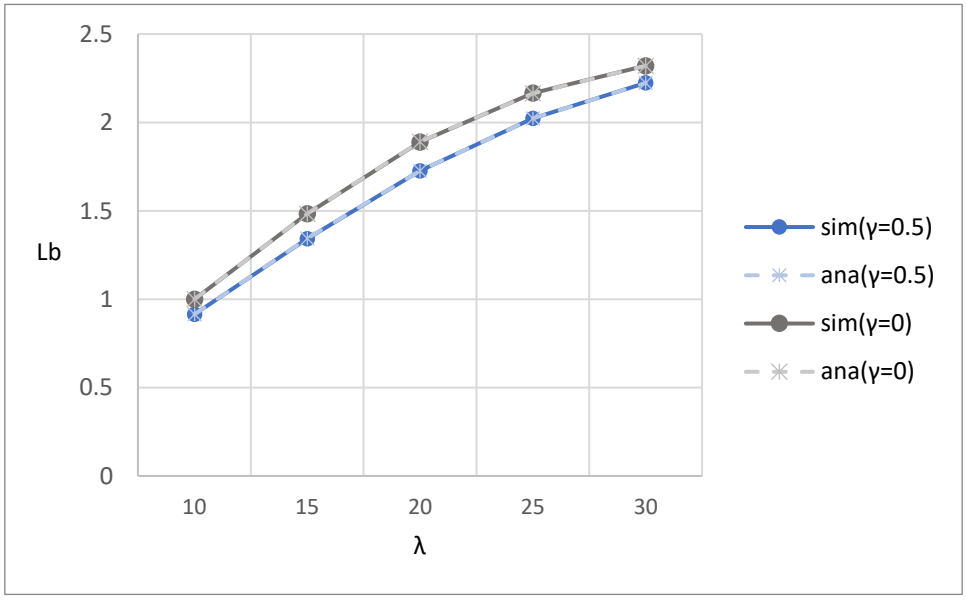


Figure 5-71: Effect of arrival rate on average number of customers in block queue

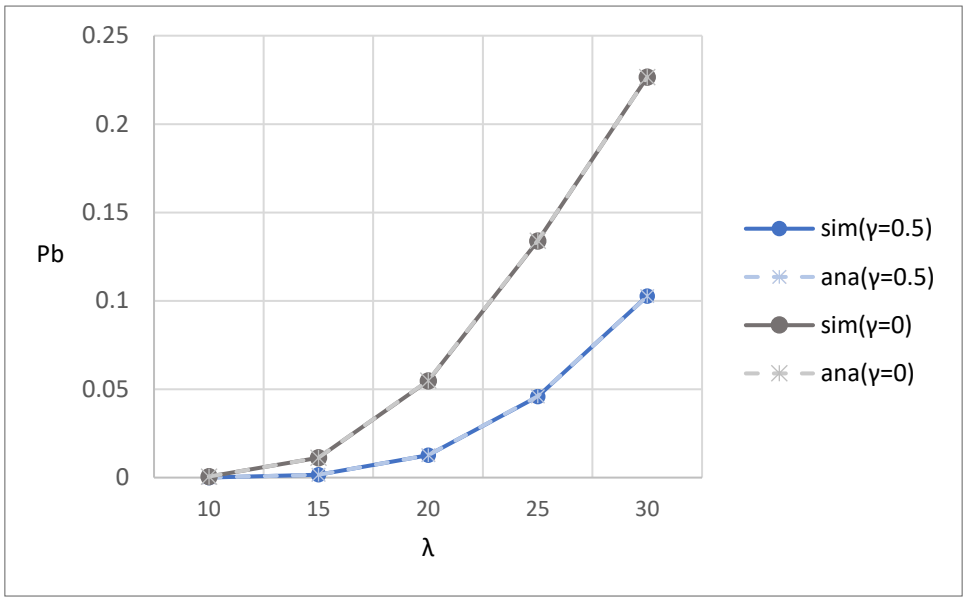


Figure 5-72: Effect of arrival rate on blocking probability

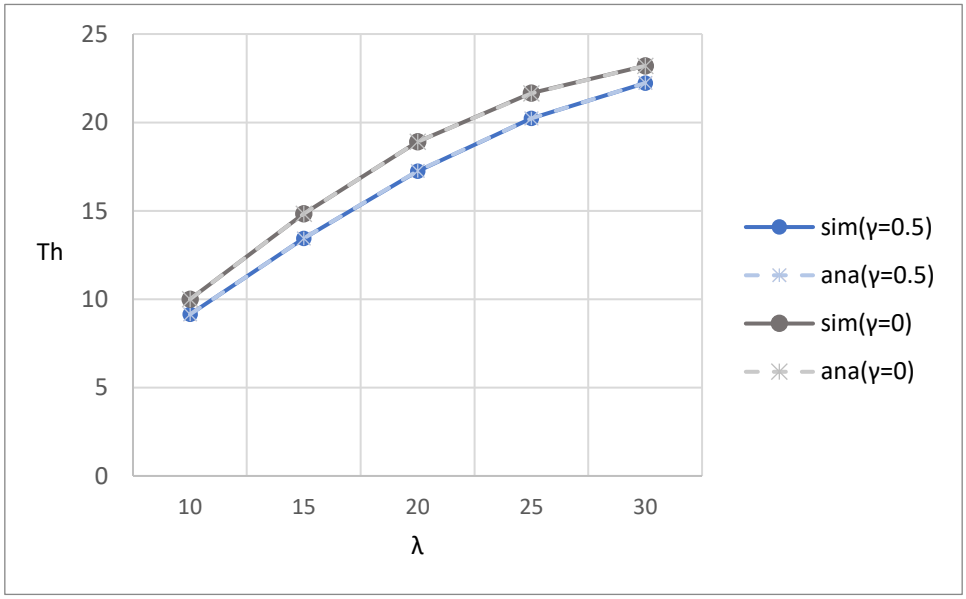


Figure 5-73: Effect of arrival rate on throughput

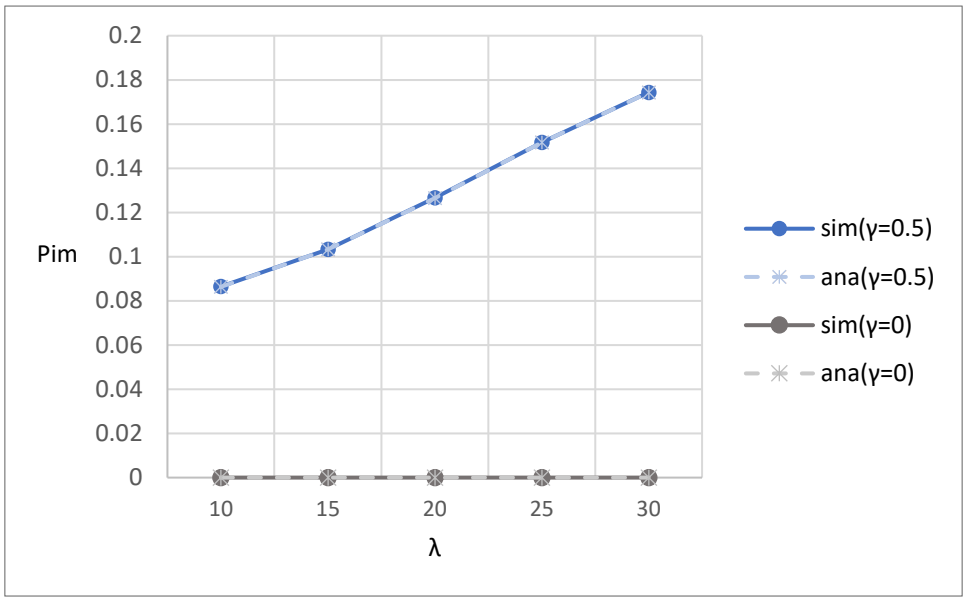


Figure 5-74: Effect of arrival rate on the impatient probability

### 5.3.3. Block generation rate

Figure 5-75 to Figure 5-81 show the relationship between various performance metrics and the block generation rate $\mu_1$. Both simulation results and analytical results are shown for comparison.

Figure 5-75 illustrates the impact of the block generation rate $\mu_1$ on the average waiting time in the customer queue ($W_c$). As $\mu_1$ increases, $W_c$ decreases significantly. This is because higher block generation rates allow customers to be grouped and processed more frequently, which reduces congestion in the customer queue. Additionally, $W_c$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which further suppresses queue buildup and contributes to the decline in $W_c$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-76 illustrates the impact of the block generation rate $\mu_1$ on the average waiting time in the block queue ($W_b$). As $\mu_1$ increases, $W_b$ remains nearly constant. This indicates that the time each block spends in the consensus queue is determined by the consensus rate and system transition rate, and is independent of the block generation rate. Furthermore, $W_b$ with impatience is the same as that without impatience. This is because once a block is formed, impatience mechanism has no effect on it. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-77 illustrates the impact of the block generation rate $\mu_1$ on the average waiting time in the system ($W$). As $\mu_1$ increases, $W$ decreases steadily. This is because higher block generation rates allow customers to be grouped and processed more frequently, which reduces congestion in the customer queue. Additionally, $W$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which further suppresses queue buildup and contributes to the decline in $W$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-78 illustrates the impact of the block generation rate $\mu_1$ on the average number of customers in the block queue ($L_b$). As $\mu_1$ increases, $L_b$ initially grows and then stabilizes around constant value. This indicates that although more frequent block generation can expedite customers service, the average block occupancy tends to saturate because of the arrival rate and the effective service rate reached a state of equilibrium. Additionally, $L_b$ with impatience is smaller than that without impatience.

This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which contributes to a reduction in $L_b$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-79 illustrates the impact of the block generation rate $\mu_1$ on the blocking probability ($P_b$). As $\mu_1$ increases, $P_b$ decreases rapidly. This is because higher block generation rates allow customers to be served more frequently, reducing the possibility that the customer queue reaches its capacity and causes incoming arrivals to be blocked. Additionally, $P_b$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which contributes to a reduction in $P_b$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-80 illustrates the impact of the block generation rate $\mu_1$ on the system throughput ($T_h$). As $\mu_1$ increases, $T_h$ rises rapidly at first and then gradually saturates. This is because higher block generation rates enable more customers to be processed per consensus cycle, but the throughput eventually approaches a limit determined by the customer arrival rate and the impatient rate, which is less than the system processing capacity. As a result, $T_h$ with impatience is smaller than that without impatience. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-81 illustrates the impact of the block generation rate $\mu_1$ on the impatience probability ($P_{im}$). As $\mu_1$ increases, $P_{im}$ gradually decreases. A higher block generation rate allows customers to be grouped into blocks and served more frequently, thereby shortening their time in the customer queue. This reduces the chance of reaching the impatience threshold before service. As a result, $P_{im}$ with impatience is larger than without impatience. Lastly, the analytical results are in good agreement with the simulation results.

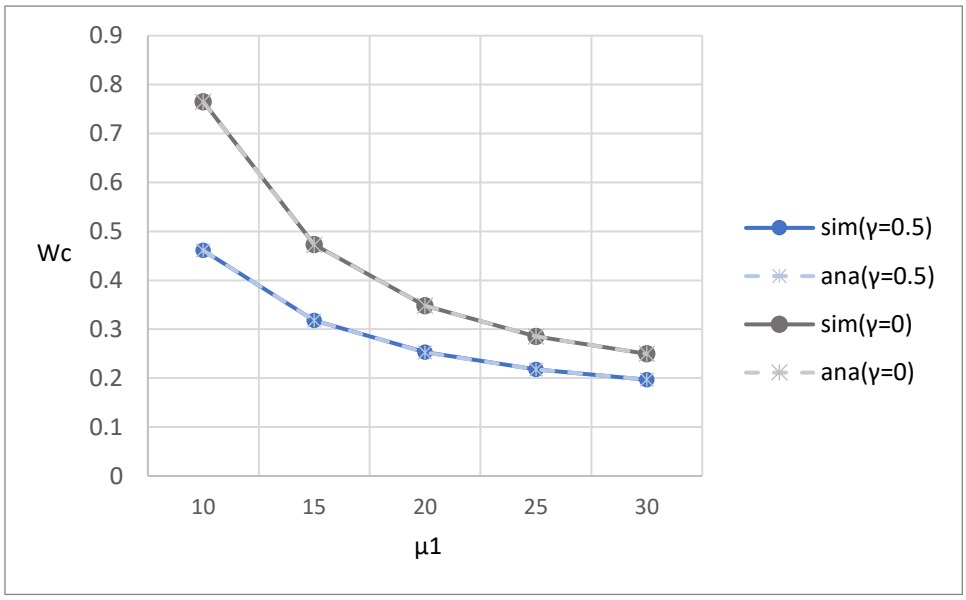


Figure 5-75: Effect of block generation rate on average waiting time in the customer queue

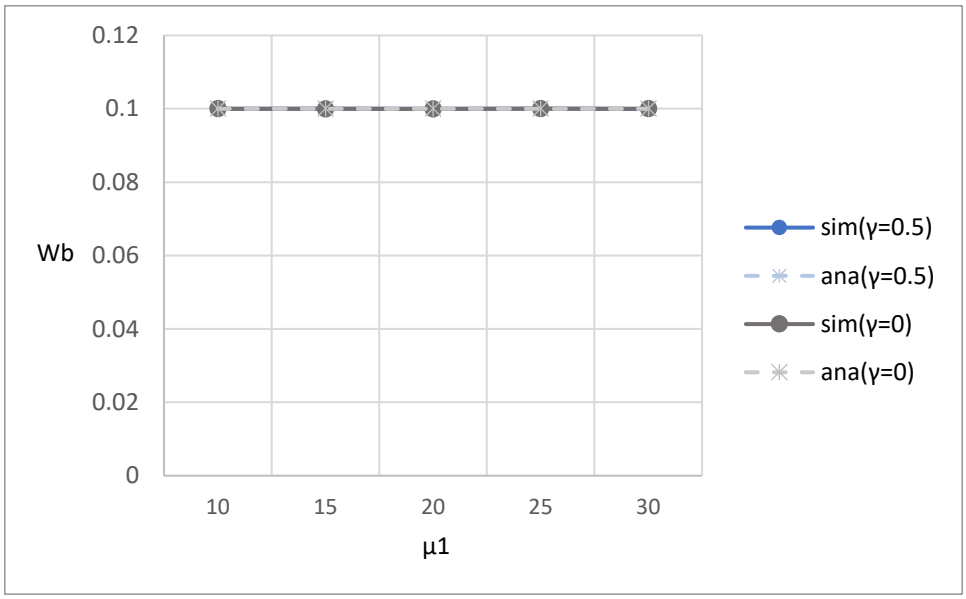


Figure 5-76: Effect of block generation rate on average waiting time in the block queue

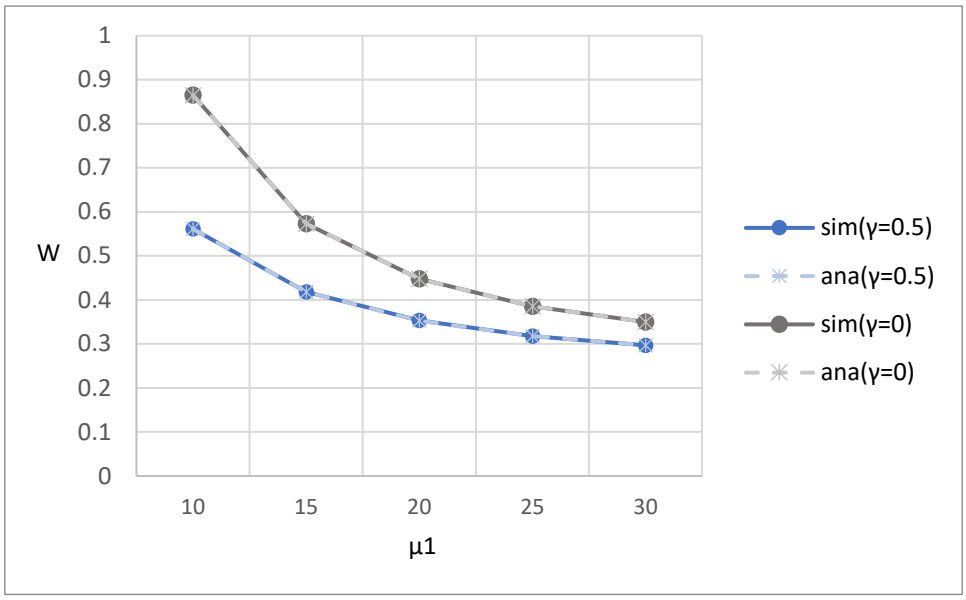


Figure 5-77: Effect of block generation rate on average waiting time in the system

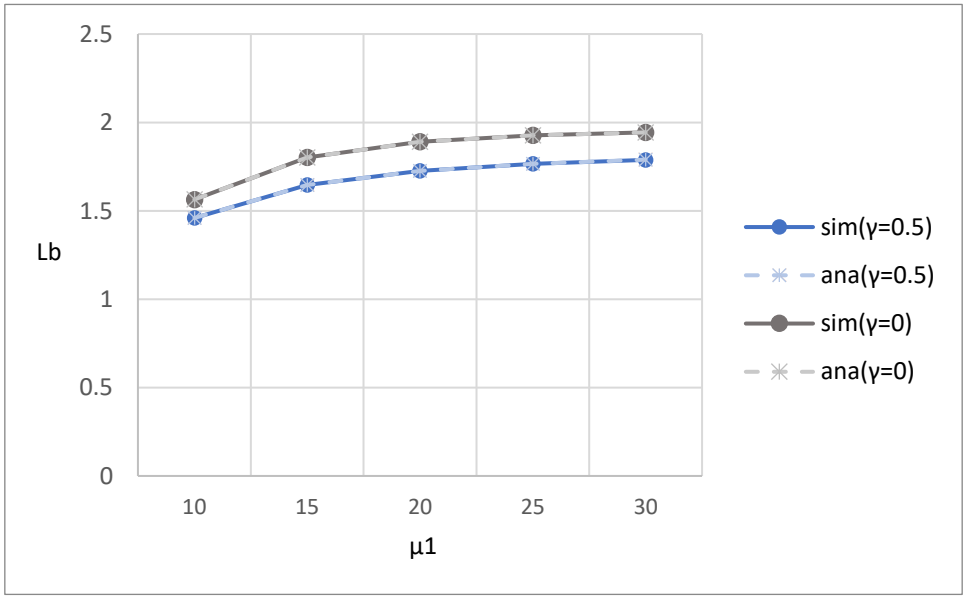


Figure 5-78: Effect of block generation rate on average number of customers in block queue

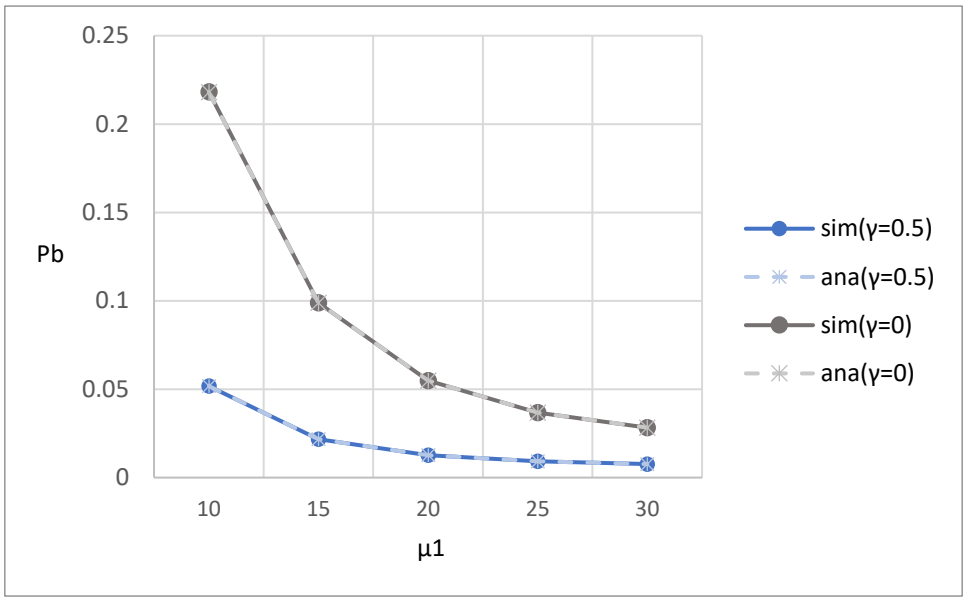


Figure 5-79: Effect of block generation rate on blocking probability

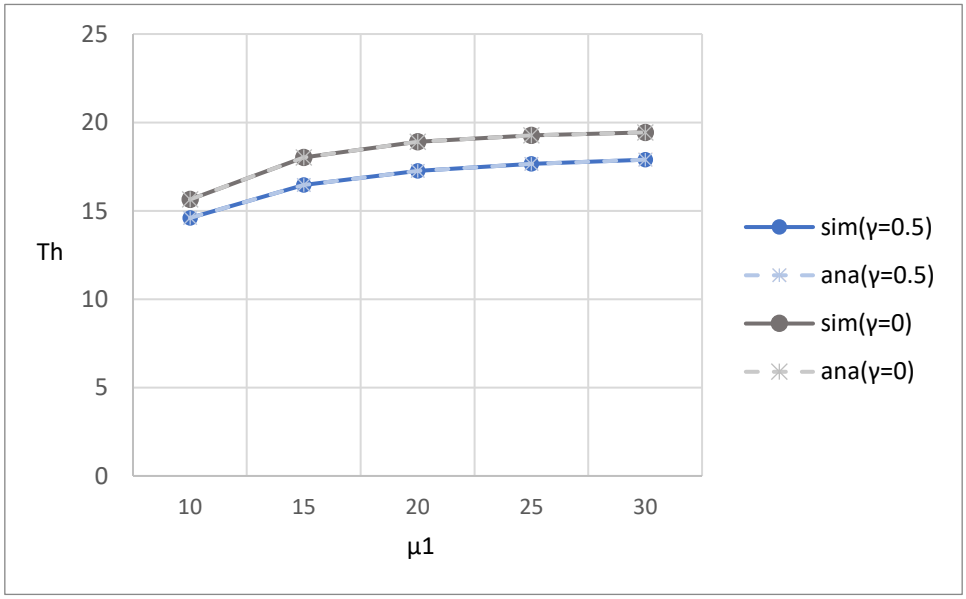


Figure 5-80: Effect of block generation rate on system throughput

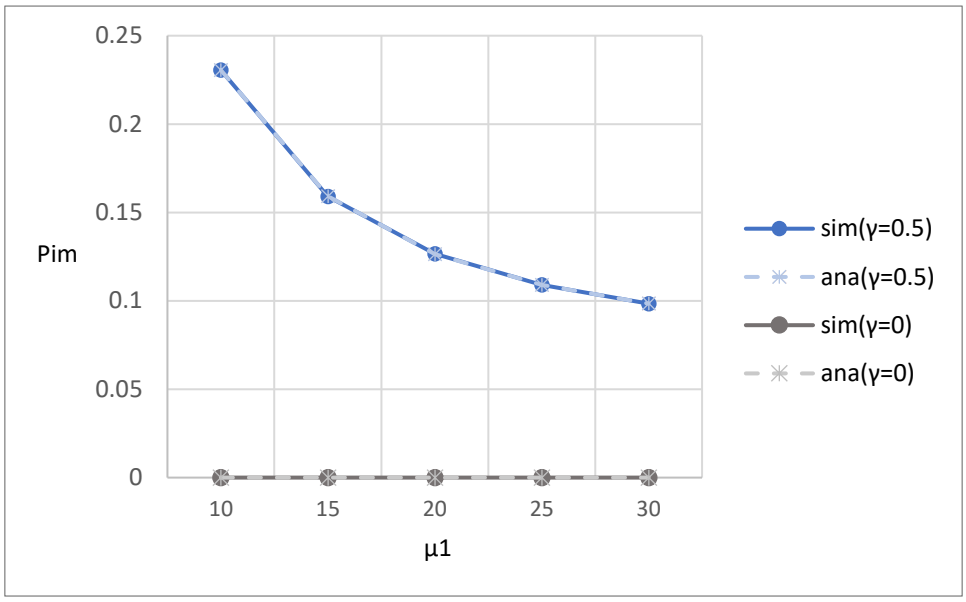


Figure 5-81: Effect of block generation on the impatient probability

### 5.3.4. Consensus rate

Figure 5-82 to Figure 5-88 show the relationship between various performance metrics and the consensus rate $\mu_2$. Both simulation results and analytical results are shown for comparison.

Figure 5-82 illustrates the impact of the consensus rate $\mu_2$ on the average waiting time in the customer queue ($W_c$). As $\mu_2$ increases, $W_c$ decrease steadily. This is because a higher consensus rate shortens the time block spend in the consensus queue, enabling faster turnover and more frequent admission of waiting customers from the customer queue. Additionally, $W_c$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which further suppresses queue buildup and contributes to the decline in $W_c$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-83 illustrates the impact of the consensus rate $\mu_2$ on the average waiting time in the block queue ($W_b$). As $\mu_2$ increases, $W_b$ decrease significantly. This is because a faster consensus rate allows blocks to be processed more quickly, thereby reducing the time that blocks spend waiting in the queue. Furthermore, $W_b$ with impatience is the same as that without impatience. This is because once a block is formed, impatience mechanism has no effect on it. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-84 illustrates the impact of the consensus rate $\mu_2$ on the average waiting time in the system ($W$). As $\mu_2$ increases, $W$ decrease significantly. This is because faster consensus processing reduces delays in the block queue, which in turn accelerates the overall service flow and shortens the total time customers spend in the system. Additionally, $W$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which further suppresses queue buildup and contributes to the decline in $W$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-85 illustrates the impact of the consensus rate $\mu_2$ on the average number of customers in the block queue ($L_b$). As $\mu_2$ increases, $L_b$ decrease significantly. This is because a higher consensus rate enables faster processing of blocks, which reduces queue accumulation and lowers the $L_b$. Additionally, $L_b$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes

customers to leave the queue once their impatience threshold is reached, which contributes to a reduction in $L_b$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-86 illustrates the impact of the consensus rate $\mu_2$ on the blocking probability ($P_b$). As $\mu_2$ increases, $P_b$ decrease rapidly. This is because faster consensus rate enables faster processing of blocks, and lowers the probability of reaching the queue capacity that triggers blocking. Additionally, $P_b$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which contributes to a reduction in $P_b$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-87 illustrates the impact of the consensus rate $\mu_2$ on the system throughput ($T_h$). As $\mu_2$ increases, $T_h$ increase rapidly at first and then gradually saturates. This is because higher consensus rate allows blocks to be processed more efficiently, but the throughput eventually approaches a limit determined by the customer arrival rate and the impatient rate, which is less than the system processing capacity. As a result, $T_h$ with impatience is smaller than that without impatience. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-88 illustrates the impact of the consensus rate $\mu_2$ on the system impatient probability ($P_{im}$). As $\mu_2$ increases, steadily decreases. A higher consensus rate makes the consensus process more quickly, thereby lowering the probability of customers in customer queue reaching their impatience threshold. As a result, $P_{im}$ with impatience is larger than without impatience. Lastly, the analytical results are in good agreement with the simulation results.

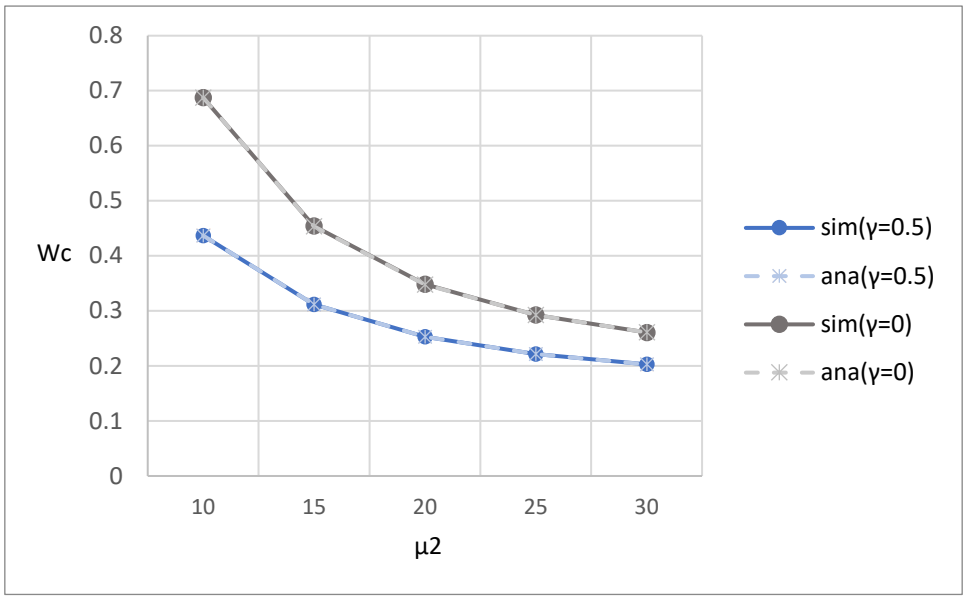


Figure 5-82: Effect of consensus rate on average waiting time in the customer queue

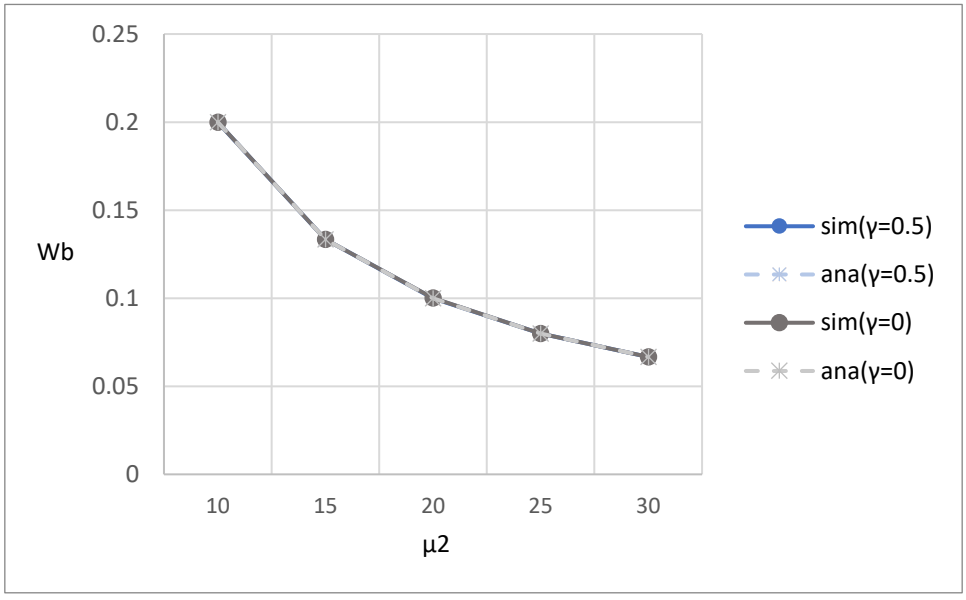


Figure 5-83: Effect of consensus rate on average waiting time in the block queue

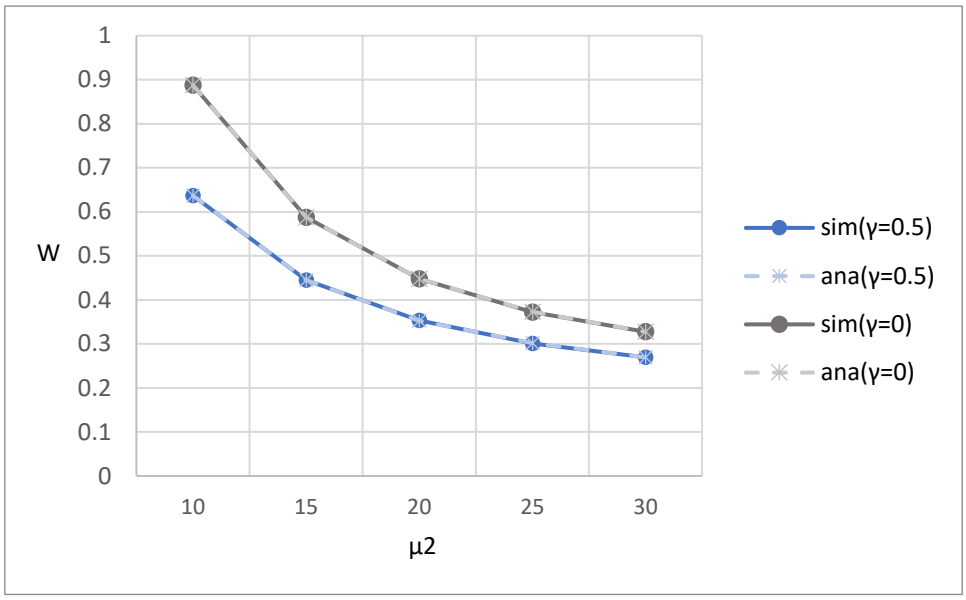


Figure 5-84: Effect of consensus rate on average waiting time in the system

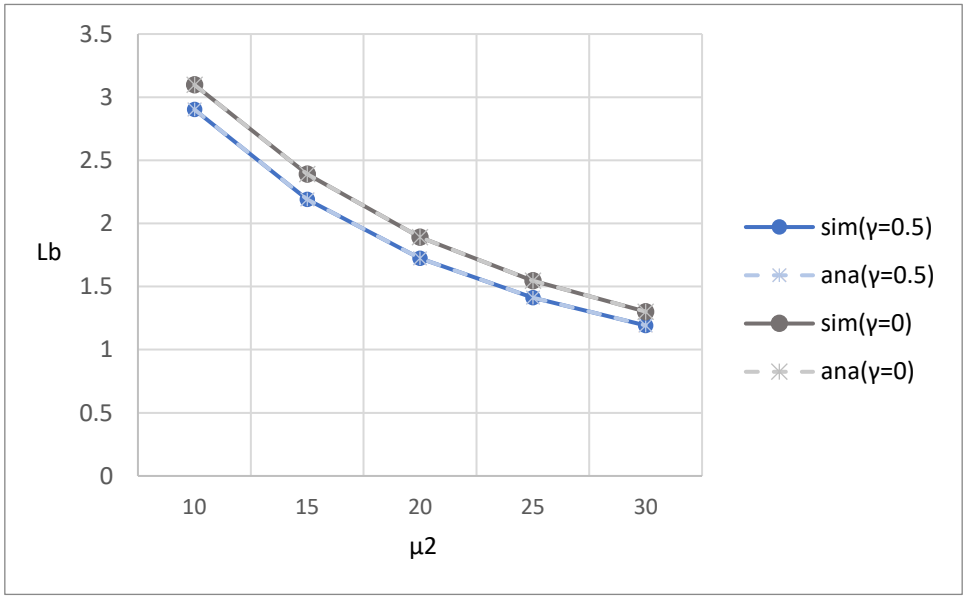


Figure 5-85: Effect of consensus rate on average number of customers in block queue

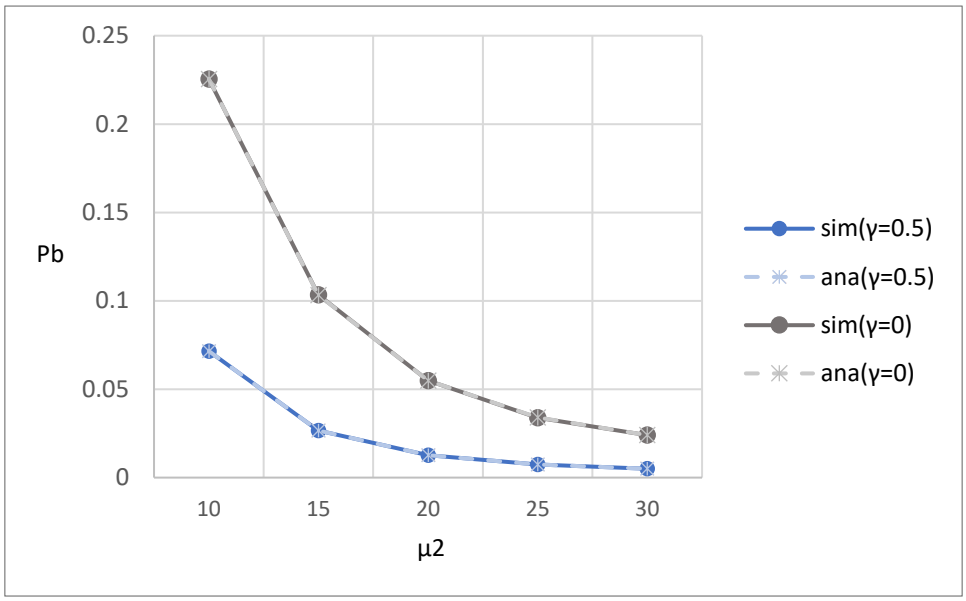


Figure 5-86: Effect of consensus rate on blocking probability

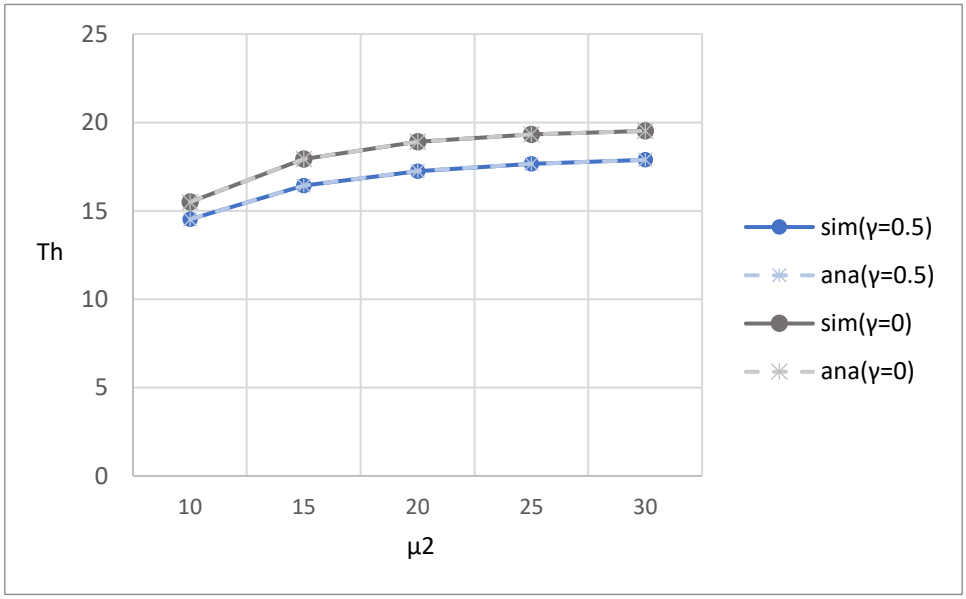


Figure 5-87: Effect of consensus rate on system throughput

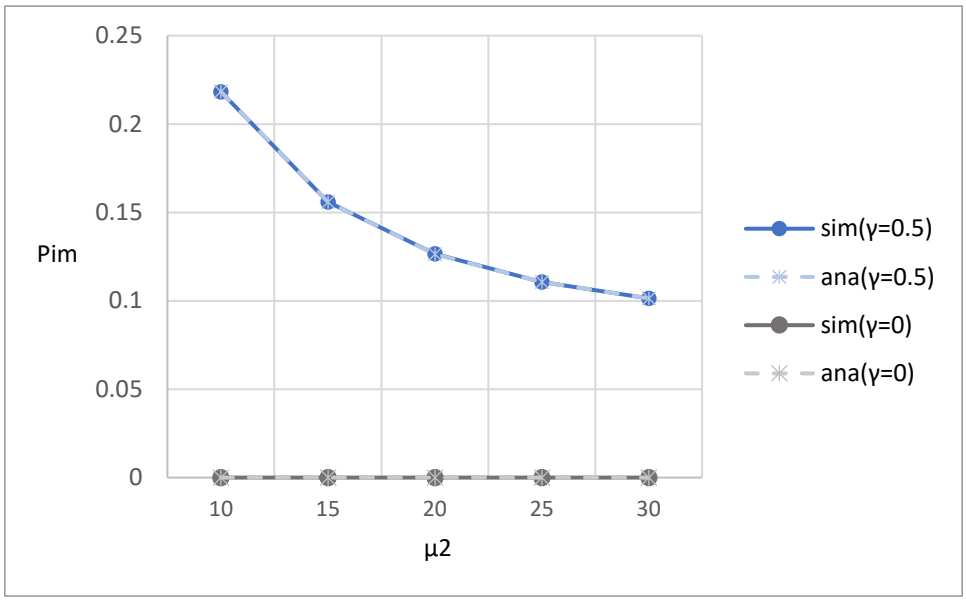


Figure 5-88: Effect of consensus rate on the impatient probability

### 5.3.5. Transition rate (from ON to OFF)

Figure 5-89 to Figure 5-95Figure 5-94 show the relationship between various performance metrics and the transition rate from ON to OFF $\alpha$. Both simulation results and analytical results are shown for comparison.

Figure 5-89 illustrates the impact of the transition rate $\alpha$ on the average waiting time in the customer queue ($W_c$). As $\alpha$ increases, $W_c$ increases steadily. This is because more frequent transitions from ON to OFF reduce the availability of block generation service, causing longer queueing delays for arriving customers. Additionally, $W_c$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which further suppresses queue buildup and contributes to the decline in $W_c$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-90 illustrates the impact of the transition rate $\alpha$ on the average waiting time in the block queue ($W_b$). As $\alpha$ increases, $W_b$ increase steadily. This is because more frequent service interruptions delay consensus processing, resulting in longer waiting times for customer(s) in block queue. Furthermore, $W_b$ with impatience is the same as that without impatience. This is because once a block is formed, impatience mechanism has no effect on it. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-91 illustrates the impact of the transition rate $\alpha$ on the average waiting time in the system ($W$). As $\alpha$ increases, $W$ increase steadily. This is because more frequent service interruptions caused by transitions to the OFF state reduce overall availability of block generation and consensus service, leading to longer queueing delays for customers in the system. Additionally, $W$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which further suppresses queue buildup and contributes to the decline in $W$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-92 illustrates the impact of the transition rate $\alpha$ on the average number of customers in the block queue ($L_b$). As $\alpha$ increases, $L_b$ increases steadily. This is because more frequent service interruptions of block generation processing cause more customers to wait in the customer queue while forming batches from the customer queue. Additionally, $L_b$ with impatience is smaller than that without impatience. This

is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which contributes to a reduction in $L_b$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-93 illustrates the impact of the transition rate $\alpha$ on the blocking probability ($P_b$). As $\alpha$ increases, $P_b$ increases steadily. This is because more frequent service interruptions reduce the system's capacity to process customers, which increases the probability that the customer queue reaches its capacity and causes incoming arrivals to be blocked. Additionally, $P_b$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which contributes to a reduction in $P_b$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-94 illustrates the impact of the transition rate $\alpha$ on the system throughput ($T_h$). As $\alpha$ increases, $T_h$ decreases gradually. This is because more frequent service interruptions reduce the chance for block generation and consensus processing. In addition, due to the impatience mechanism, customers are more likely to abandon the queue when waiting becomes longer, which contributes to a decline in $T_h$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-95 illustrates the impact of the transition rate $\alpha$ on the impatience probability ($P_{im}$). As $\alpha$ increases, $P_{im}$ increases steadily. This is because more frequent system on the OFF state which limits service availability, causing customers to wait longer in the queue and thus increasing the probability of reaching their impatience threshold. As a result, $P_{im}$ with impatience is larger than without impatience. Lastly, the analytical results are in good agreement with the simulation results.

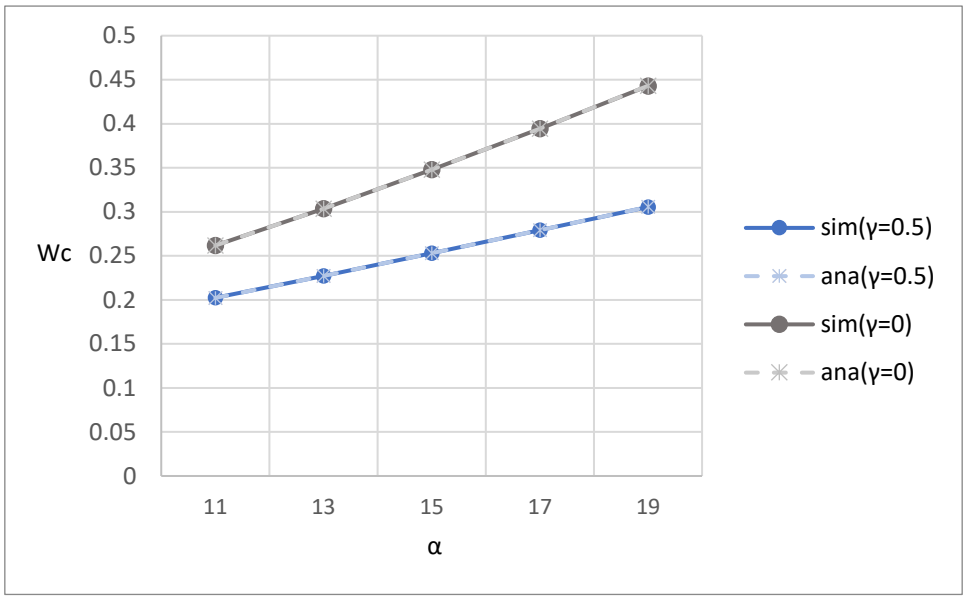


Figure 5-89: Effect of transition rate on average waiting time in the customer queue in the system

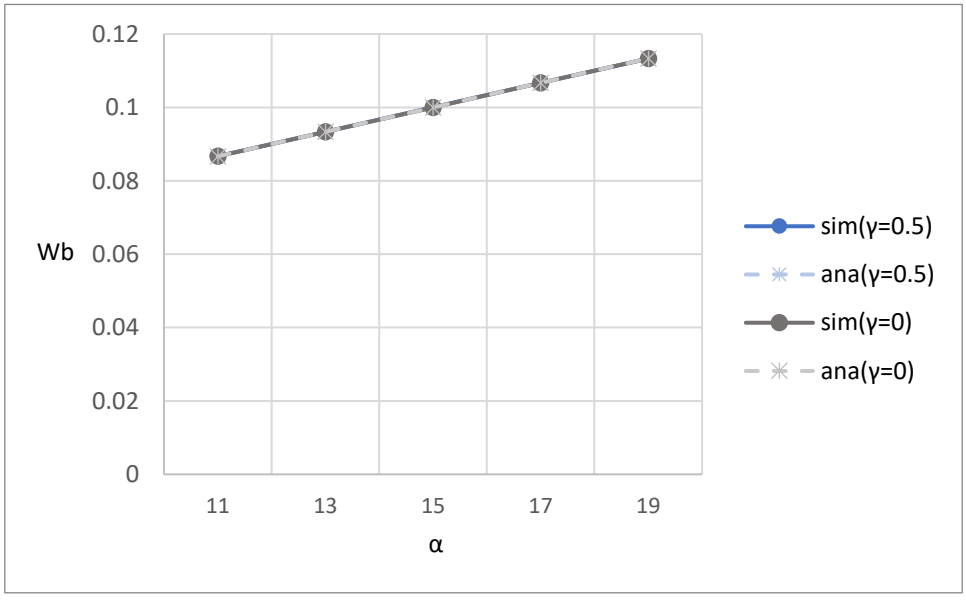


Figure 5-90: Effect of transition rate on average waiting time in the block queue

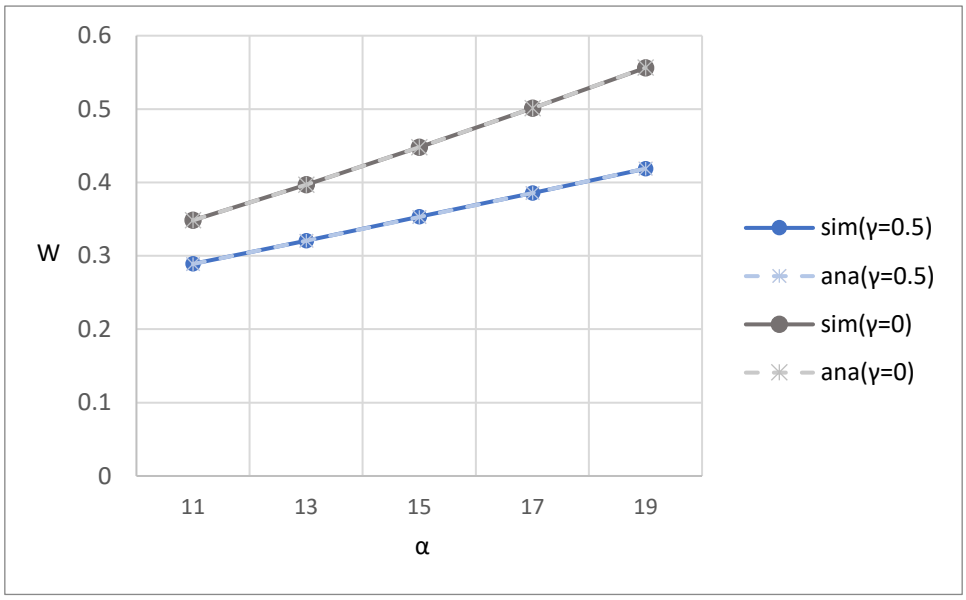


Figure 5-91: Effect of transition rate on average waiting time in the system

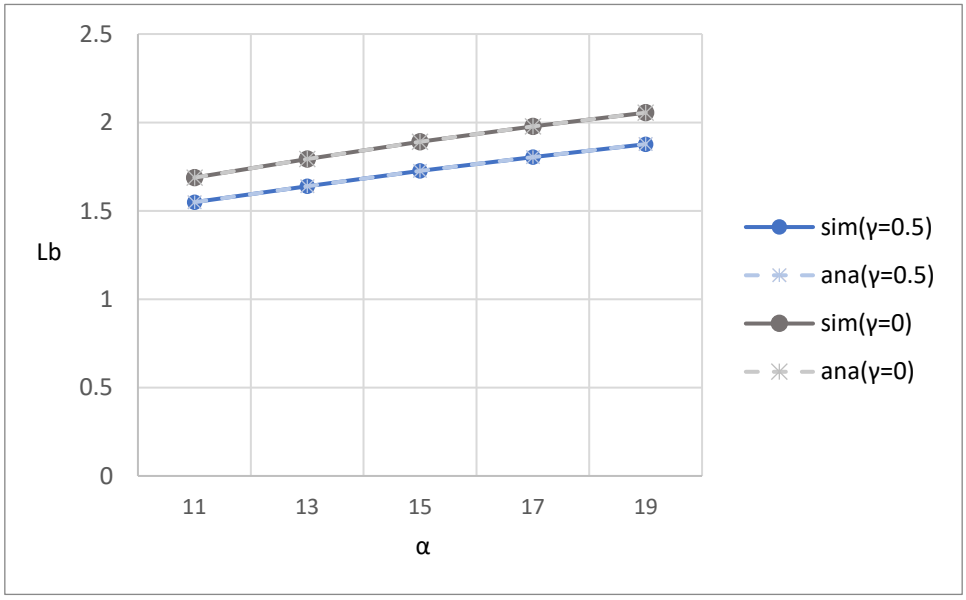


Figure 5-92: Effect of transition rate on average number of customers in block queue

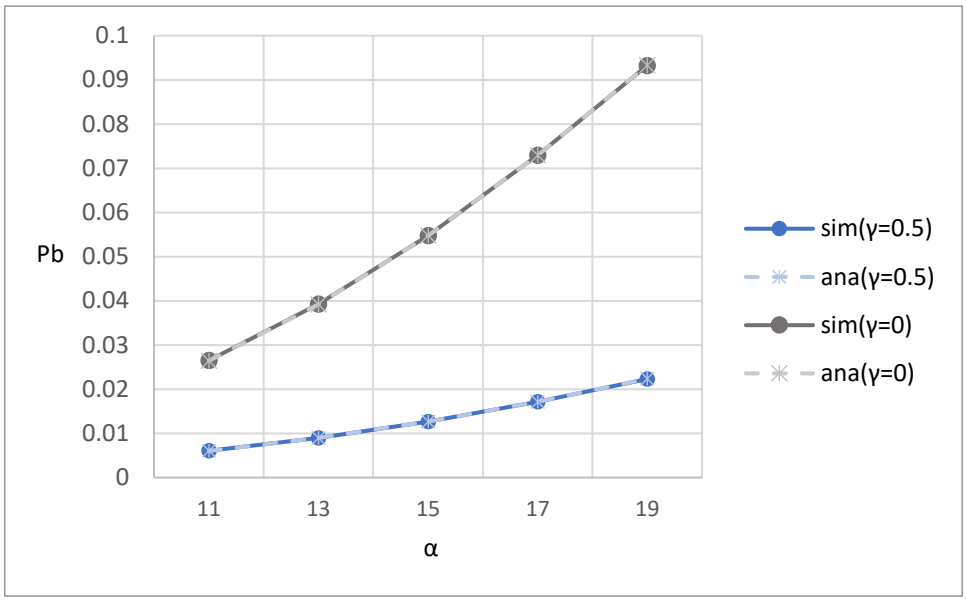


Figure 5-93: Effect of transition rate on blocking probability

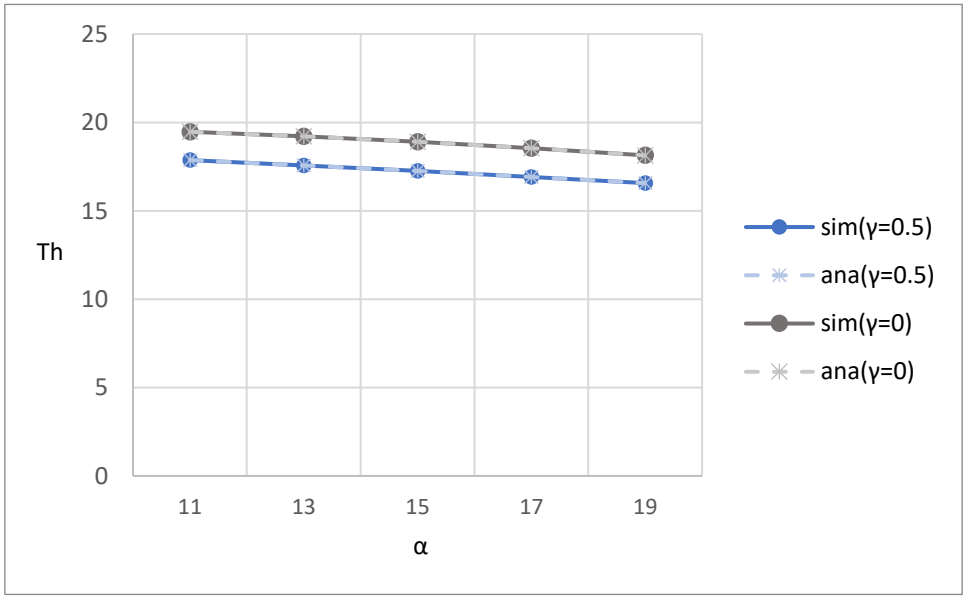


Figure 5-94: Effect of transition rate on system throughput

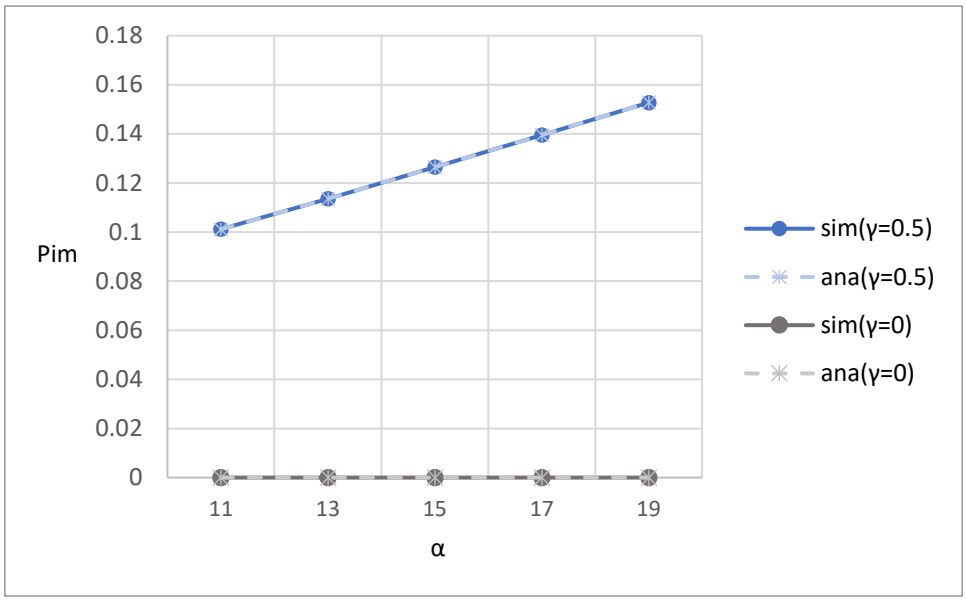


Figure 5-95: Effect of transition rate on the impatient probability

### 5.3.6. Impatient rate

Figure 5-96 to Figure 5-102 show the relationship between various performance metrics and the impatient rate $\gamma$. Both simulation results and analytical results are shown for comparison.

Figure 5-96 illustrates the impact of the impatience rate $\gamma$ on the average waiting time in the customer queue ($W_c$). As $\gamma$ increases, $W_c$ gradually decreases. This is because higher impatience rate makes customers more likely to abandon the customer queue once they reach their impatience threshold, thereby shortening the overall queue length and reducing the average waiting time in customer queue. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-97 illustrates the impact of the impatience rate $\gamma$ on the average waiting time in the block queue ($W_b$). As $\gamma$ increases, $W_b$ remains nearly constant. This is because the impatience mechanism only affects customers while they are in the customer queue. Once a block is formed, they are no longer subject to abandonment due to impatience. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-98 illustrates the impact of the impatience rate $\gamma$ on the average waiting time in the system ($W$). As $\gamma$ increases, $W$ steadily decreases. This is because higher impatience rate makes customers more likely abandon the customer queue once they reach their impatience threshold, which reduces congestion and lowers the overall time customers spend in the system. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-99 illustrates the impact of the impatience rate $\gamma$ on the average number of customers in the block queue ($L_b$). As $\gamma$ increases, $L_b$ steadily decreases. This is because higher impatience rate makes customers more likely to abandon the customer queue once they reach their impatience threshold, reducing the number of customers that eventually form a block. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-100 illustrates the impact of the impatience rate $\gamma$ on the blocking probability ($P_b$). As $\gamma$ increases, $P_b$ gradually decreases. This is because higher impatience rate makes customers more likely abandon the customer queue once they reach their impatience threshold, thereby reducing the probability that queue reaches its capacity and triggers blocking. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-101 illustrates the impact of the impatience rate $\gamma$ on the system throughput ($T_h$). As $\gamma$ increases, $T_h$ decreases gradually. This is because higher impatience rate makes customers more likely abandon the customer queue once they reach their impatience threshold, which reduces the total number of customers successfully processed by the system. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-102 illustrates the impact of the impatience rate $\gamma$ on the impatience probability ($P_{im}$). As $\gamma$ increases, $P_{im}$ increases significantly. This is because higher impatience rate makes customers more likely abandon the customer queue once they reach their impatience threshold. As a result, the proportion of customers abandoning the system increases, leading to a higher impatience probability. Lastly, the analytical results are in good agreement with the simulation results.

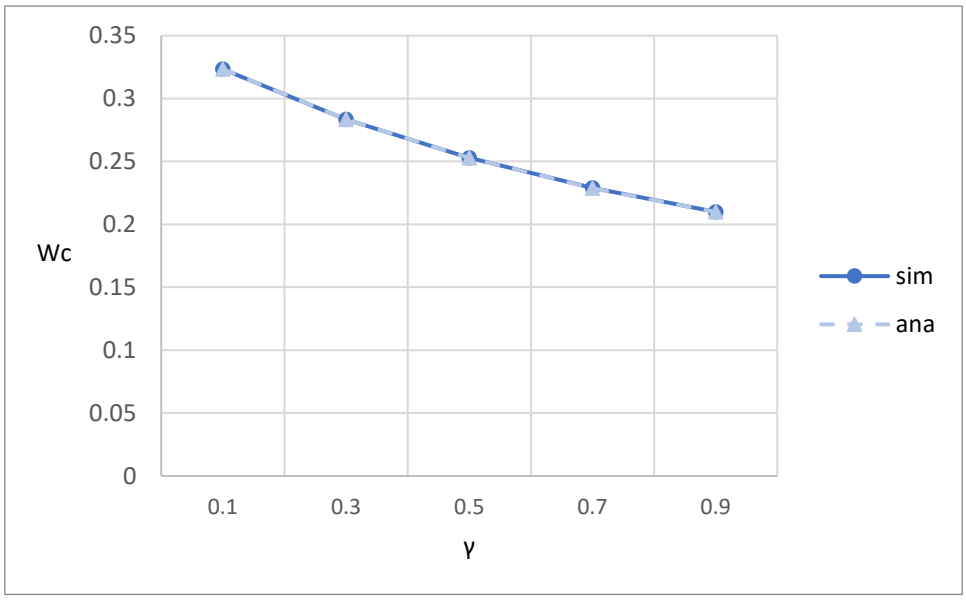


Figure 5-96: Effect of transition rate on average waiting time in the customer queue in the system

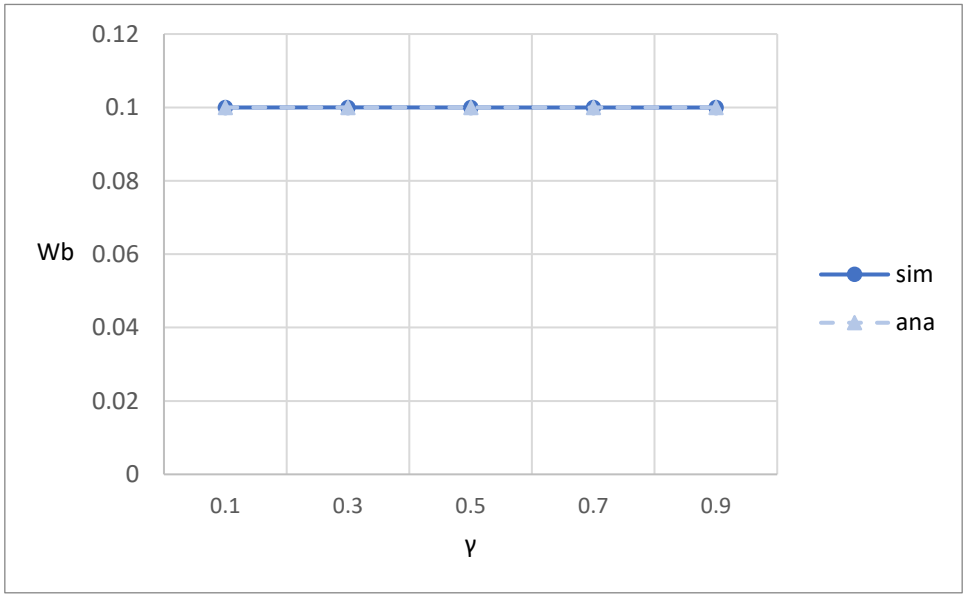


Figure 5-97: Effect of transition rate on average waiting time in the block queue

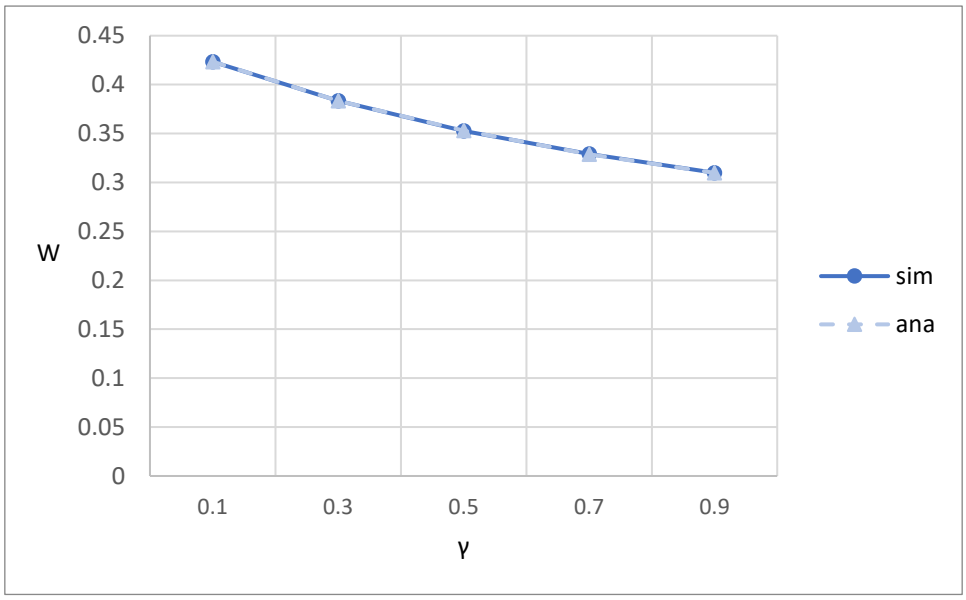


Figure 5-98: Effect of transition rate on average waiting time in the system

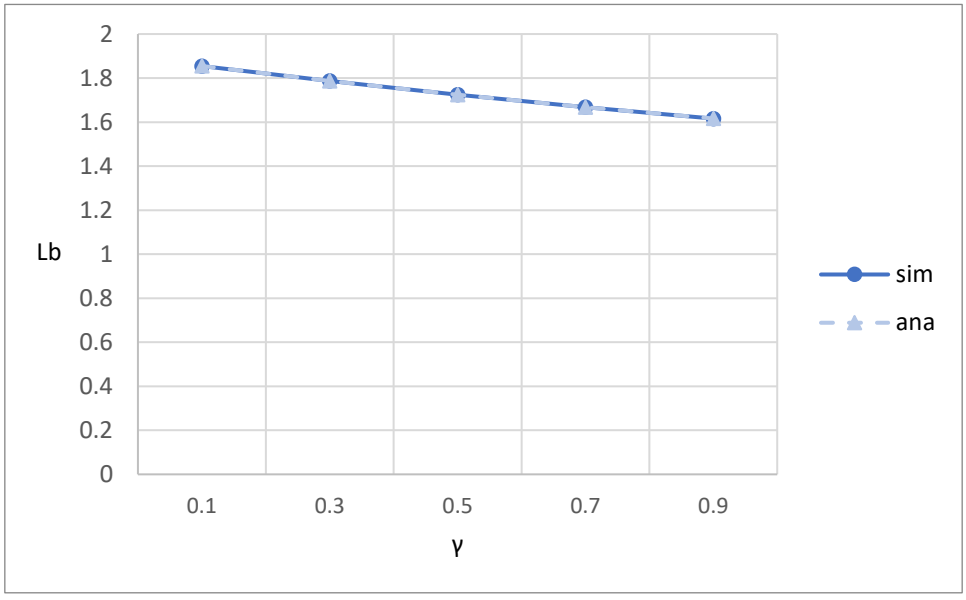


Figure 5-99: Effect of transition rate on average number of customers in block queue

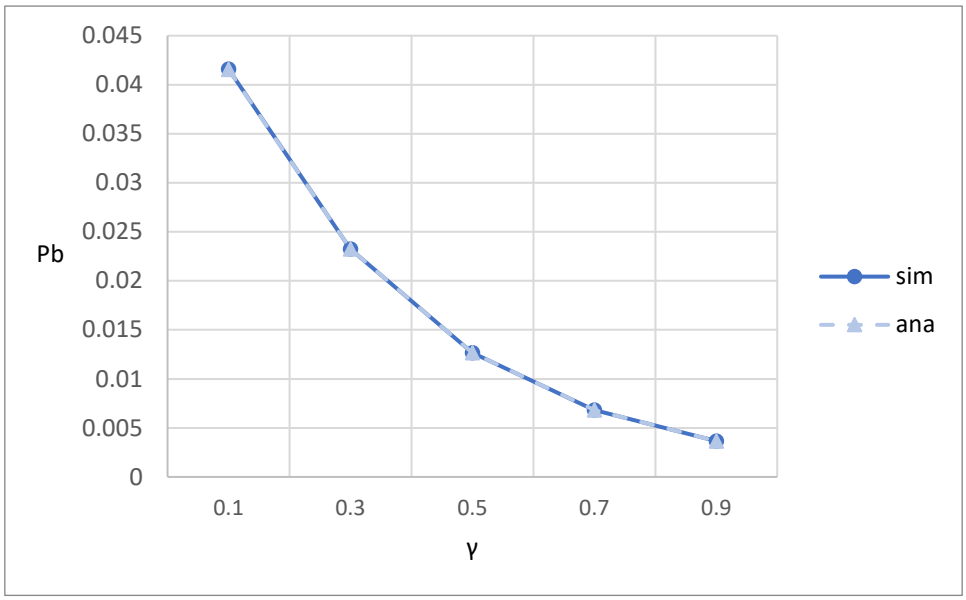


Figure 5-100: Effect of transition rate on blocking probability

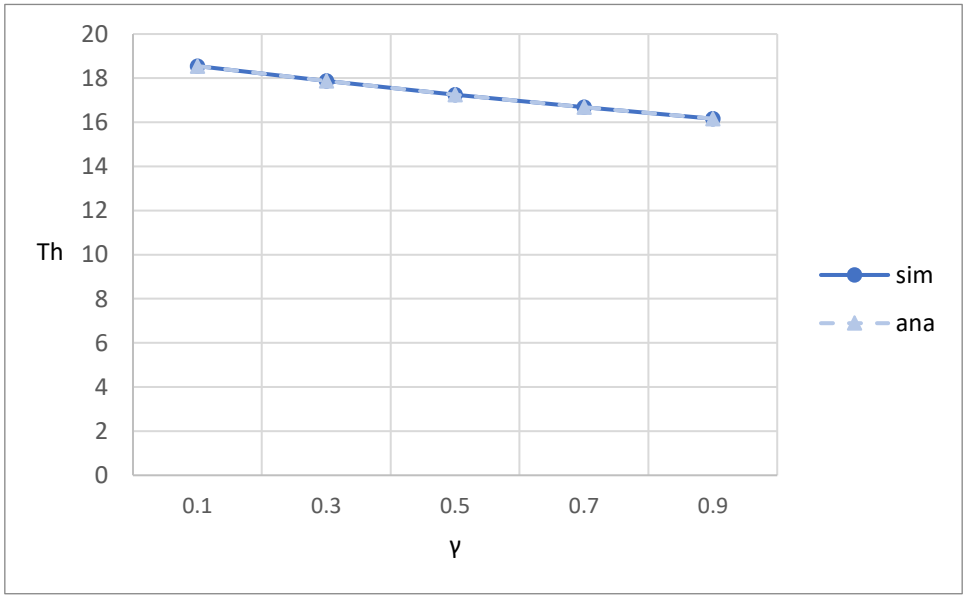


Figure 5-101: Effect of transition rate on system throughput

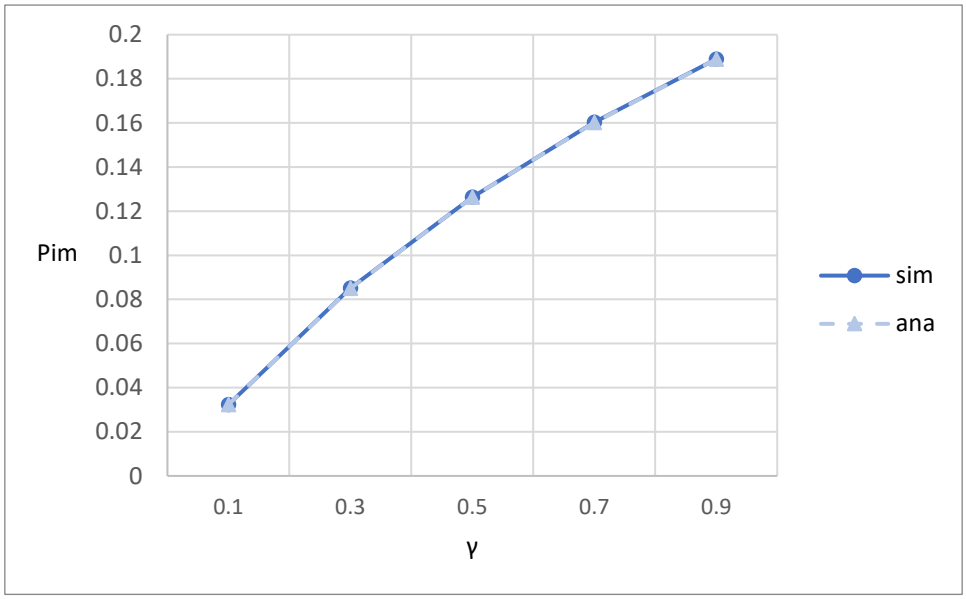


Figure 5-102: Effect of transition rate on the impatient probability

## 5.4. Scenario 4: Two-Class Customer with Impatience

The default values are as provided as below: $\lambda_H = 5$, $\lambda_L = 15$, $\mu_{1_H} = 20$, $\mu_{1_L} = 20$, $\mu_{2_H} = 25$, $\mu_{2_L} = 20$, $\alpha = \beta = 15$, $\gamma_H = 1$, $\gamma_L = 0.5$. The maximum capacity of the system is $N = 20$, and the maximum block size is $b = 5$.

### 5.4.1. Block size

Figure 5-103 to Figure 5-108 show the relationship between various performance metrics and the block size $b$. Both simulation results and analytical results are shown for comparison.

Figure 5-103 illustrates the impact of the block size $b$ on the average waiting times in the customer queue for high-priority, low-priority, and overall customers. As $b$ increases, the $W_c$ decreases. The reduction is more pronounced for $W_{c_L}$, while $W_{c_H}$ remain consistently low. This is because larger blocks allow more customers to be served per service cycle, thereby reducing the time, especially for low-priority customers who tend to experience longer delays when $b$ is small. In addition, the $W_{c_H}$ is much smaller than $W_{c_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue. Furthermore, $W_c$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which further suppresses queue buildup and contributes to the decline in $W_c$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-104 illustrates the impact of the block size $b$ on the average waiting times in the block queue for high-priority, low-priority, and overall customers. As $b$ increases, the average waiting time in the block queue remains nearly constant for all priority levels. This indicates that the time each block spends in the consensus queue is determined by the consensus rate and system state transition rate, and is independent of block size. In addition, the $W_{b_H}$ is smaller than $W_{b_L}$. This is because $\mu_{2_H}$ is larger than $\mu_{2_L}$. Furthermore, $W_b$ with impatience is the same as that without impatience. This is because once a block is formed, impatience mechanism has no effect on it. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-105 illustrates the impact of the block size $b$ on the average waiting times in the system for high-priority, low-priority, and overall customers. As $b$ increases, the $W$ decreases. The decline is especially significant for low-priority customers, while the $W_H$ remains relatively constant. This is because larger blocks

allow more customers to be served per service cycle, which benefits low-priority customers who are otherwise delayed by the non-preemptive priority mechanism. In addition, the $W_H$ is much smaller than $W_L$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue and $\mu_{2_H}$ is larger than $\mu_{2_L}$. Furthermore, $W$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which further suppresses queue buildup and contributes to the decline in $W$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-106 illustrates the impact of the block size $b$ on the average numbers of customers in the block queue for high-priority, low- priority, and overall customers. As $b$ increases, the average number of customers in the block queue rises gradually across all priority levels. The increase is most noticeable for $L_b$ and $L_{b_L}$, while the $L_{b_H}$ remains relatively low and stable. This indicates that although larger blocks permit more customers per batch, the average block occupancy tends to saturate when the customer arrival rate is equal to the effective service rate of the customer queue. In addition, $L_{b_H}$ is smaller than $L_{b_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue and the high-priority customers have faster consensus rate than low-priority customers in the block queue, and therefore more low-priority customers remain waiting in the customer queue before being batched. Additionally, $L_b$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which contributes to a reduction in $L_b$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-107 illustrates the impact of the block size $b$ on the blocking probabilities for high-priority, low- priority, and overall customers. As $b$ increases, the blocking probability decreases across all priority levels. The decline is more pronounced for $P_{b_L}$, which is initially much higher and drops significantly with increasing $b$. This is because larger blocks allow more customers to be served per block generation cycle, thereby reducing the chance of the customer queue reaching its capacity limit, especially for low-priority customers who are more likely to be blocked under limited queue capacity. In addition, $P_{b_H}$ is smaller than $P_{b_L}$. This is because the capacity limit of the customer queue for the high-priority customers is no smaller than that for low-priority customers. Additionally, $P_b$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which contributes to a

reduction in $P_b$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-108 illustrates the impact of the block size $b$ on the system throughputs for high-priority, low-priority, and overall customers. As $b$ increases, the system throughput increases across all priority levels and then gradually saturates. Both $T_{h_H}$ and $T_{h_L}$ increase with $b$, with the growth being more significant for low-priority customers. This is because larger blocks enable more customers to be processed per consensus cycle. However, the throughput eventually approaches a limit determined by the customer arrival rate and the impatient rate, which is less than the system processing capacity. In addition, $T_{h_H}$ is smaller than $T_{h_L}$. This is because the customer arrival rate of the high-priority customers is smaller than that of low-priority customers. As a result, $T_h$ with impatience is smaller than that without impatience. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-109 illustrates the impact of the block size $b$ on the impatient probabilities for high-priority, low- priority, and overall customers. As $b$ increases, the impatient probability across all priority levels. This is because larger blocks allow more customers to be served in each service cycle, which reduces the waiting time in the customer queue. As a result, the probability that customers reach their impatience threshold and leave the system becomes lower. In addition, $P_{im_L}$ is higher than $P_{im_H}$, as low-priority customers are more likely to experience longer waits due to the non-preemptive priority discipline. As described above, $P_{im}$ with impatience is larger than without impatience. Lastly, the analytical results are in good agreement with the simulation results.

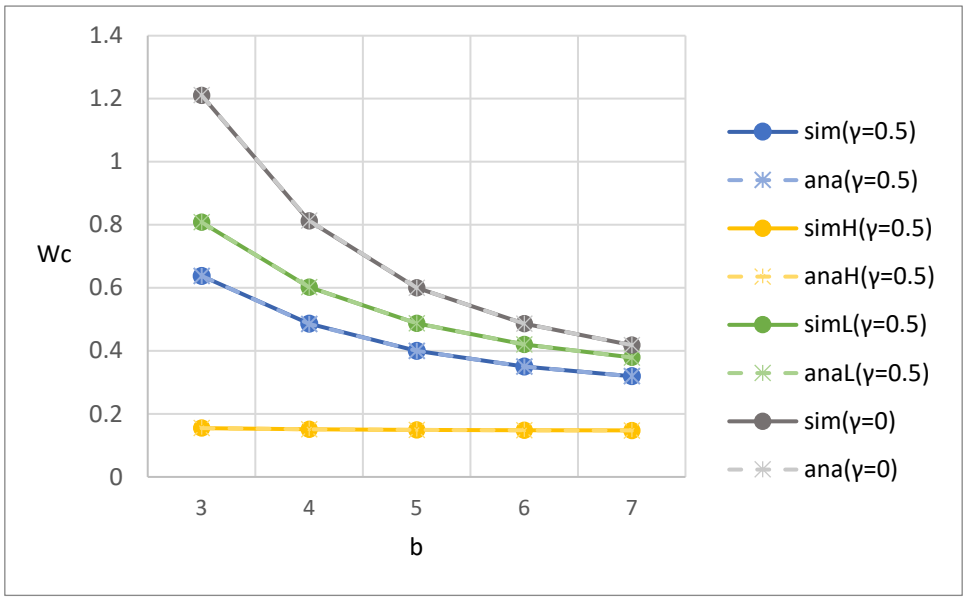


Figure 5-103: Effect of block size on average waiting time in the customer queue

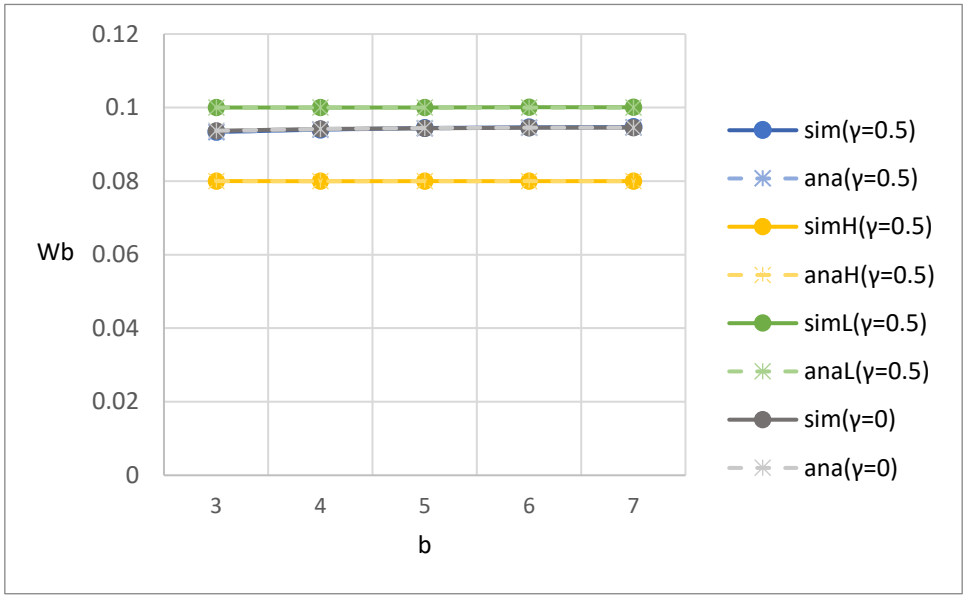


Figure 5-104: Effect of block size on average waiting time in the block queue

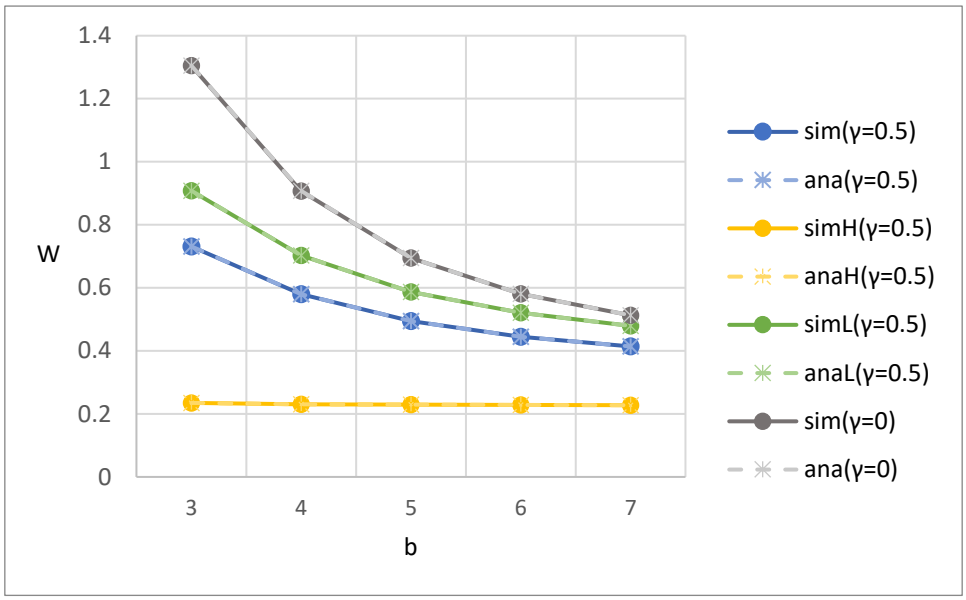


Figure 5-105: Effect of block size on average waiting time in the system

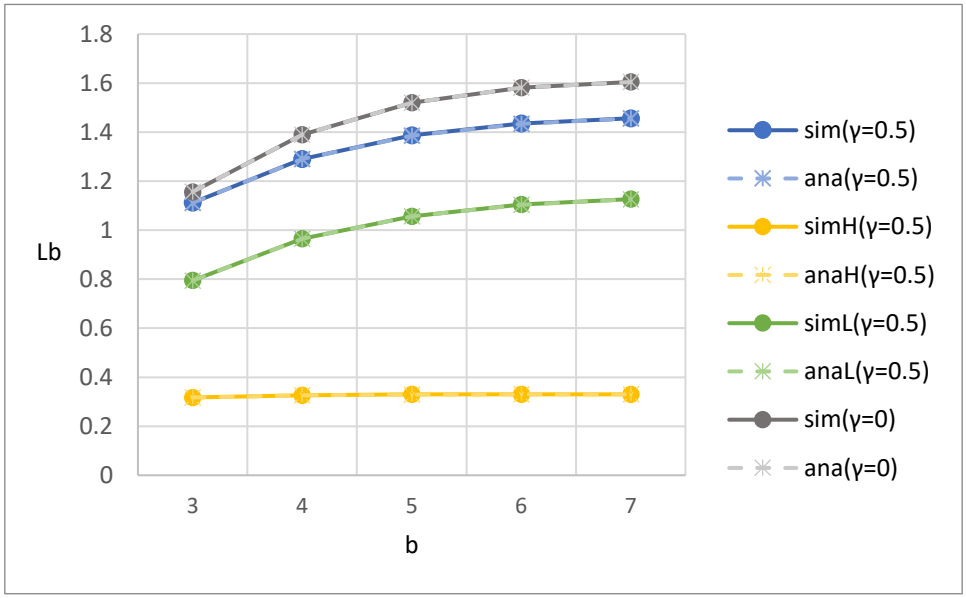


Figure 5-106: Effect of block size on average number of customers in block queue

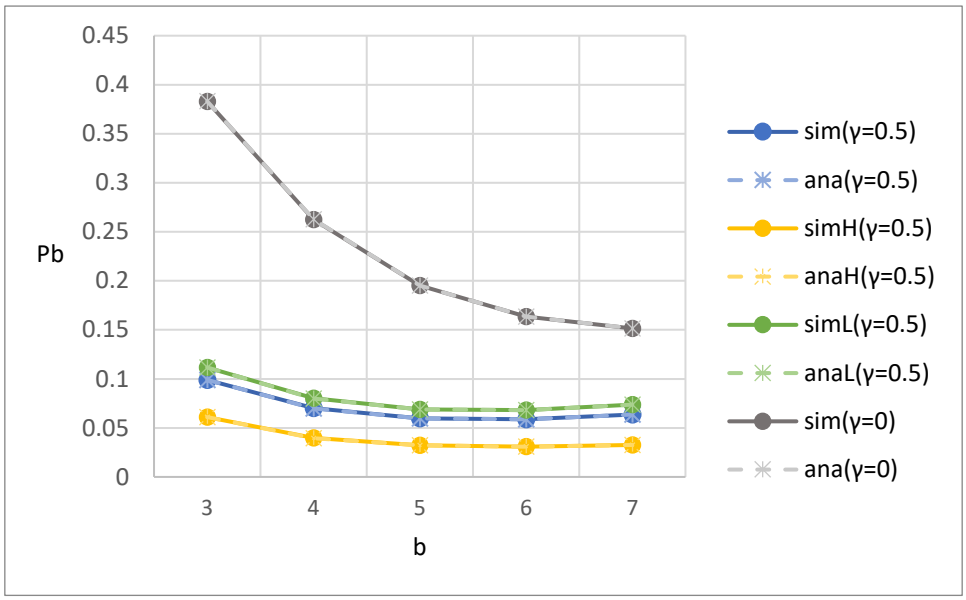


Figure 5-107: Effect of block size on blocking probability

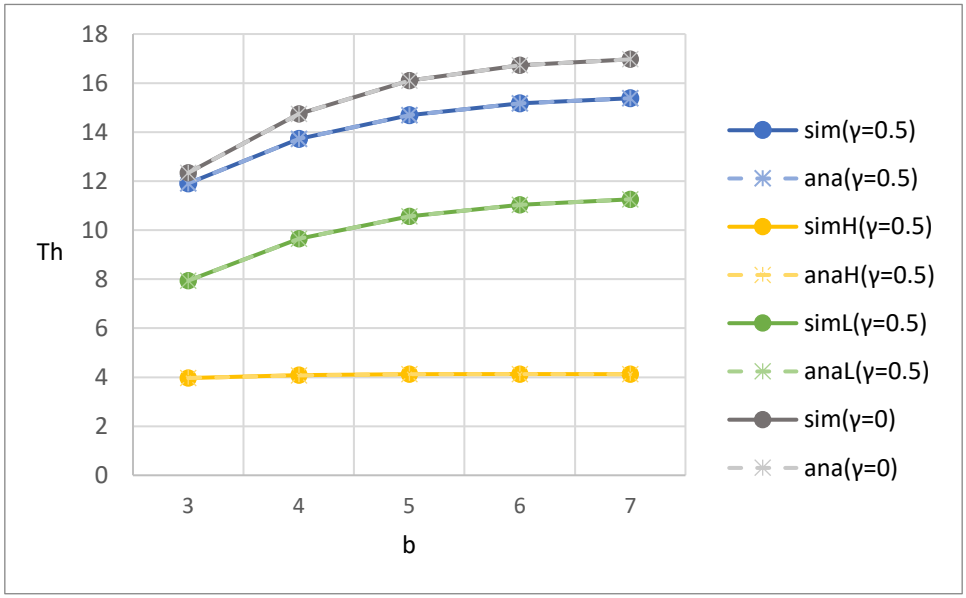


Figure 5-108: Effect of block size on system throughput

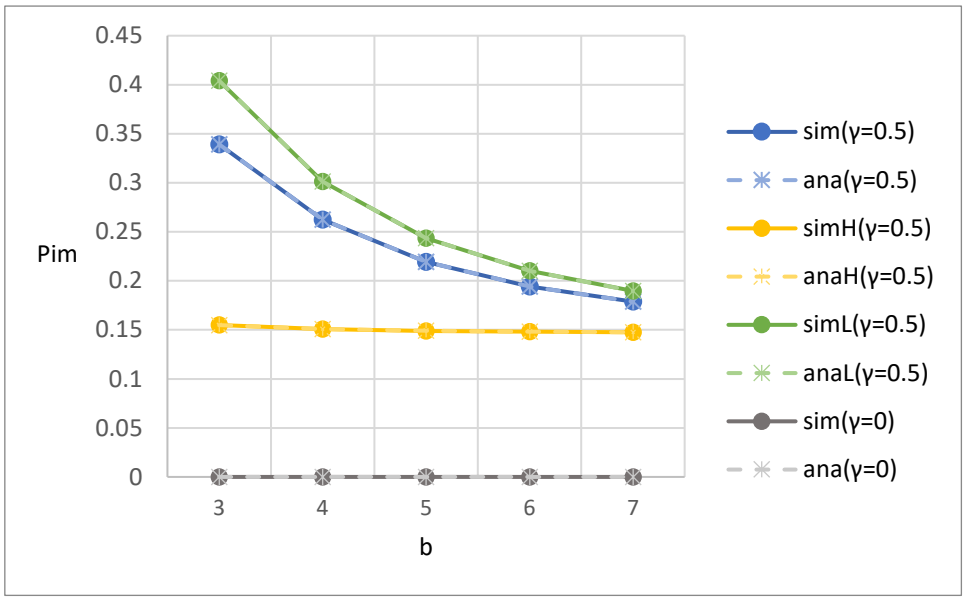


Figure 5-109: Effect of block size on the impatient probability

### 5.4.2. Arrival rate

Figure 5-110 to Figure 5-115 show the relationship between various performance metrics and the arrival rate of high-priority customers $\lambda_H$. Both simulation results and analytical results are shown for comparison.

Figure 5-110 illustrates the impact of the arrival rate of high-priority customers $\lambda_H$ on the average waiting times in the customer queue for high-priority, low-priority, and overall customers. As $\lambda_H$ increases, the $W_c$ increases steadily. The rise is mainly due to the significant increase in $W_{c_L}$, while $W_{c_H}$ remains nearly constant. This is because more high-priority arrivals dominate the queue under the non-preemptive priority mechanism, causing low-priority customers to wait longer in the customer queue. In addition, the $W_{c_H}$ is much shorter than $W_{c_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue. Additionally, $W_c$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which further suppresses queue buildup and contributes to the decline in $W_c$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-111 illustrates the impact of the arrival rate of high-priority customers $\lambda_H$ on the average waiting times in the block queue for high-priority, low-priority, and overall customers. As $\lambda_H$ increases, $W_{b_H}$ and $W_{b_L}$ remains nearly constant. This indicates that the time each block spends in the consensus queue is determined by the associated consensus rate and system transition rate, and is independent of $\lambda_H$. In addition, $W_{b_H}$ is smaller than $W_{b_L}$. This is because $\mu_{2_H}$ is larger than $\mu_{2_L}$. Furthermore, as $\lambda_H$ increases, $W_b$ decreases. This is because as $\lambda_H$ increases, more high-priority blocks are formed and the consensus rate of high-priority customers is larger than that of low-priority customers. In addition, $W_b$ with impatience is the same as that without impatience. This is because once a block is formed, impatience mechanism has no effect on it. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-112 illustrates the impact of the arrival rate of high-priority customers $\lambda_H$on the average waiting times in the system for high-priority, low-priority, and overall customers. As $\lambda_H$ increases, the $W$ increases steadily. The rise is mainly due to the significant increase in $W_L$, while $W_H$ remains nearly constant. This is because more $\lambda_H$ dominate the queue under the non-preemptive priority mechanism, causing low-priority customers to spend more time in the system. In addition, $W_H$ is much smaller

than $W_L$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue and $\mu_{2_H}$ is larger than $\mu_{2_L}$. Furthermore, $W$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which further suppresses queue buildup and contributes to the decline in $W$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-113 illustrates the impact of the arrival rate of high-priority customers $\lambda_H$ on the average numbers of customers in the block queue for high-priority, low-priority, and overall customers. As $\lambda_H$ increases, the $L_b$ remains relatively stable, but with diverging trends across priority class. Specifically, $L_{b_H}$ increases and $L_{b_L}$ decreases. This behavior reflects the shift in queue composition under the non-preemptive priority mechanism, where more high-priority customers are admitted into the system while low-priority customers are blocked earlier or delayed at the customer queue. Thus, more high-priority blocks and less low-priority blocks are formed. In addition, $L_{b_H}$ is smaller than $L_{b_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue and the high-priority customers have faster consensus rate than low-priority customers in the block queue, and therefore more low-priority customers remain waiting in the customer queue before being batched. Furthermore, $L_b$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which contributes to a reduction in $L_b$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-114 illustrates the impact of the arrival rate of high-priority customers $\lambda_H$ on the blocking probabilities for high-priority, low-priority, and overall customers. As $\lambda_H$ increases, the blocking probability increases across all priority levels. The rise is most significant for $P_{b_L}$, who face greater difficulty being admitted into the system due to the increased presence of high-priority arrivals. This trend reflects the effect of the non-preemptive priority mechanism, where high-priority customers dominate the queue and are less probability to be blocked, while low-priority customers experience higher blocking rates as system congestion intensifies. In addition, $P_{b_H}$ is smaller than $P_{b_L}$. This is because the capacity limit of the customer queue for the high-priority customers is no smaller than that for low-priority customers. Furthermore, $P_b$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which contributes to a reduction in $P_b$. Lastly, the analytical results are in

good agreement with the simulation results.

Figure 5-115 illustrates the impact of the arrival rate of high-priority customers $\lambda_H$ on the system throughputs for high-priority, low-priority, and overall customers. As $\lambda_H$ increases, $T_{h_H}$ increases and $T_{h_L}$ decreases. The $T_h$ remains relatively stable, as the gain in $T_{h_H}$ compensates for the loss in $T_{h_L}$. This behavior reflects the shift in resource allocation under the non-preemptive priority mechanism, where increasing $\lambda_H$ leads to more system capacity being devoted to high-priority customers at the expense of low-priority ones. In addition, $T_{h_H}$ is smaller than $T_{h_L}$. This is because the customer arrival rate of the high-priority customers is smaller than that of low-priority customers. Furthermore, $T_h$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, reducing effective $T_h$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-116 illustrates the impact of the arrival rate of high-priority customers $\lambda_H$ on the blocking probabilities for high-priority, low-priority, and overall customers. As $\lambda_H$ increases, the impatience probability increases across all priority levels. The increase is primarily driven by $P_{im_L}$, since a higher $\lambda_H$ leads to more high-priority customers occupying the queue under the non-preemptive mechanism, causing longer delays for low-priority customers and increasing their probability of abandonment. In contrast, $P_{im_H}$ remains nearly constant. Consequently, $P_{im_L}$ is higher than $P_{im_H}$, and $P_{im}$ with impatience is larger than without impatience. As described above, $P_{im}$ with impatience is larger than without impatience. Lastly, the analytical results are in good agreement with the simulation results.

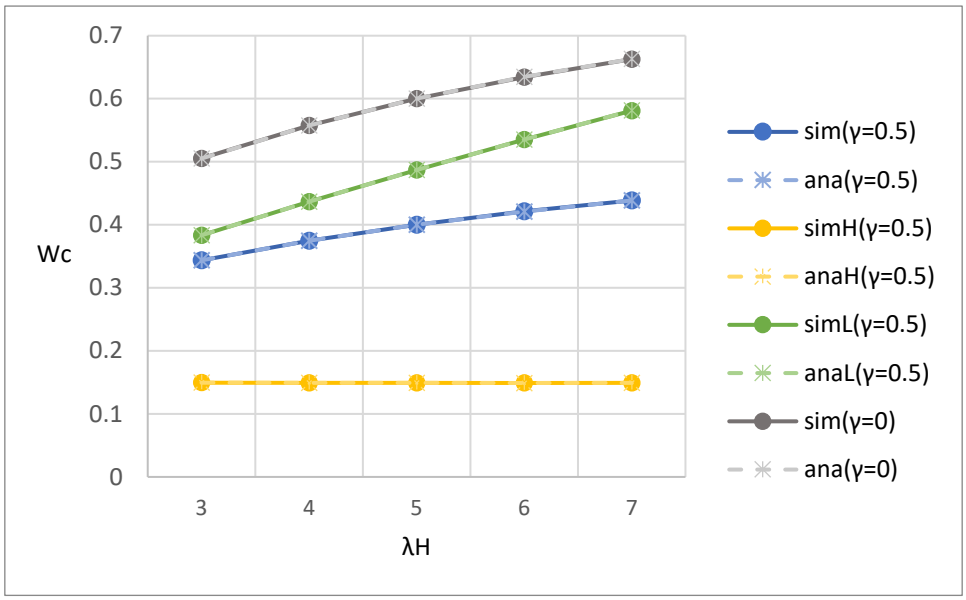


Figure 5-110: Effect of arrival rate on average waiting time in the customer queue

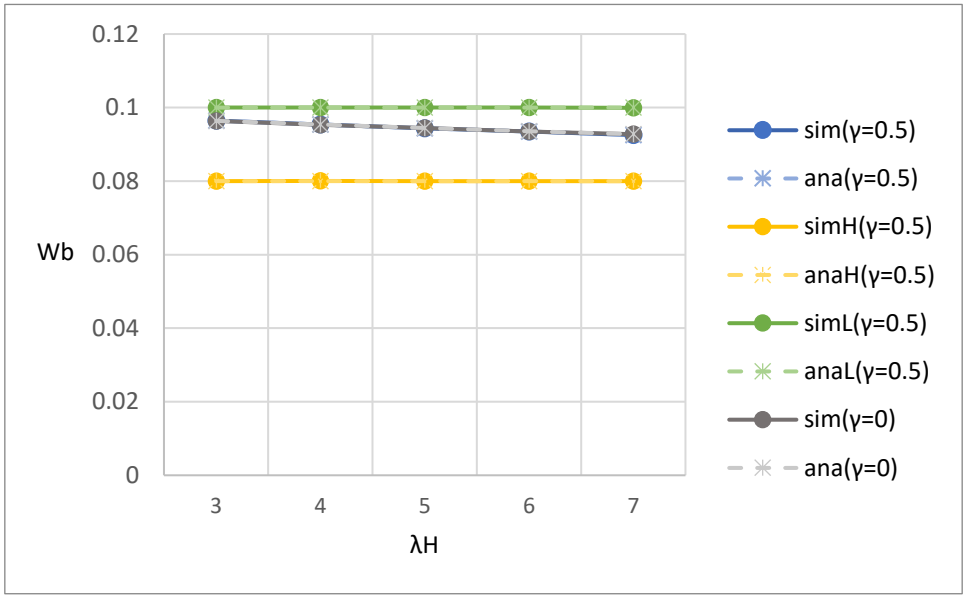


Figure 5-111: Effect of arrival rate on average waiting time in the block queue

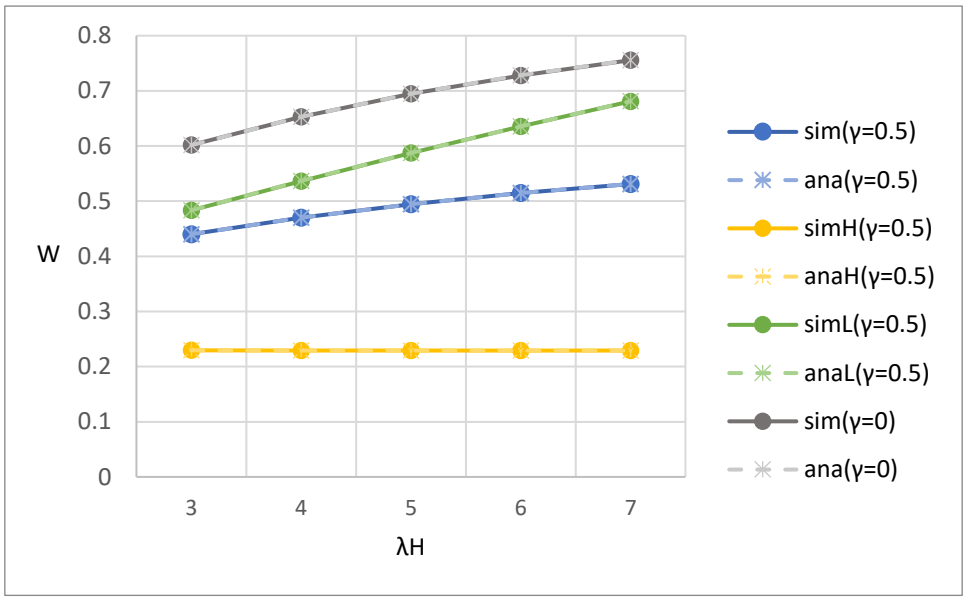


Figure 5-112: Effect of arrival rate on average waiting time in the system

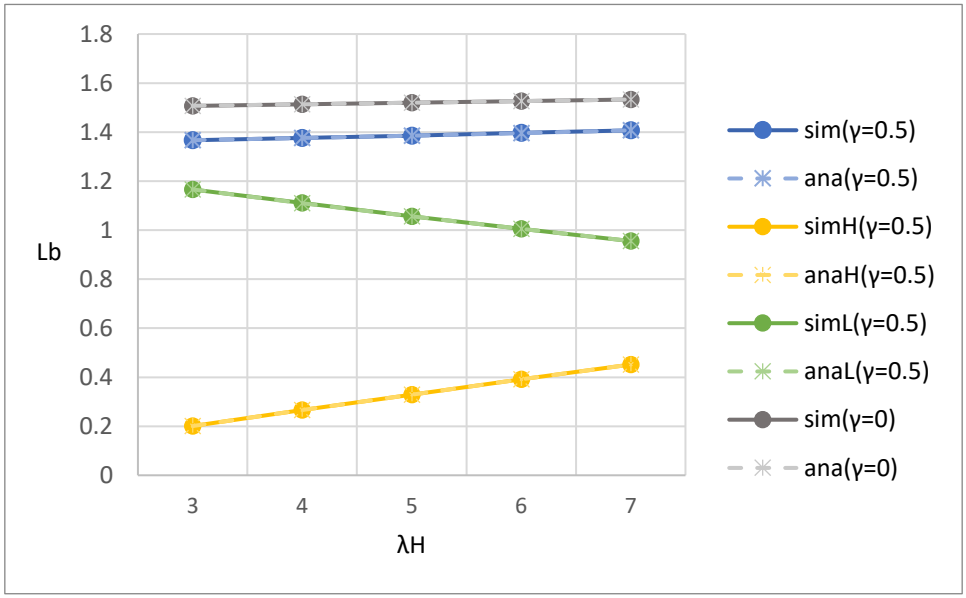


Figure 5-113: Effect of arrival rate on average number of customers in block queue

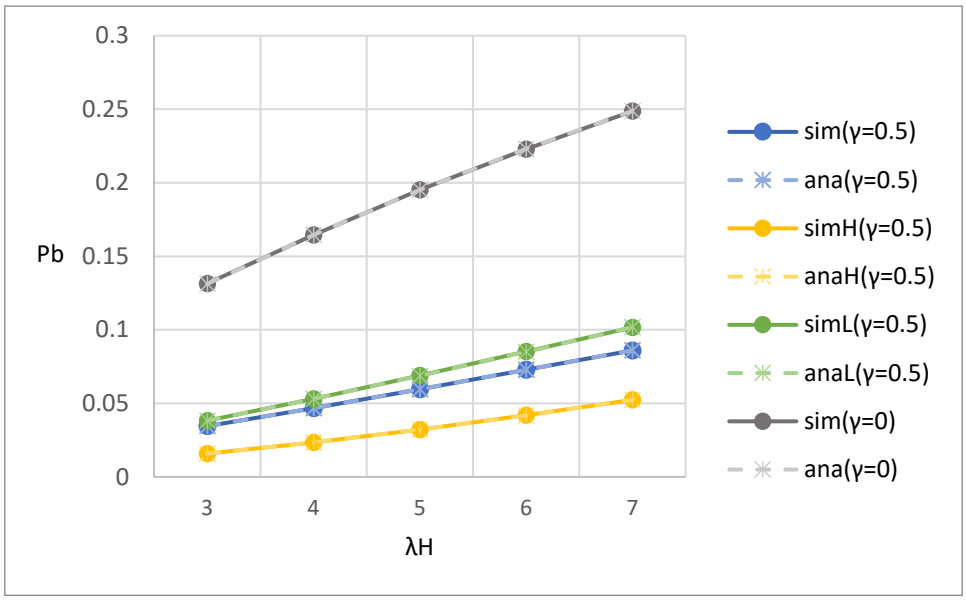


Figure 5-114: Effect of arrival rate on blocking probability

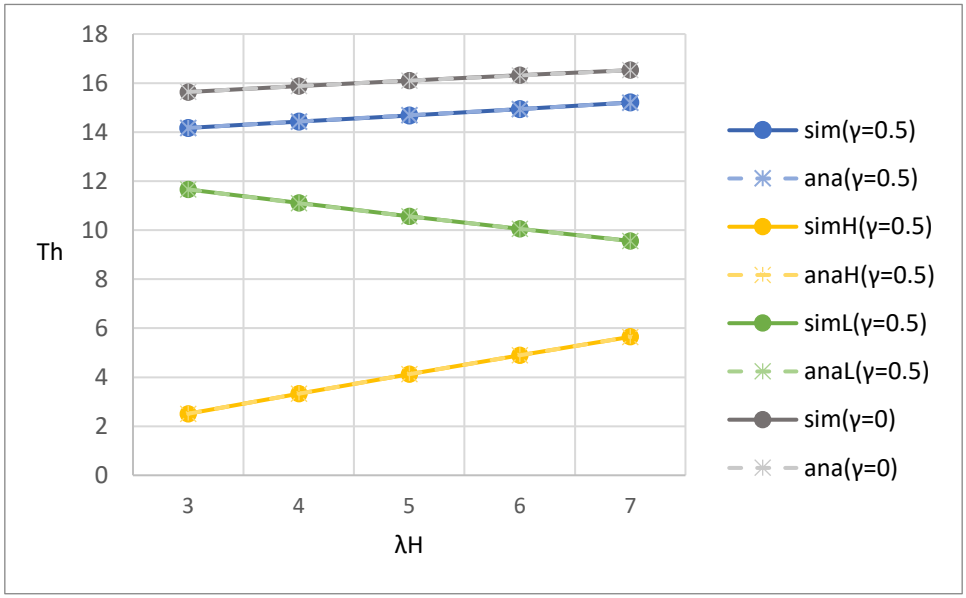


Figure 5-115: Effect of arrival rate on throughput

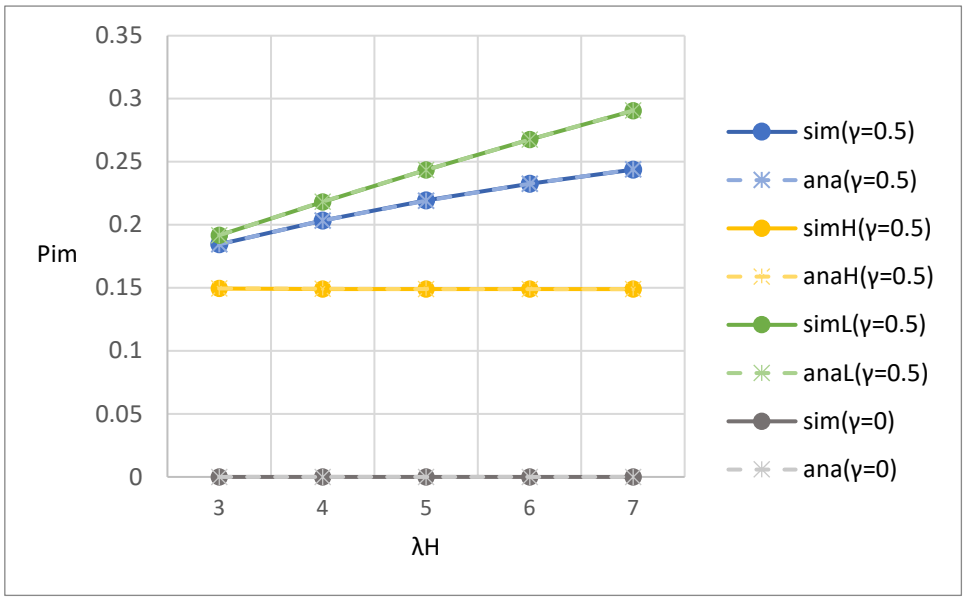


Figure 5-116: Effect of arrival rate on the impatient probability

### 5.4.3. Block generation rate

Figure 5-117 to Figure 5-122 show the relationship between various performance metrics and the block generation rate of high priority customers $\mu_{1_H}$. Both simulation results and analytical results are shown for comparison.

Figure 5-117 illustrates the impact of the block generation rate of high-priority customers $\mu_{1_H}$ on the average waiting times in the customer queue for high-priority, low-priority, and overall customers. As $\mu_{1_H}$ increases, the average waiting time in the customer queue decreases across all priority levels. The decline is more substantial for low-priority customers, who benefit from the increased service opportunities enabled by faster block generation. Although high-priority customers also experience shorter waiting times, their improvement is less pronounced since their queuing delay is already low. In addition, the $W_{c_H}$ is much smaller than $W_{c_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue. Furthermore, $W_c$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which further suppresses queue buildup and contributes to the decline in $W_c$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-118 illustrates the impact of the block generation rate of high-priority customers $\mu_{1_H}$ on the average waiting times in the block queue for high-priority, low-priority, and overall customers. As $\mu_{1_H}$ increases, the average waiting time in the block queue remains nearly constant across all priority levels. This indicates that the time each block spends in the consensus queue is determined by the associated consensus rate and system transition rate, and is independent of the block generation rate. In addition, the $W_{b_H}$ is smaller than $W_{b_L}$. This is because $\mu_{2_H}$ is larger than $\mu_{2_L}$. Furthermore, $W_b$ with impatience is the same as that without impatience. This is because once a block is formed, impatience mechanism has no effect on it. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-119 illustrates the impact of the block generation rate of high-priority customers $\mu_{1_H}$ on the average waiting times in the system for high-priority, low-priority, and overall customers. As $\mu_{1_H}$ increases, average waiting time in the system decreases across all priority levels. The rise is primarily contributed by $W_L$, while high-priority customers also benefit from more frequent block formation. This is because higher block generation rates allow customers to be grouped and processed more frequently, which reduces congestion in the customer queue. In addition, the $W_H$ is

much smaller than $W_L$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue and $\mu_{2_H}$ is larger than $\mu_{2_L}$. Furthermore, $W$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which further suppresses queue buildup and contributes to the decline in $W$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-120 illustrates the impact of the block generation rate of high-priority customers $\mu_{1_H}$ on the average numbers of customers in the block queue for high-priority, low-priority, and overall customers. As $\mu_{1_H}$ increases, the $L_b$ gradually increases. The rise is primarily contributed by $L_{b_L}$, while $L_{b_H}$ increases only slightly and remains at a relatively low level. This is because a higher $\mu_{1_H}$ allows high-priority customers to be processed more quickly, which indirectly leads to more low-priority customers forming batches. In addition, $L_{b_H}$ is smaller than $L_{b_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue and therefore more low-priority customers remain waiting in the customer queue before being batched. Furthermore, $L_b$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which contributes to a reduction in $L_b$.Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-121 illustrates the impact of the block generation rate of high-priority customers $\mu_{1_H}$ on the blocking probabilities for high-priority, low-priority, and overall customers. As $\mu_{1_H}$ increases, the blocking probability decreases across all priority levels. This is because the higher block generation rate allows high-priority customers to be served more frequently, which in turn release the capacity in the customer queue for low-priority customers. As a result, the probability that low-priority customers are blocked is reduced. In addition, $P_{b_H}$ is smaller than $P_{b_L}$. This is because the capacity limit of the customer queue for the high-priority customers is no smaller than that for low-priority customers. Furthermore, $P_b$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which contributes to a reduction in $P_b$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-122 illustrates the impact of the block generation rate of high-priority customers $\mu_{1_H}$ on the system throughputs for high-priority, low-priority, and overall customers. As $\mu_{1_H}$ increases, the system throughput increases across all priority levels

and then gradually saturates. The rise is primarily contributed by $T_{h_L}$, which increases more significantly due to the release of queue capacity made possible by faster processing of high-priority blocks. In contrast, $T_{h_H}$ eventually approaches a limit determined by $\lambda_H$. In addition, $T_{h_H}$ is smaller than $T_{h_L}$. This is because the customer arrival rate of the high-priority customers is smaller than that of low-priority customers. Furthermore, $T_h$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, reducing effective $T_h$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-123 illustrates the impact of the block generation rate of high-priority customers $\mu_{1_H}$ on the impatience probabilities for high-priority, low-priority, and overall customers. As $\mu_{1_H}$ increases, the impatient probability decreases across all priority levels. This trend is mainly due to the faster $\mu_{1_H}$ allowing high-priority customers to be served more quickly, which indirectly frees up capacity in the system. Consequently, low-priority customers experience shorter waiting times, reducing their likelihood of reaching their impatience threshold. In addition, $P_{im_L}$ is higher than $P_{im_H}$, as low-priority customers are more likely to experience longer waits due to the non-preemptive priority discipline. As described above, $P_{im}$ with impatience is larger than without impatience. Lastly, the analytical results are in good agreement with the simulation results.

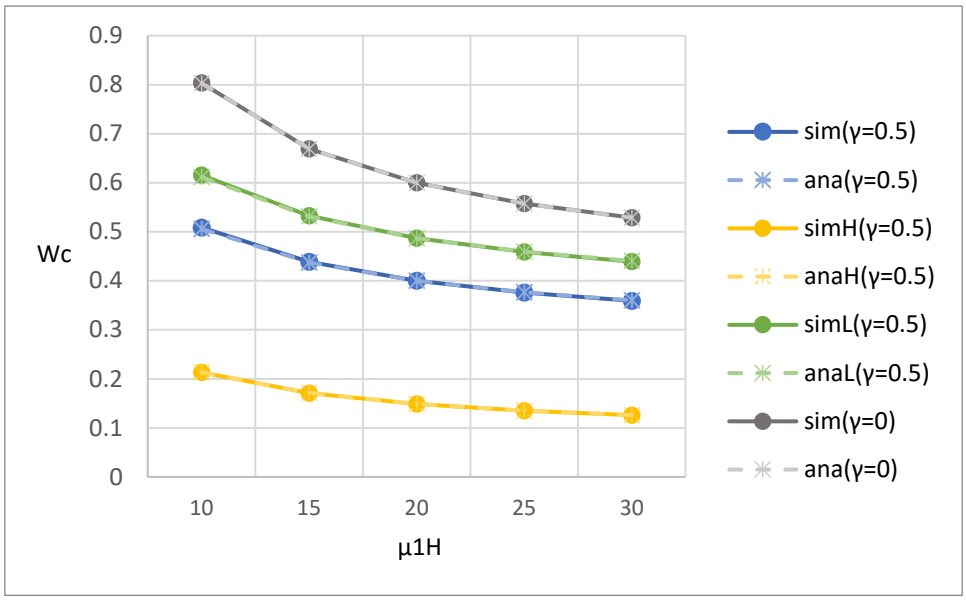


Figure 5-117: Effect of block generation rate on average waiting time in the customer queue

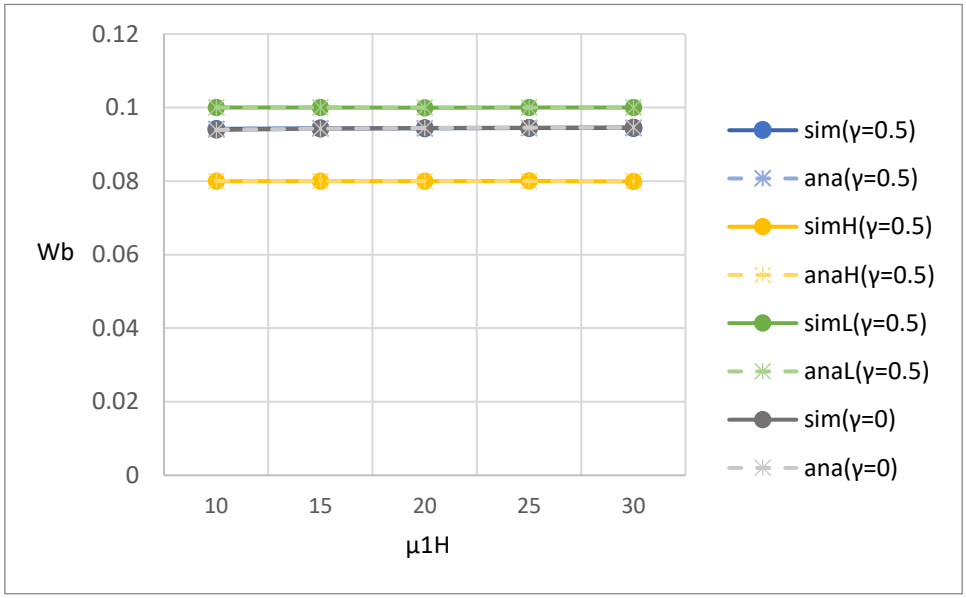


Figure 5-118: Effect of block generation rate on average waiting time in the block queue

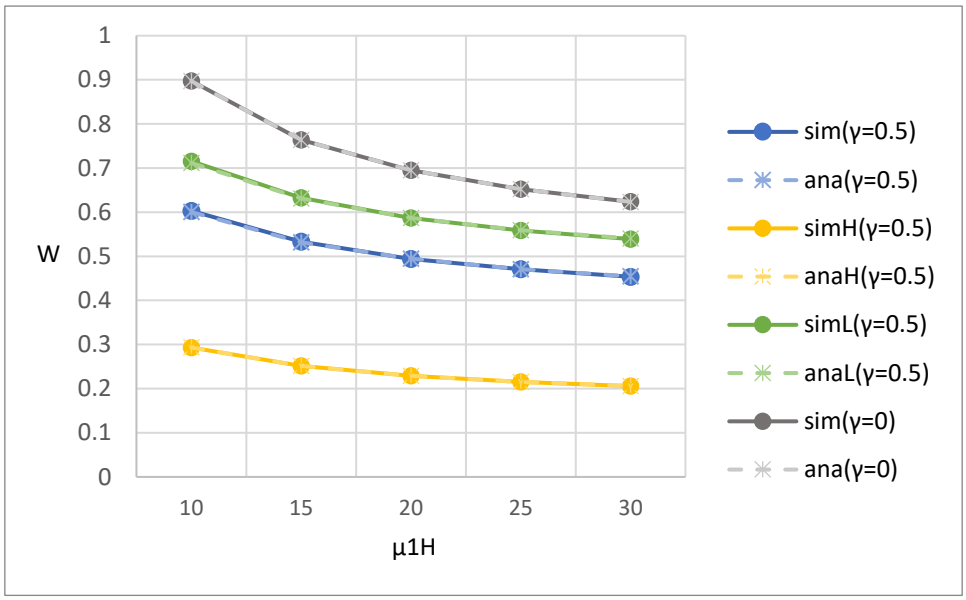


Figure 5-119: Effect of block generation rate on average waiting time in the system

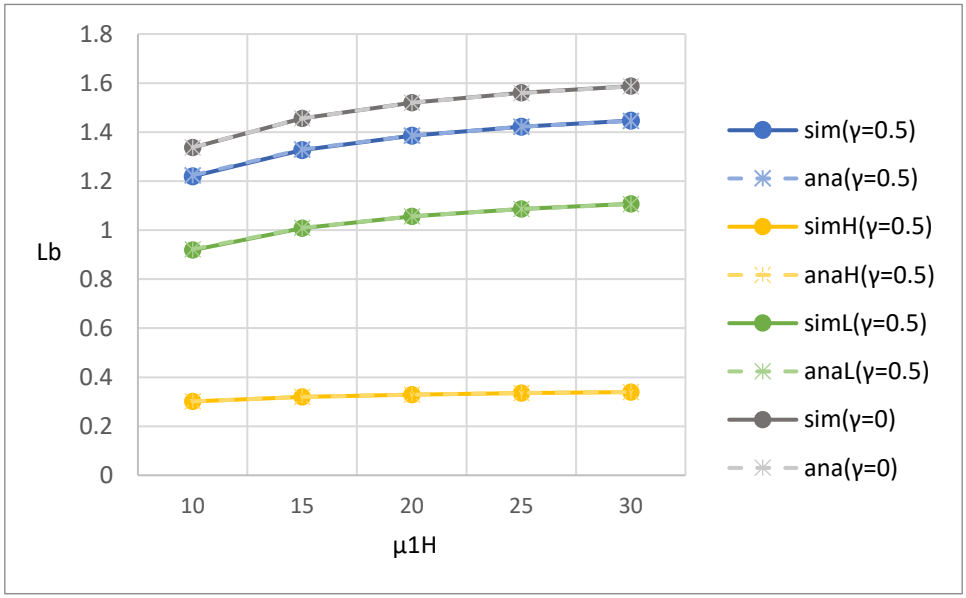


Figure 5-120: Effect of block generation rate on average number of customers in block queue

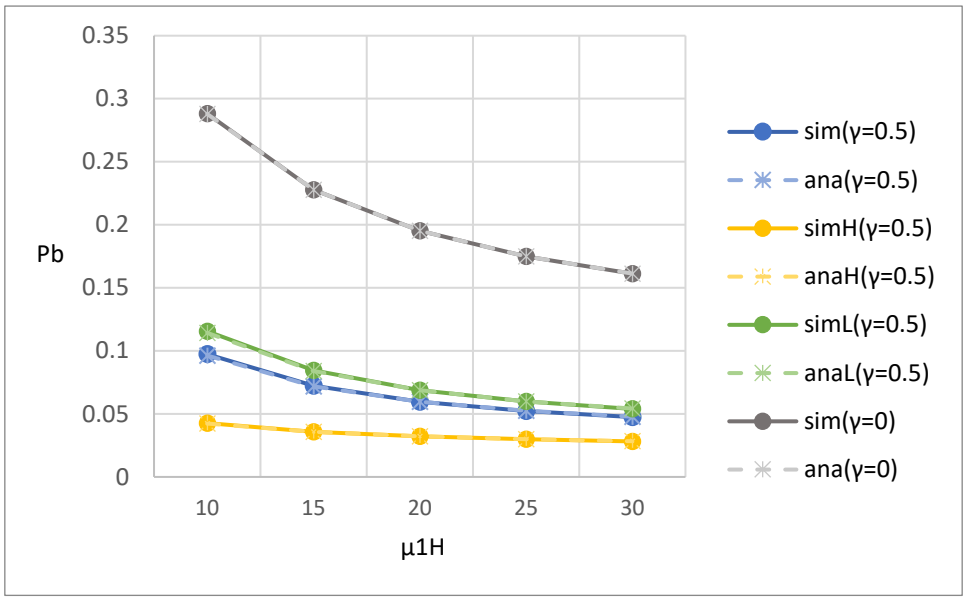


Figure 5-121: Effect of block generation rate on blocking probability

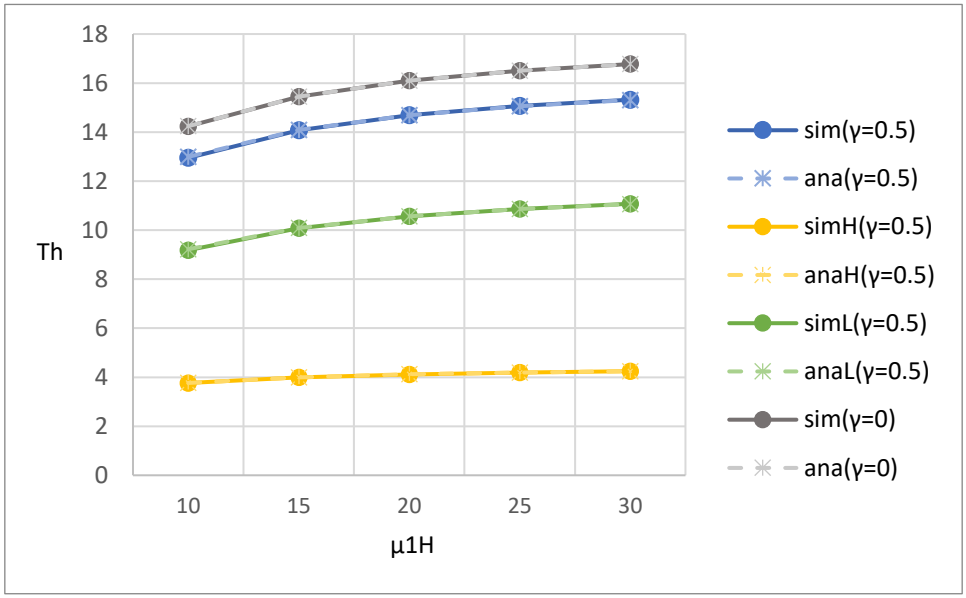


Figure 5-122: Effect of block generation rate on system throughput

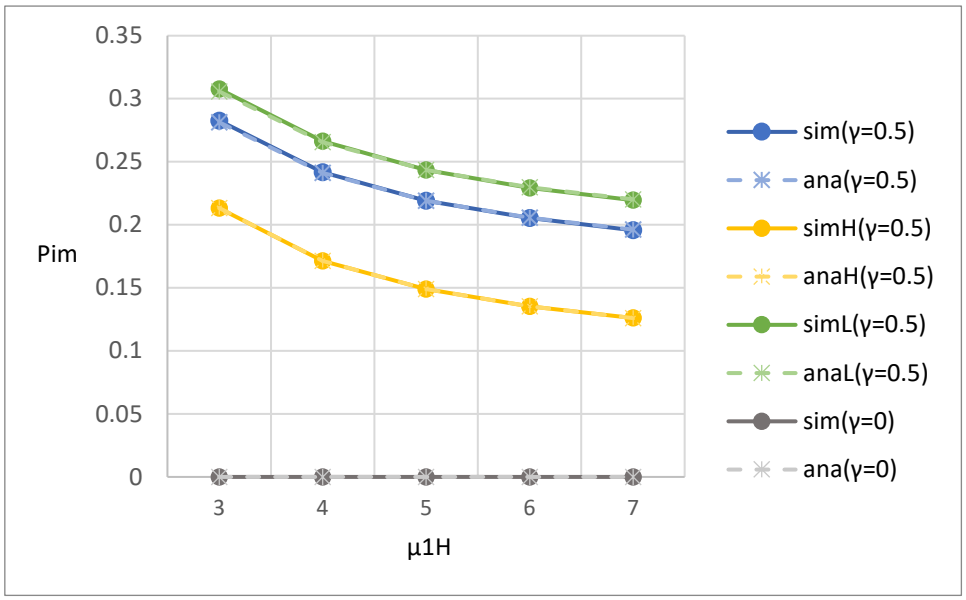


Figure 5-123: Effect of block generation rate on the impatient probability

### 5.4.4. Consensus rate

Figure 5-124 to Figure 5-129 show the relationship between various performance metrics and the consensus rate of high-priority customers $\mu_2$. Both simulation results and analytical results are shown for comparison.

Figure 5-124 illustrates the impact of the consensus rate of high-priority customers $\mu_{2_H}$ on the average waiting times in the customer queue for high-priority, low-priority, and overall customers. As $\mu_{2_H}$ increases, the $W_c$ decreases. The reduction is more pronounced for $W_{c_L}$, while $W_{c_H}$ remains relatively stable. This is because higher $\mu_{2_H}$ allows high-priority blocks to complete consensus more quickly, thereby shortening the duration that high-priority customers occupy system capacity. As a result, more capacity becomes available for low-priority customers, so that reducing $W_{c_L}$. In addition, the $W_{c_H}$ is much smaller than $W_{c_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue. Furthermore, $W_c$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which further suppresses queue buildup and contributes to the decline in $W_c$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-125 illustrates the impact of the consensus rate of high-priority customers $\mu_{2_H}$ on the average waiting times in the block queue for high-priority, low-priority, and overall customers. As $\mu_{2_H}$ increases, the $W_b$ decreases slightly. The decline is primarily contributed by $W_{b_H}$, while $W_{b_L}$ remains nearly constant. This is because a higher $\mu_{2_H}$ allows high-priority blocks to complete consensus more quickly, reducing the amount of time the high-priority customers spend waiting in the block queue. Since the consensus rate of low-priority customers is not affected, their waiting time remains unchanged. In addition, at $\mu_{2_H} = 15$, $W_{b_H}$ is noticeably higher than $W_{b_L}$, but the two curves intersect around $\mu_{2_H} = 20$, after which $W_{b_H}$ becomes lower. This crossover occurs because $\mu_{2_H}$ starts lower than the fixed $\mu_{2_L} = 20$, and increasing $\mu_{2_H}$ improves high-priority processing speed. Furthermore, $W_b$ with impatience is the same as that without impatience. This is because once a block is formed, impatience mechanism has no effect on it. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-126 illustrates the impact of the consensus rate of high-priority customers $\mu_{2_H}$ on the average waiting times in the system for high-priority, low-priority, and overall customers. As $\mu_{2_H}$ increases, the average waiting time in the system decreases

steadily for all priority levels. This is because faster $\mu_{2_H}$ reduces delays in the block queue, thereby improving system efficiency. While $W_L$ also decreases slightly as system capacity is released from high-priority customers. This trend highlights how improving consensus efficiency for high-priority customers can enhance overall system flow and reduce $W_H$ and $W_L$. In addition, the $W_H$ is much smaller than $W_L$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue. Furthermore, $W$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which further suppresses queue buildup and contributes to the decline in $W$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-127 illustrates the impact of the consensus rate of high-priority customers $\mu_{2_H}$ on the average numbers of customers in the block queue for high-priority, low-priority, and overall customers. As $\mu_{2_H}$ increases, the $L_b$ slightly decreases. This change is mainly driven by a noticeable reduction in $L_{b_H}$, while $L_{b_L}$ gradually increases due to the admission of more low-priority customers as system capacity is released by faster $\mu_{2_H}$. This reflects how increasing $\mu_{2_H}$ can reduce queue capacity for high-priority customers while indirectly increasing $L_{b_L}$. In addition, $L_{b_H}$ is smaller than $L_{b_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue and $\mu_{2_H}$ is larger than $\mu_{2_L}$, and therefore more low-priority customers remain waiting in the customer queue before being batched. Furthermore, $L_b$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which contributes to a reduction in $L_b$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-128 illustrates the impact of the consensus rate of high-priority customers $\mu_{2_H}$ on the blocking probabilities for high-priority, low-priority, and overall customers. As $\mu_{2_H}$ increases, the blocking probability decreases across all priority levels. This decline is more significant for high-priority customers, whose blocks are processed more rapidly with higher $\mu_{2_H}$, resulting in fewer delays and fewer blocked arrivals. Also, faster processing of high-priority customers indirectly frees up capacity in the customer queue, reducing the $P_{b_L}$ as well. In addition, $P_{b_H}$ is smaller than $P_{b_L}$. This is because the capacity limit of the customer queue for the high-priority customers is no smaller than that for low-priority customers. Furthermore, $P_b$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which contributes to a reduction in $P_b$. Lastly, the analytical results are in good agreement

with the simulation results.

Figure 5-129 illustrates the impact of the consensus rate of high-priority customers $\mu_{2_H}$ on the system throughputs for high-priority, low-priority, and overall customers. As $\mu_{2_H}$ increases, the system throughput increases across all priority levels and then gradually saturates. The rise is primarily contributed by $T_{h_L}$, which increases more significantly due to the release of queue capacity made possible by faster processing of high-priority blocks. In contrast, $T_{h_H}$ eventually approaches a limit determined by $\lambda_H$. In addition, $T_{h_H}$ is smaller than $T_{h_L}$. This is because the customer arrival rate of the high-priority customers is smaller than that of low-priority customers. Furthermore, $T_h$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, reducing effective $T_h$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-130 illustrates the impact of the consensus rate of high-priority customers $\mu_{2_H}$ on the impatience probabilities for high-priority, low-priority, and overall customers. As $\mu_{2_H}$ increases, the impatience probability decreases across all priority levels. This is because a higher $\mu_{2_H}$ allows high-priority blocks to be processed more quickly, releasing system capacity and enabling faster admission of waiting customers. Consequently, customers spend less time in the customer queue, reducing the probability of reaching their impatience threshold. In addition, $P_{im_L}$ is higher than $P_{im_H}$, as low-priority customers are more likely to experience longer waits due to the non-preemptive priority discipline. As described above, $P_{im}$ with impatience is larger than without impatience. Lastly, the analytical results are in good agreement with the simulation results.

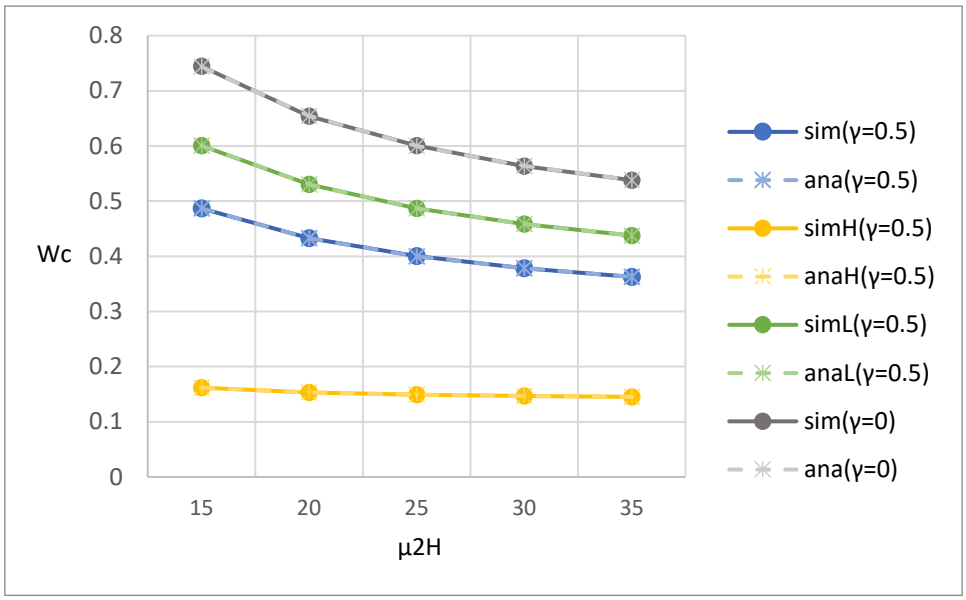


Figure 5-124: Effect of consensus rate on average waiting time in the customer queue

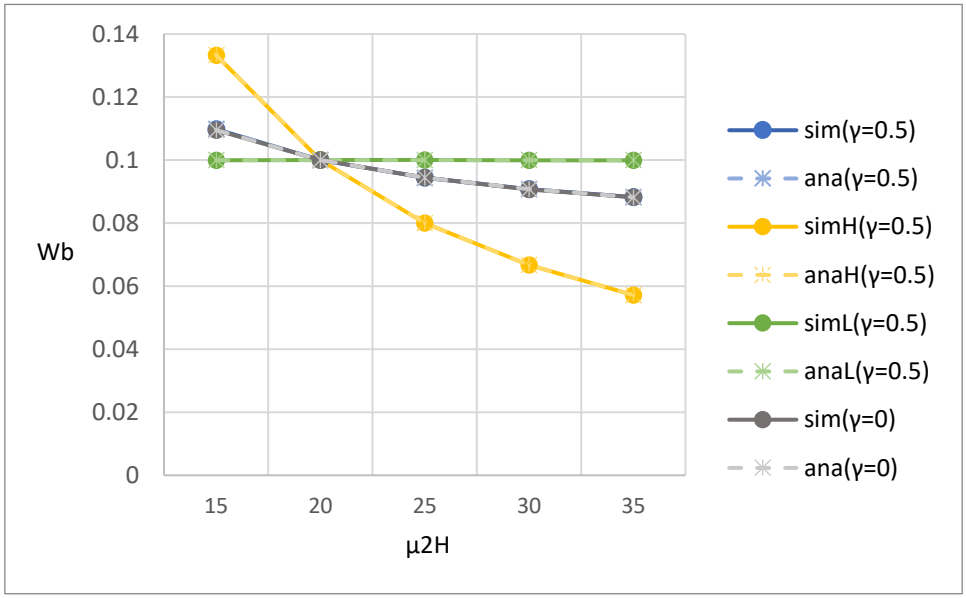


Figure 5-125: Effect of consensus rate on average waiting time in the block queue

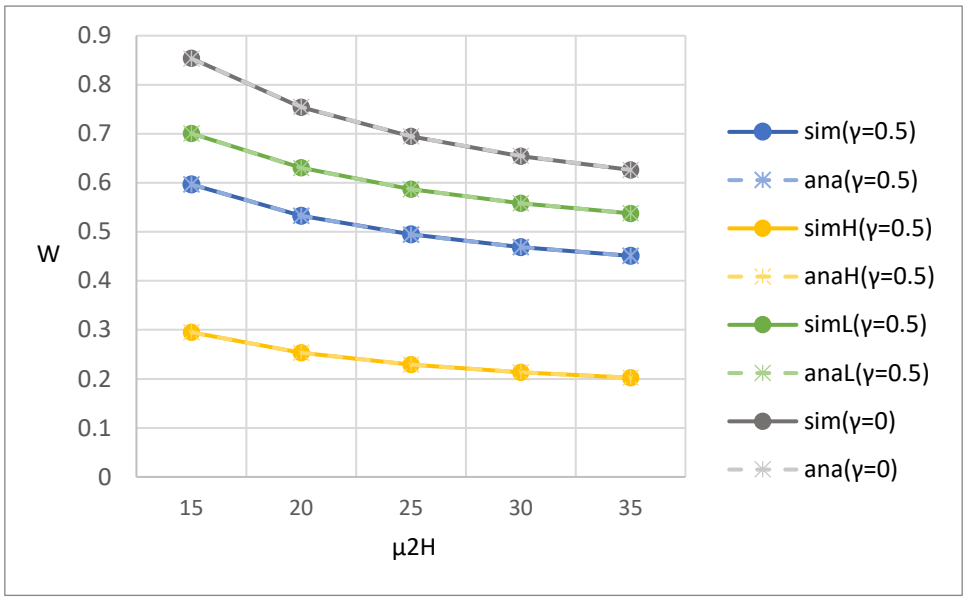


Figure 5-126: Effect of consensus rate on average waiting time in the system

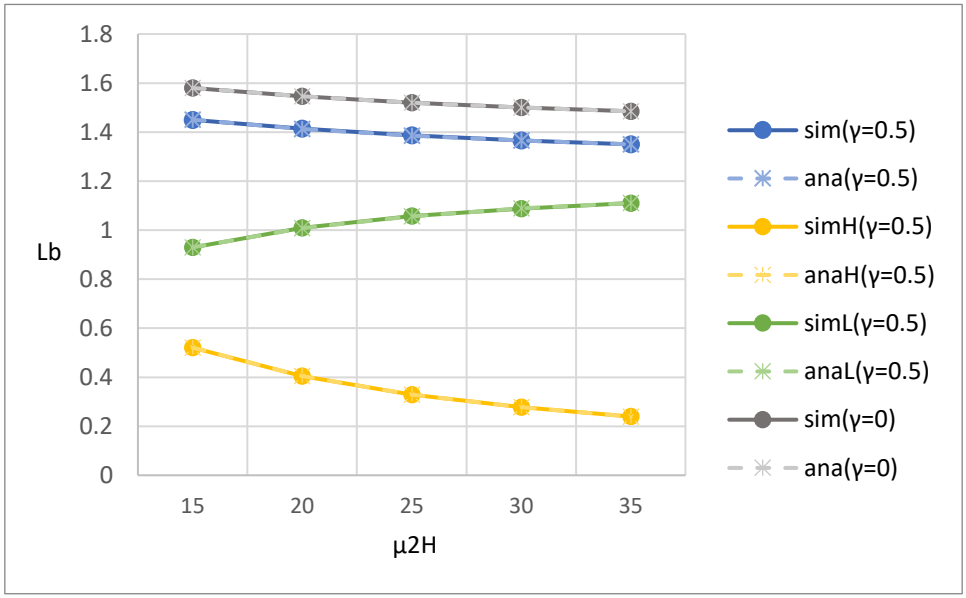


Figure 5-127: Effect of consensus rate on average number of customers in block queue

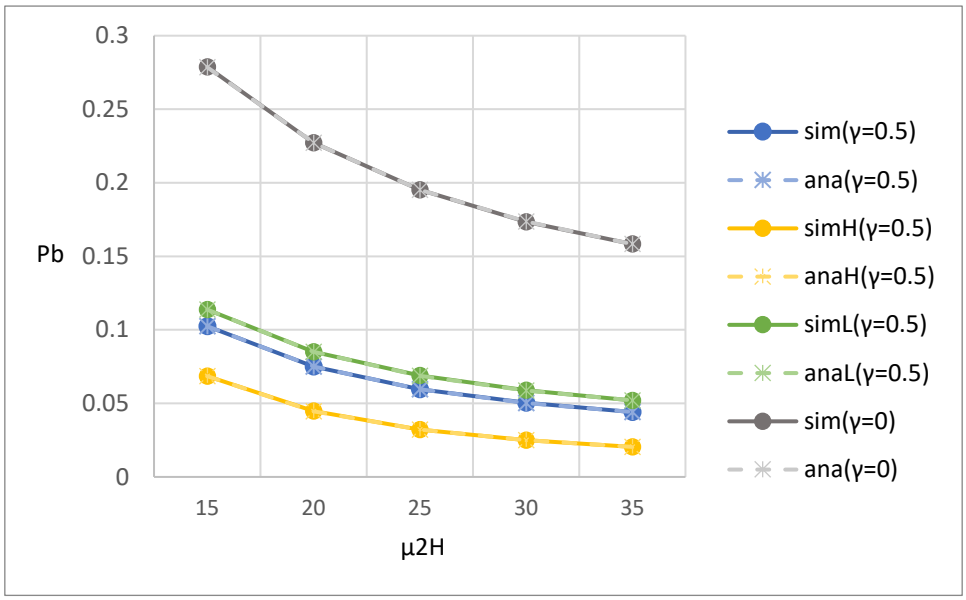


Figure 5-128: Effect of consensus rate on blocking probability

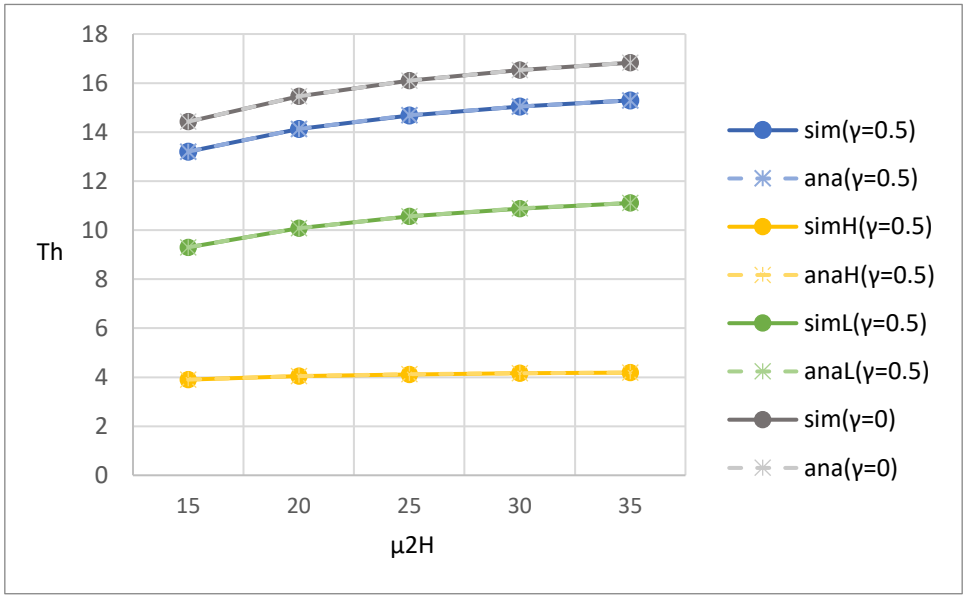


Figure 5-129: Effect of consensus rate on system throughput

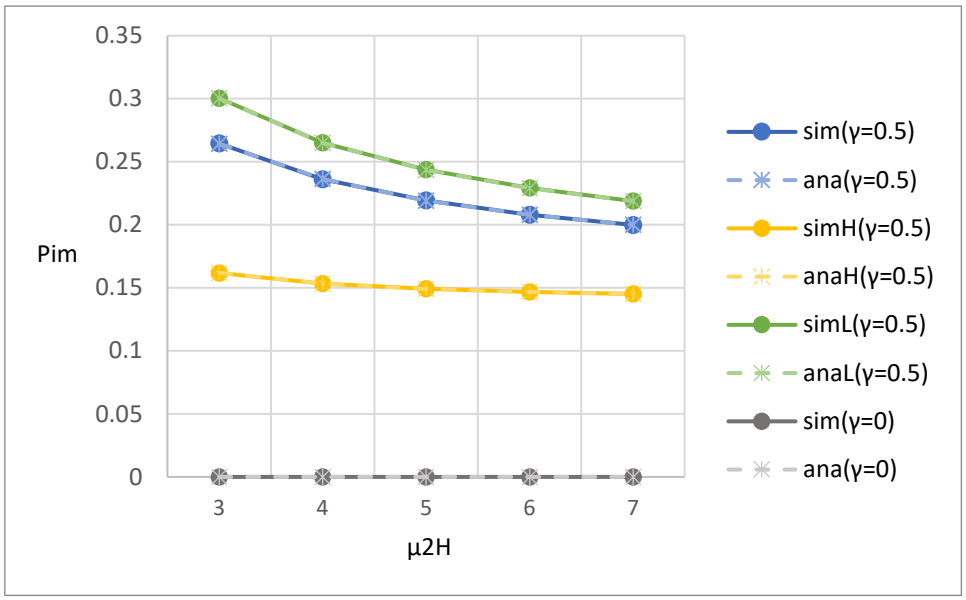


Figure 5-130: Effect of consensus rate on the impatient probability

### 5.4.5. Transition rate (ON to OFF)

Figure 5-131 to Figure 5-136 show the relationship between various performance metrics and the transition rate $\alpha$. Both simulation results and analytical results are shown for comparison.

Figure 5-131 illustrates the impact of the transition rate $\alpha$ on the average waiting times in the customer queue for high-priority, low-priority, and overall customers. As $\alpha$ increases, the average waiting time in the customer queue rises steadily across all priority levels. The increase is more pronounced for low-priority customers, who are more affected by interruptions in service. This is because more frequent transitions from ON to OFF reduce the availability of block generation service, causing longer queueing delays for arriving customers, especially for customers with lower priority. In addition, the $W_{c_H}$ is much smaller than $W_{c_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue. Furthermore, $W_c$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which further suppresses queue buildup and contributes to the decline in $W_c$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-132 illustrates the impact of the transition rate $\alpha$ on the average waiting times in the block queue for high-priority, low-priority, and overall customers. As $\alpha$ increases, the average waiting time in the block queue increase steadily across all priority levels. This is because more frequent service interruptions delay consensus processing, resulting in longer waiting times for customer(s) in block queue. This effect is observed for both high- and low-priority customers, with low-priority customers experiencing slightly longer delay. In addition, the $W_{b_H}$ is smaller than $W_{b_L}$. This is because $\mu_{2_H}$ is larger than $\mu_{2_L}$. Furthermore, $W_b$ with impatience is the same as that without impatience. This is because once a block is formed, impatience mechanism has no effect on it. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-133 illustrates the impact of the transition rate $\alpha$ on the average waiting times in the system for high-priority, low-priority, and overall customers. As $\alpha$ increases, the average waiting time in the system increase steadily across all priority levels. The increase is more significant for $W_L$, while $W_H$ increases slightly. This is because more frequent service interruptions caused by transitions to the OFF state reduce overall availability of block generation and consensus service, leading to longer

queueing delays for customers in the system. In addition, the $W_H$ is much smaller than $W_L$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue and $\mu_{2_H}$ is larger than $\mu_{2_L}$. Furthermore, $W$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which further suppresses queue buildup and contributes to the decline in $W$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-134 illustrates the impact of the transition rate $\alpha$ on the average numbers of customers in the block queue for high-priority, low-priority, and overall customers. As $\alpha$ increases, the average number of customers in the block queue increase steadily across all priority levels. This is because more frequent service interruptions of block generation processing cause more customers to wait in the customer queue while forming batches from the customer queue. In addition, $L_{b_H}$ is smaller than $L_{b_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue and $\mu_{2_H}$ is larger than $\mu_{2_L}$, and therefore more low-priority customers remain waiting in the customer queue before being batched. Furthermore, $L_b$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which contributes to a reduction in $L_b$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-135 illustrates the impact of the transition rate $\alpha$ on the blocking probabilities for high-priority, low-priority, and overall customers. As $\alpha$ increases, the blocking probability increase steadily across all priority levels. The rise is more pronounced for $P_{b_L}$, while $P_{b_H}$ increases at a slower rate. This is because more frequent service interruptions reduce the system's capacity to process customers, which increases the probability that the customer queue reaches its capacity and causes incoming arrivals to be blocked. In addition, $P_{b_H}$ is smaller than $P_{b_L}$. This is because the capacity limit of the customer queue for the high-priority customers is no smaller than that for low-priority customers. Furthermore, $P_b$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, which contributes to a reduction in $P_b$.Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-136 illustrates the impact of the transition rate $\alpha$ on the system throughputs for high-priority, low-priority, and overall customers. As $\alpha$ increases, the system throughput decreases gradually across all priority levels. This is because more

frequent service interruptions reduce the chance for block generation and consensus processing, thereby limiting the rate at which customers are served and ultimately lowering the $T_h$. The decline is more pronounced for $T_{h_L}$, whose service opportunities are more severely impacted by limited availability, while $T_{h_H}$ remains relatively stable. In addition, $T_{h_H}$ is smaller than $T_{h_L}$. This is because the customer arrival rate of the high-priority customers is smaller than that of low-priority customers. Furthermore, $T_h$ with impatience is smaller than that without impatience. This is because the impatience mechanism causes customers to leave the queue once their impatience threshold is reached, reducing effective $T_h$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-137 illustrates the impact of the transition rate $\alpha$ on the impatience probabilities for high-priority, low-priority, and overall customers. As $\alpha$ increases, the impatience probability increase steadily across all priority levels. This is because more frequent service interruptions reduce the effective service availability, causing customers to experience longer waiting times in the customer queue, particularly for low-priority customers. As a result, $P_{im_L}$ are more likely to reach their impatience threshold and leave the system. In addition, $P_{im_L}$ is higher than $P_{im_H}$, as low-priority customers are more likely to experience longer waits due to the non-preemptive priority discipline. As described above, $P_{im}$ with impatience is larger than without impatience. Lastly, the analytical results are in good agreement with the simulation results.

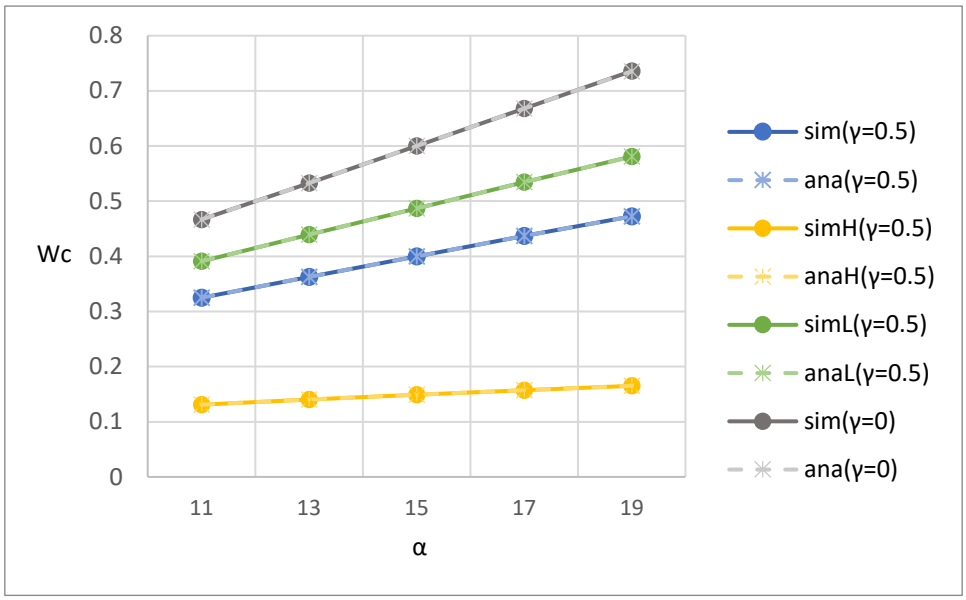


Figure 5-131: Effect of transition rate on average waiting time in the customer queue in the system

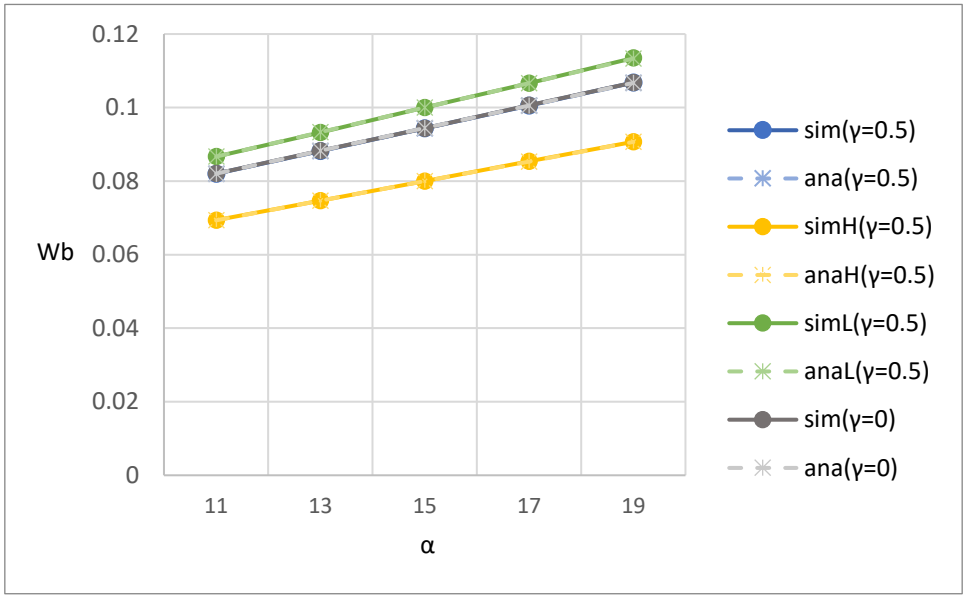


Figure 5-132: Effect of transition rate on average waiting time in the block queue

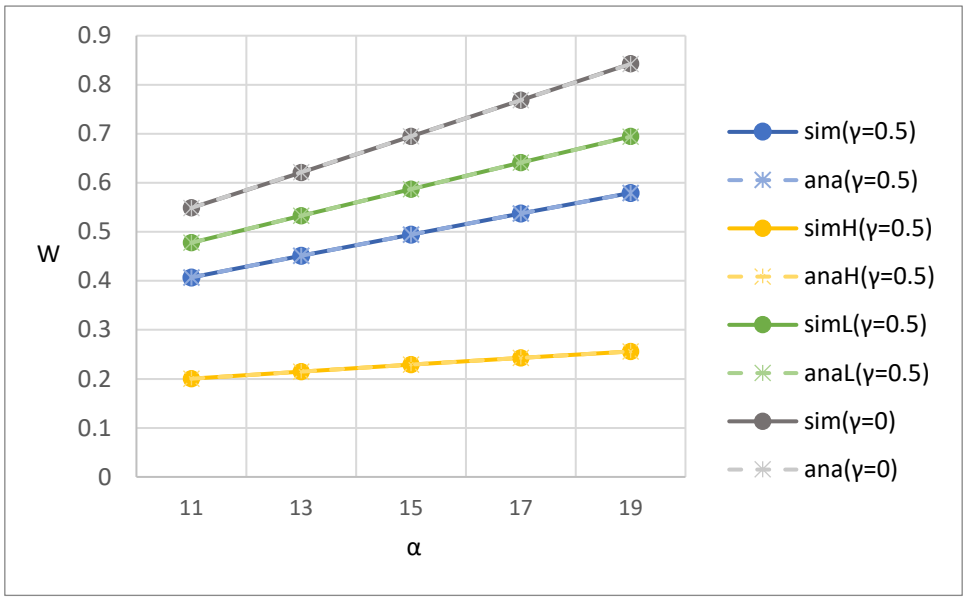


Figure 5-133: Effect of transition rate on average waiting time in the system

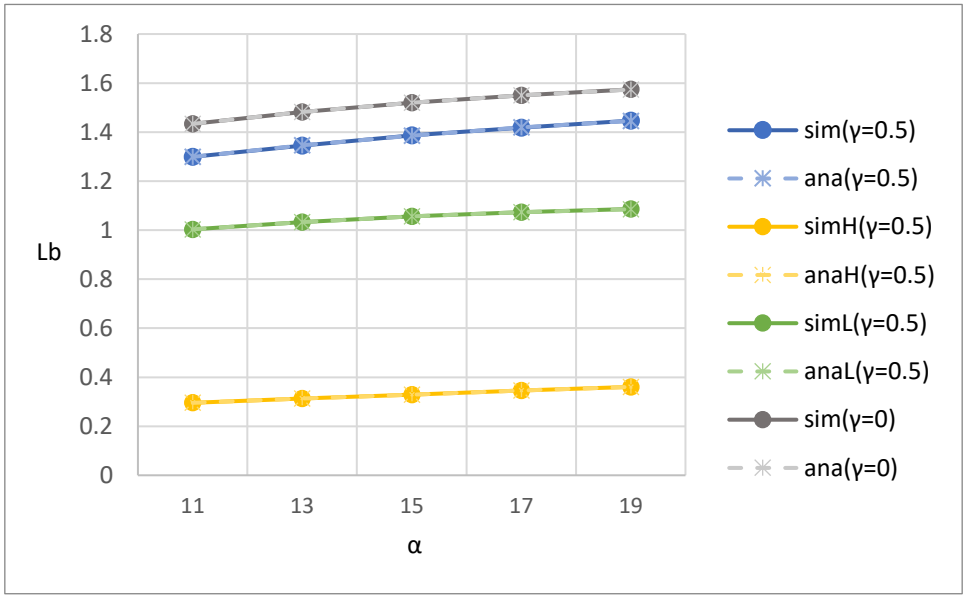


Figure 5-134: Effect of transition rate on average number of customers in block queue

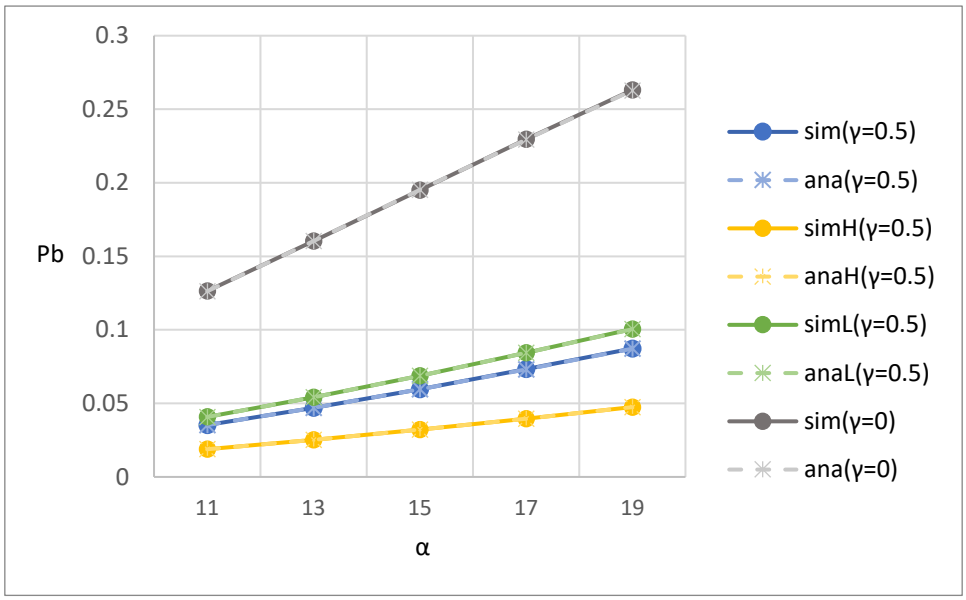


Figure 5-135: Effect of transition rate on blocking probability

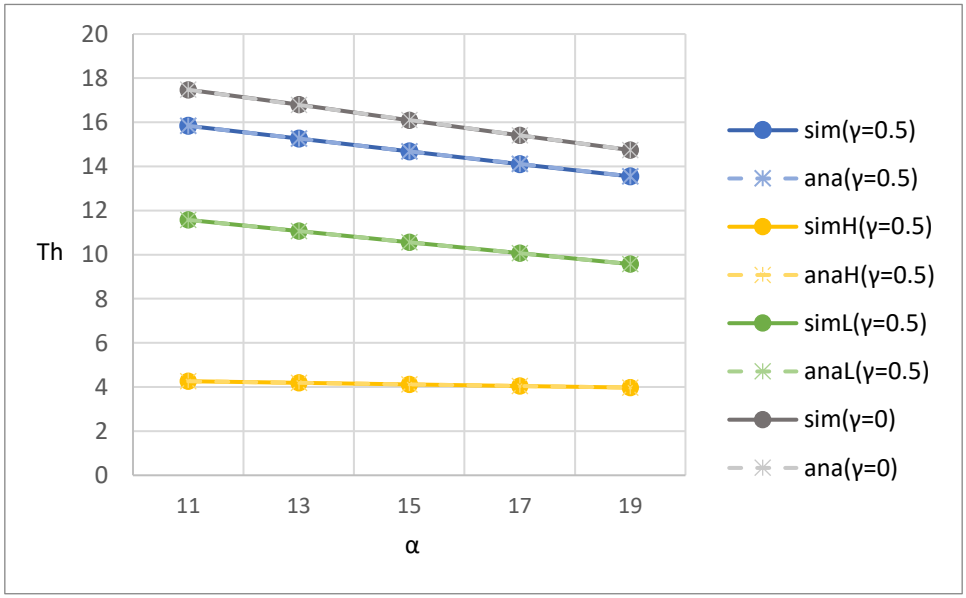


Figure 5-136: Effect of transition rate on system throughput

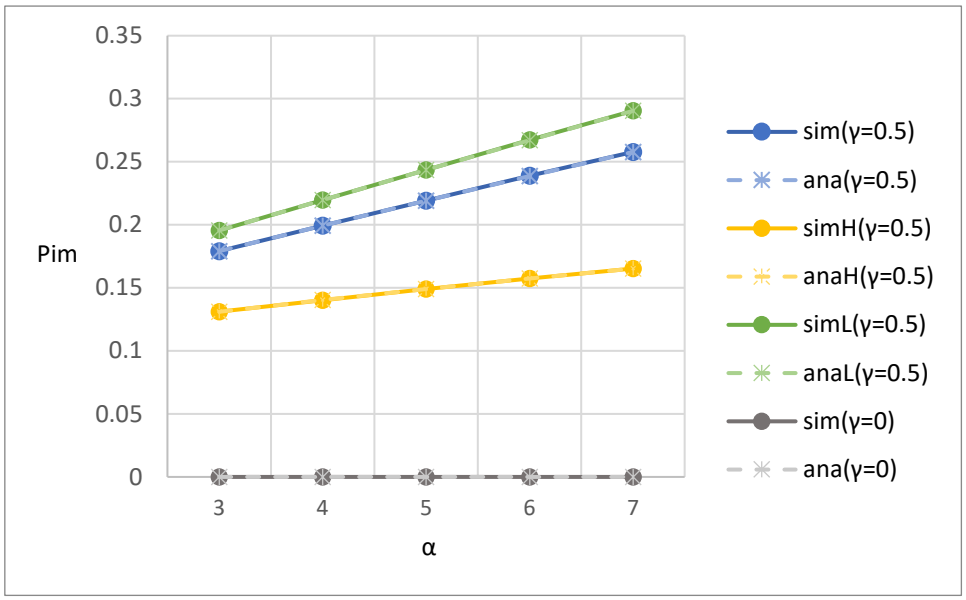


Figure 5-137: Effect of transition rate on the impatient probability

### 5.4.6. Impatient rate

Figure 5-138 to Figure 5-143 show the relationship between various performance metrics and the impatient rate $\gamma$. Both simulation results and analytical results are shown for comparison.

Figure 5-138 illustrates the impact of the impatience rate $\gamma$ on the average waiting times in the customer queue for high-priority, low-priority, and overall customers. As $\gamma_H$ increases, the average waiting time in the customer queue decreases slightly across all priority levels. This is because higher $\gamma_H$ decreases the likelihood that high-priority customers will abandon the queue upon reaching their impatience threshold, thereby releasing system capacity and allowing more low-priority customers to be served. This shortens the overall queue length and reducing the average waiting time in customer queue. In addition, the $W_{c_H}$ is much smaller than $W_{c_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-139 illustrates the impact of the impatience rate $\gamma$ on the average waiting times in the block queue for high-priority, low-priority, and overall customers. As $\gamma_H$ increases, the average waiting time in the block queue decrease steadily across all priority levels. This is because the impatience mechanism only affects customers while they are in the customer queue. Once a block is formed, they are no longer subject to abandonment due to impatience. In addition, the $W_{b_H}$ is smaller than $W_{b_L}$. This is because $\mu_{2_H}$ is larger than $\mu_{2_L}$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-140 illustrates the impact of the impatience rate $\gamma_H$ on the average waiting times in the system for high-priority, low-priority, and overall customers. As $\gamma_H$ increases, the average waiting time in the system decrease steadily across all priority levels. This is because higher $\gamma_H$ increases the likelihood that high-priority customers will abandon the queue upon reaching their impatience threshold, thereby releasing system capacity and allowing more low-priority customers to be served. In addition, the $W_H$ is much smaller than $W_L$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue and $\mu_{2_H}$ is larger than $\mu_{2_L}$. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-141 illustrates the impact of the impatience rate $\gamma$ on the average

numbers of customers in the block queue for high-priority, low-priority, and overall customers. As $\gamma$ increases, $L_{b_H}$ and $L_b$ decreases slightly, and $L_{b_L}$ increases gradually. This is because higher $\gamma_H$ makes high-priority customers more likely abandon the customer queue once they reach their impatience threshold, thereby releasing system capacity for low-priority customers to be served. As a result, more low-priority customers are able to form the low-priority. In addition, $L_{b_H}$ is smaller than $L_{b_L}$. This is because the high-priority customers have non-preemptive priority over low-priority customers in the customer queue and $\mu_{2_H}$ is larger than $\mu_{2_L}$, and therefore more low-priority customers remain waiting in the customer queue before being batched. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-142 illustrates the impact of the the impatience rate $\gamma_H$ on the blocking probabilities for high-priority, low-priority, and overall customers. As $\gamma_H$ increases, the blocking probability decrease steadily across all priority levels. This is because higher $\gamma_H$ makes high-priority customers more likely abandon the customer queue once they reach their impatience threshold, thereby reducing the probability that queue reaches its capacity and triggers blocking. In addition, $P_{b_H}$ is smaller than $P_{b_L}$. This is because the capacity limit of the customer queue for the high-priority customers is no smaller than that for low-priority customers. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-143 illustrates the impact of the impatience rate $\gamma$ on the system throughputs for high-priority, low-priority, and overall customers. As $\gamma$ increases, $T_{h_H}$ gradually decreases, while $T_{h_L}$ increases slightly, resulting in an almost stable $T_h$. This is because a higher $\gamma_H$ causes more high-priority customers to abandon the system before receiving service, reducing their contribution to throughput. Meanwhile, the released system capacity allows more low-priority customers to be served, slightly increasing their throughput. In addition, $T_{h_H}$ is smaller than $T_{h_L}$. This is because the customer arrival rate of the high-priority customers is smaller than that of low-priority customers. Lastly, the analytical results are in good agreement with the simulation results.

Figure 5-144 illustrates the impact of the impatience rate $\gamma_H$ on the impatience probabilities for high-priority, low-priority, and overall customers. As $\gamma_H$ increases, $P_{im_H}$ rises sharply, while $P_{im_L}$ decreases slightly. As a result, more high-priority customers abandon the queue due to the higher $\gamma_H$. In contrast, $P_{im_L}$ decreases because the early departure of high-priority customers frees up queue capacity, allowing more low-priority customers to be served. As described above, $P_{im}$ with impatience is

larger than without impatience. Lastly, the analytical results are in good agreement with the simulation results.

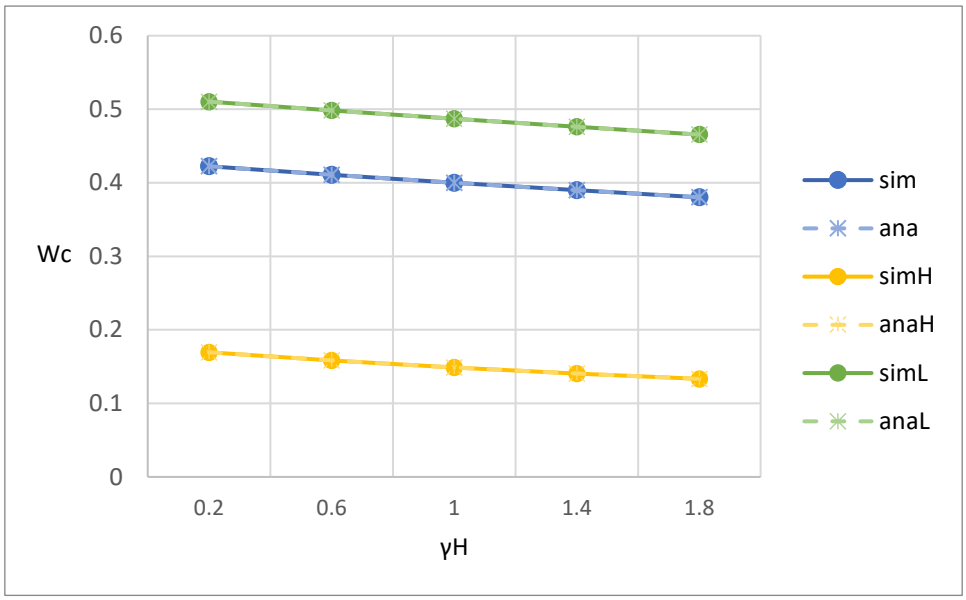


Figure 5-138: Effect of transition rate on average waiting time in the customer queue in the system

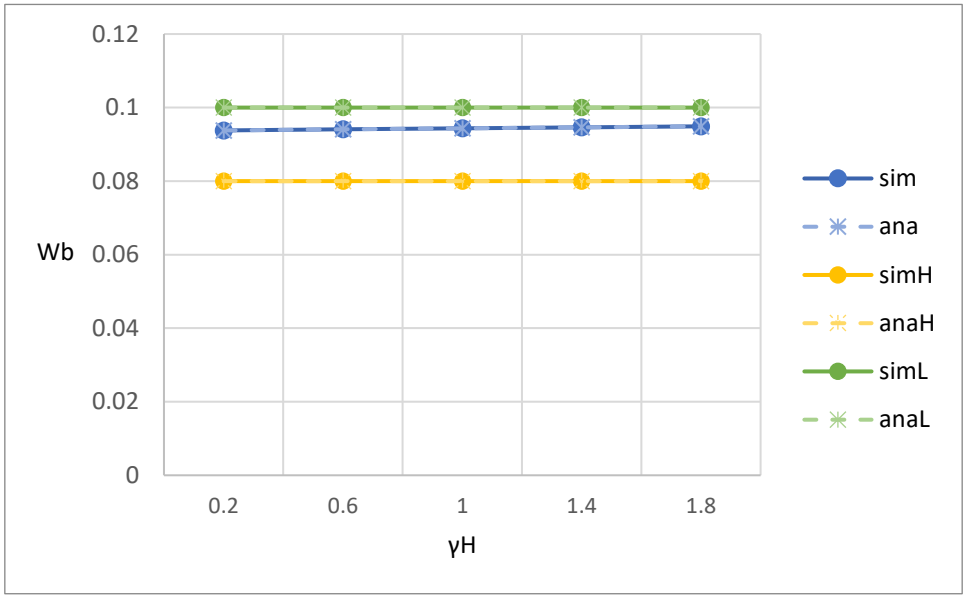


Figure 5-139: Effect of transition rate on average waiting time in the block queue

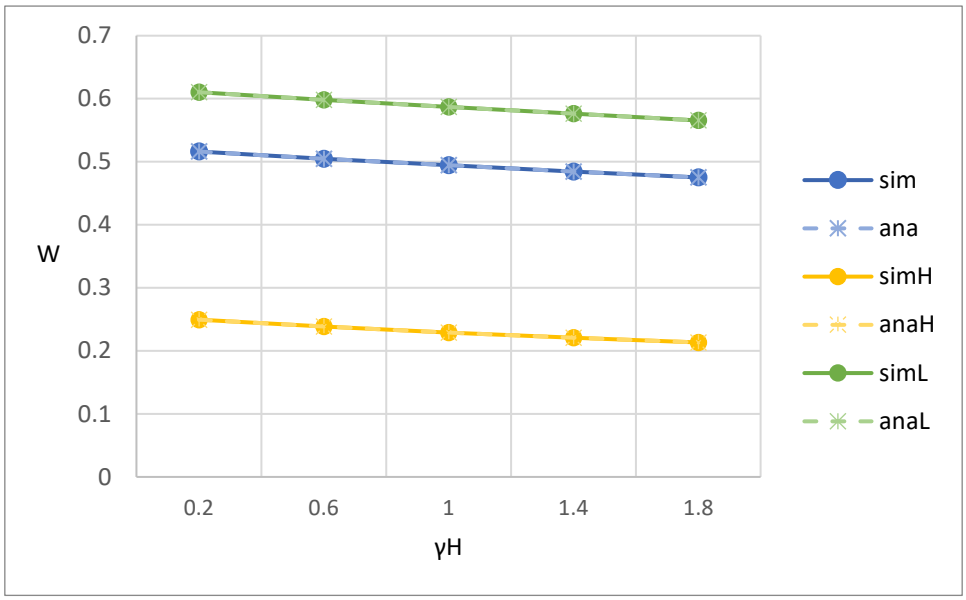


Figure 5-140: Effect of transition rate on average waiting time in the system

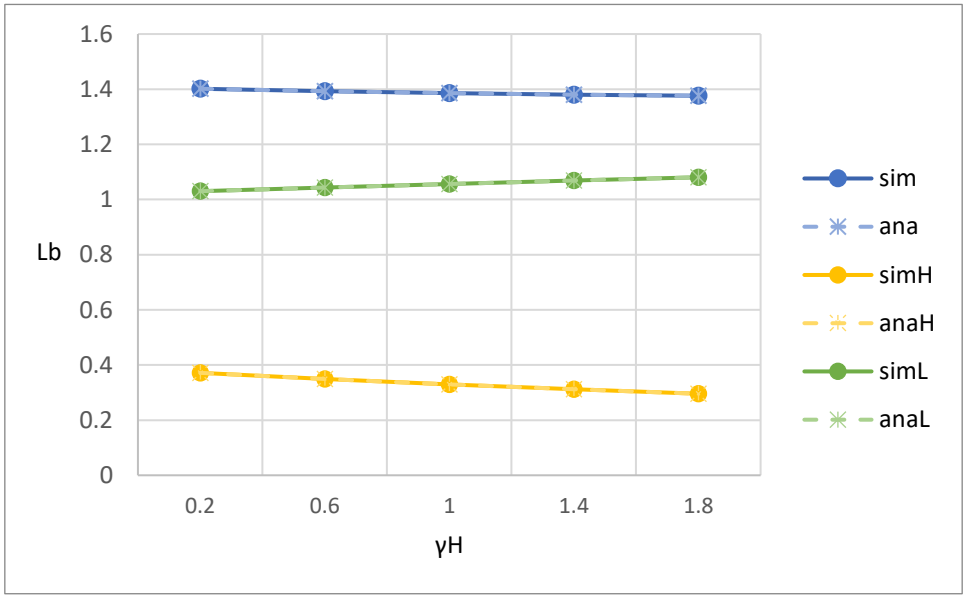


Figure 5-141: Effect of transition rate on average number of customers in block queue

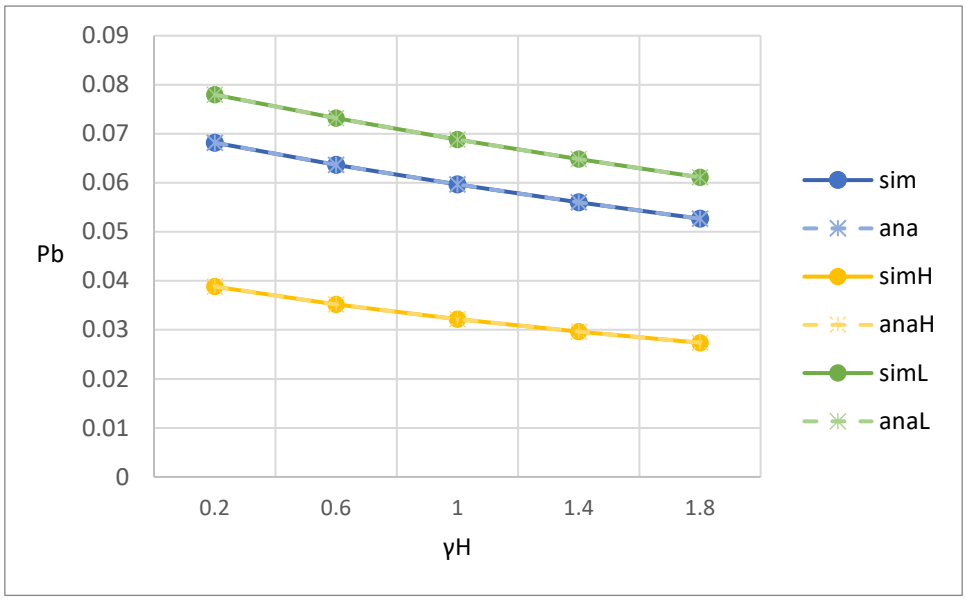


Figure 5-142: Effect of transition rate on blocking probability

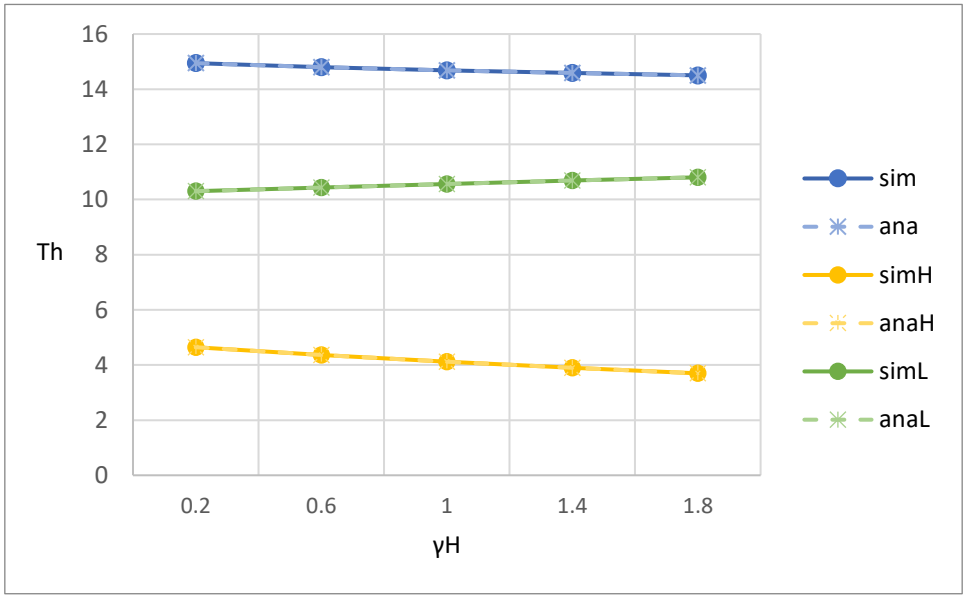


Figure 5-143: Effect of transition rate on system throughput

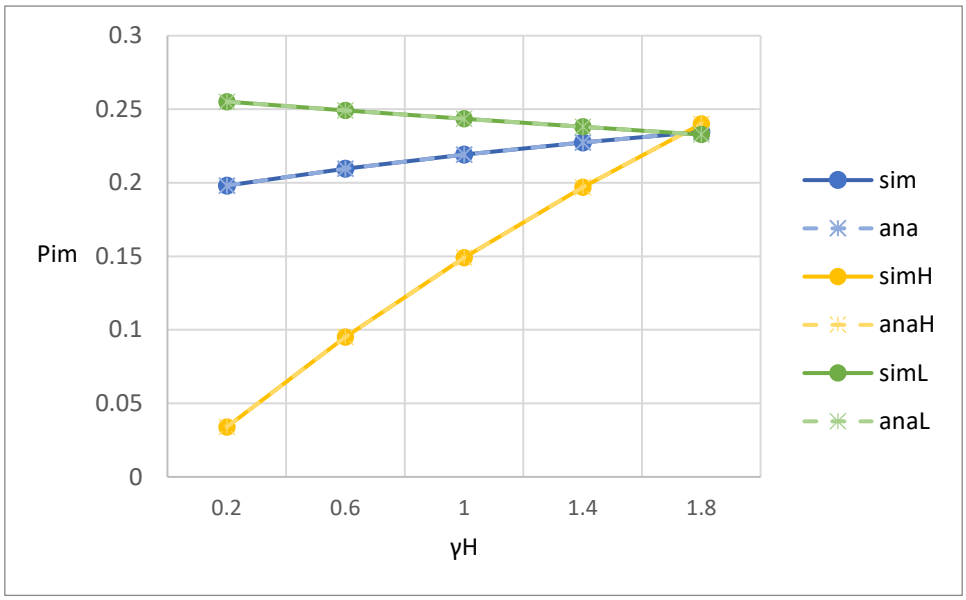


Figure 5-144: Effect of block size on the impatient probability

# 6. Conclusion

This research investigates a blockchain-based queuing system that models the transaction process under different combinations of customer priority and impatience, while incorporating ON/OFF operational states that reflect the stochastic availability of block generation and consensus phase. The blockchain queuing system comprises two queues in series: a customer queue, where customers wait to be grouped into a block, and a block queue, where grouped customers await consensus. The system supports a partial batch generation mechanism, allowing up to $b$ customers to be grouped at once. The ON/OFF states represent the operational availability of the system; during OFF periods, block generation and consensus are suspended, while arrivals continue. We model four scenarios: (1) single-class customers without impatience, (2) two-class customers without impatience, (3) Single-class customers with impatience, and (4) two-class customers with impatience.

For each scenario, we construct a multi-dimensional Markov chain that captures the state of the system. Balance equations are formulated and solved iteratively until the steady-state probability distribution is obtained. Based on these probabilities, key performance metrics are calculated, including throughput, blocking probability, and average waiting times. To validate our analytical findings, we develop a discrete-event simulation implemented in C++. The simulation strictly adheres to the queuing logic and service discipline of each scenario to ensure consistency with the analytical model.

In conclusion, based on our simulation and analytical results, we have made several interesting observations. First, as $b$ increases, $L_b$ initially grows but eventually stabilizes around a constant value. Second, in the priority-based scenarios, $T_{h_H}$ increases with $b$ but the increase gradually slows down. Third, when increasing $\lambda_H$ in the priority-based scenarios, $W_{b_H}$ and $W_{b_L}$ remains constant. However, $W_b$ shows a downward trend. Fourth, in the priority-base scenarios, increasing $\mu_{2_H}$ leads to an upward trend in $L_{b_L}$. Lastly, scenarios with impatience leads to improvements in most performance metrics compared to scenarios without impatience. Notably, however, $W_b$ remains unaffected by the impatience rate.

Future research directions include incorporating a more realistic voting mechanism into the consensus phase to better reflect practical blockchain operations. In addition, further extensions may involve optimizing the batch policy dynamically based on the real-time queue state, and enhancing the impatience modeling to capture the total time until departure, including both queueing and consensus delays.

# Reference

[1] S. R. Al-Hafidh and E. H. Al-Hemiary, "Integrating Blockchain With IoT-Edge-Cloud Netwotk for Tracking Offloaded Tasks," *2024 4th International Conference of Science and Information Technology in Smart Administration (ICSINTESA)*, Balikpapan, Indonesia, 2024, pp. 101-106, doi: 10.1109/ICSINTESA62455.2024.10747972.

[2] R. A. Memon, J. Li, J. Ahmed, A. Khan, M. I. Nazir, and M. I. Mangrio, "Modeling of Blockchain Based Systems Using Queuing Theory Simulation," *2018 15th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, Chengdu, China, 2018, pp. 107-111, doi: 10.1109/ICCWAMTIP.2018.8632560.

[3] T. Meng, Y. Zhao, K. Wolter, and C. -Z. Xu, "On Consortium Blockchain Consistency: A Queueing Network Model Approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 6, pp. 1369-1382, 1 June 2021, doi: 10.1109/TPDS.2021.3049915.

[4] Y. -X. Chang, Q. Wang, Q. -L. Li, Y. Ma, and C. Zhang, "Performance and Reliability Analysis for PBFT-Based Blockchain Systems With Repairable Voting Nodes," *IEEE Transactions on Network and Service Management*, vol. 21, no. 4, pp. 4039-4060, Aug. 2024, doi: 10.1109/TNSM.2024.3384506.

[5] P. Thakkar, S. Nathan and B. Viswanathan, "Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform," *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Milwaukee, WI, USA, 2018, pp. 264-276, doi: 10.1109/MASCOTS.2018.00034.

[6] Q. -L. Li, Y. -X. Chang, and C. Zhang, "Tree Representation, Growth Rate of Blockchain and Reward Allocation in Ethereum With Multiple Mining Pools," in *IEEE Transactions on Network and Service Management*, vol. 20, no. 1, pp. 182-200, March 2023, doi: 10.1109/TNSM.2022.3195292.

[7] T. Li, Y. Ren, and J. Xia, “Blockchain Queuing Model with Non-Preemptive Limited-Priority,” *Intell. Automat. Soft Comput.*, vol. 26, no. 5, pp. 1111–1122, 2020.

[8] Q. Li, Y. Ma, J. Ma, and Y. Chang. “Information Theory of Blockchain Systems.” *Lecture Notes in Computer Science*, vol 14462. pp. 443-454,2024

[9] M. S. Peelam, B. K. Chaurasia, A. K. Sharma, V. Chamola and B. Sikdar, "Unlocking the Potential of Interconnected Blockchains: A Comprehensive Study of Cosmos Blockchain Interoperability," *IEEE Access*, vol. 12, pp. 171753-171776, 2024, doi: 10.1109/ACCESS.2024.3497298.

[10] O. Wu, S. Li, Y. Wang, H. Li, and H. Zhang, "Modeling Cross-blockchain Process Using Queueing Theory: The Case of Cosmos," *2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS)*, Nanjing, China, 2023, pp. 274-281, doi: 10.1109/ICPADS56603.2022.00043.