# 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 users before block generation, and the consensus queue, which represents the stage where users 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 and , so that the total arrival rate satisfies . After arriving at the customer queue, users wait for the block generation process, which occurs at a rate of (or and in the two-class case). Once a block is formed, a group of users is transferred to the consensus queue, where the consensus process is carried out at a service rate denoted by (or and 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 for single-class users, and rates and for high-priority and low-priority users, respectively. Once a customer enters the consensus 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 (ON to OFF) and (OFF to ON) respectively.

一張含有 黑色, 黑暗 的圖片

AI 產生的內容可能不正確。

Figure 3‑1

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 consensus 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 |  |  |
|  |
| BLock generation rate |  |  |
|  |
| Consensus rate |  |  |
|  |
| Impatient rate |  |  |
|  |
| Transition rate (ON to OFF) |  |  |
| Transition rate (OFF to ON) |  |  |

Table 3.1 The parameters used in different scenarios

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

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 and , respectively.

### State Balance Equations

The system under consideration is described as a three-dimensional Markov chain with state denoted by , where denotes the number of customers in the customer queue, denotes the number of customers in the consensus queue, and denotes the system state. When , the maximum number of customers in the customer queue is . When , meaning that the consensus queue is occupied, the maximum number of customers allowed in the customer queue is reduced to . The system state 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 , 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:

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

For example, if and , the number of feasible states is 862. The steady state probability of state is denoted as . In this scenario, the feasible states can be categorized into 16 distinct cases, as described below.

1. System off,
2. System on,

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.

一張含有 文字, 螢幕擷取畫面, 圓形, 字型 的圖片

AI 產生的內容可能不正確。

Figure 3‑2 The state transition diagram of Case 11:

### Iterative Algorithm

We use the iterative algorithm provided below, we 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 for all , where is the total number of feasible states.

**Step 2**: Substitute into the balance equations from Case 1 to Case 16 to find , .

**Step 3**: Normalize , .

**Step 4**: If , then stop the iteration. Otherwise, set , , and return to **Step 2**.

In our analysis, the convergence threshold is set to , and the algorithm typically converges after about 75 iterations.

### Performance Measure

After obtaining the steady-state probabilities 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, denote by , is given by:

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

Third, the average number of customers in consensus queue, denoted by , is given by:

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

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

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

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

Eighth, the average waiting time in the consensus queue, denoted by , is given by:

Finally, the average number of blocks participating in the consensus process, denoted by , is given below.

## 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 and , 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, users wait for the block generation process, which occurs at a rate of and , respectively. Each block is generated according to the partial batch policy, i.e., each block can contain 1 to customers of the same policy class. Once a block is formed, it is transferred to the consensus queue, where the consensus process is carried out at the service rate denoted by and , 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 and , respectively.

### State Balance Equations

The system under consideration is described as a five-dimensional Markov chain denoted by , where and represent the number of high-priority and low-priority customers in the customer queue, respectively. and represent the number of high-priority and low-priority customers in the consensus queue, respectively. And denotes the system state. When the consensus queue is empty (i.e., and ), the maximum number of customers allowed in the customer queue is , implying that . However, when the consensus queue is occupied (i.e., or ), the maximum number of customers in the customer queue is reduced to , and thus .

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 consensus queue as a batch for processing without preemption.

The system state 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 , 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:

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

For example, if and , the number of feasible states is 22524. The steady state probability of state is denoted as . In this scenario, the feasible states can be categorized into 99 distinct cases, as described below.

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.

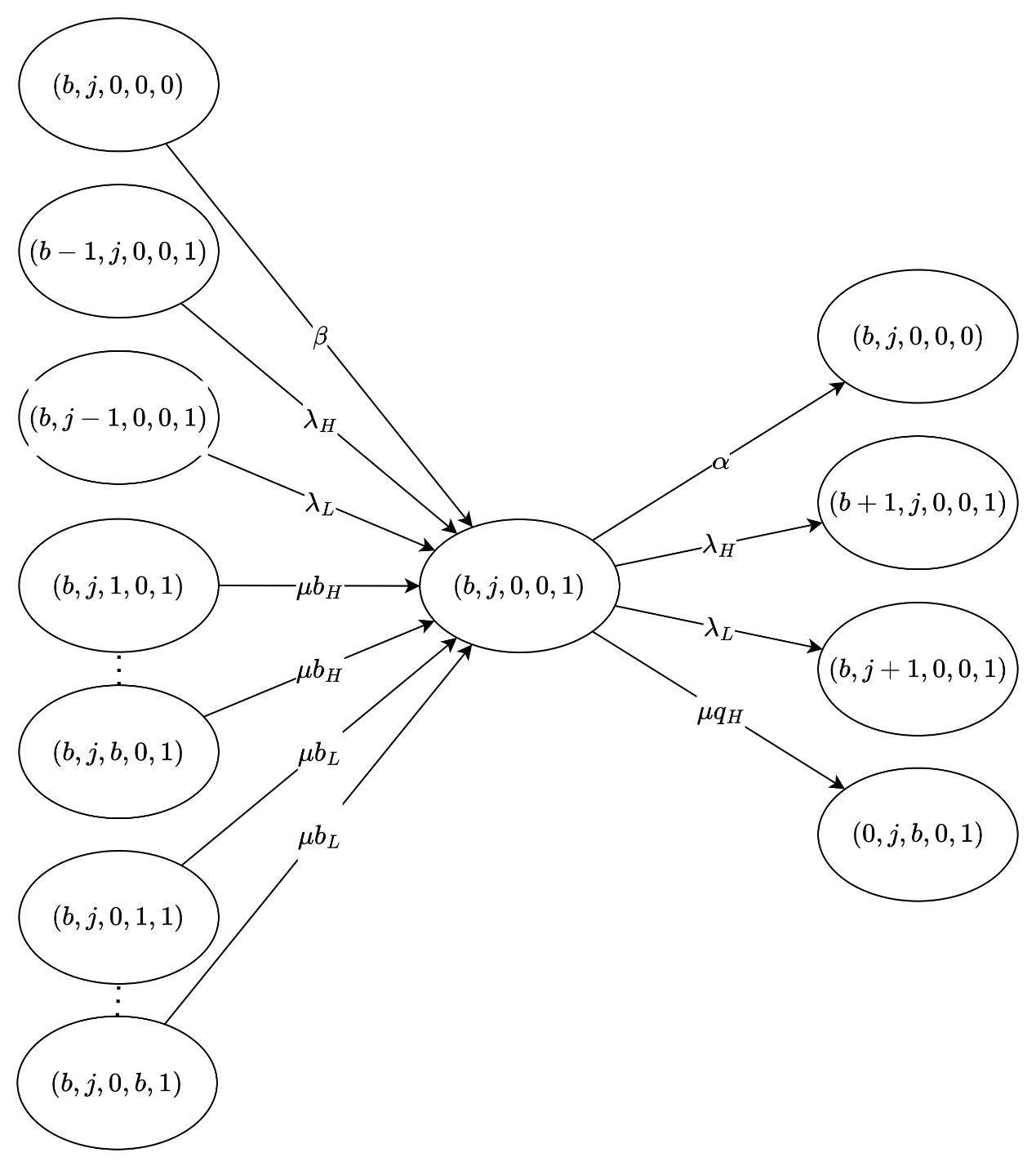
****

Figure 3‑3 The state transition diagram of Case 45:

### Iterative Algorithm

We use the iterative algorithm provided below, we 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 for all , where is the total number of feasible states.

**Step 2**: Substitute into the balance equations from Case 1 to Case 16 to find , .

**Step 3**: Normalize , .

**Step 4**: If , then stop the iteration. Otherwise, set , , and return to **Step 2**.

In our analysis, the convergence threshold is set to , and the algorithm typically converges after about 80 iterations.

### Performance Measure

After obtaining the steady-state probabilities 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, denote by and , respectively, is given by:

The average number of customers in the whole system, denote by , is given by:

Second, the average number of high-priority and low-priority customers in customer queue, denoted by and , respectively, is given by:

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

Third, the average number of high-priority and low-priority customers in consensus queue, denoted by and , respectively, is given by:

The average number of customers in consensus queue, denoted by , is given by:

Fourth, the blocking probability of high-priority and low-priority customers in the system, denoted by and , respectively, is given by:

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

Fifth, the throughput of high-priority and low-priority customers in the system, denoted by and , respectively, is given by:

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

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

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

Eighth, the average waiting time of the high-priority and low-priority customers in the customer queue, denoted by and , respectively, is given by:

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

Ninth, the average waiting time of the high-priority and low-priority customers in the consensus queue, denoted by and , respectively, is given by:

The average waiting time in the consensus queue, is given by:

Finally, the average number of high-priority and low-priority blocks participating in the consensus process, denoted by and , is given by:

The average number of customers participating in the consensus process within a block, denoted by , is given by:

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

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 . Customers in the consensus 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 and , respectively.

### State Balance Equations

The system under consideration is described as a three-dimensional Markov chain with state denoted by , where denotes the number of customers in the customer queue, denotes the number of customers in the consensus queue, and denotes the system state. When , the maximum number of customers in the customer queue is . When , meaning that the consensus queue is occupied, the maximum number of customers allowed in the customer queue is reduced to .

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 , and abandonment occurs only while the customer is waiting in the customer queue.

The system state 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 , 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:

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

For example, if and , the number of feasible states is 862. The steady state probability of state is denoted as . In this scenario, the feasible states can be categorized into 16 distinct cases, as described below.

1. System off,
2. System on,

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.

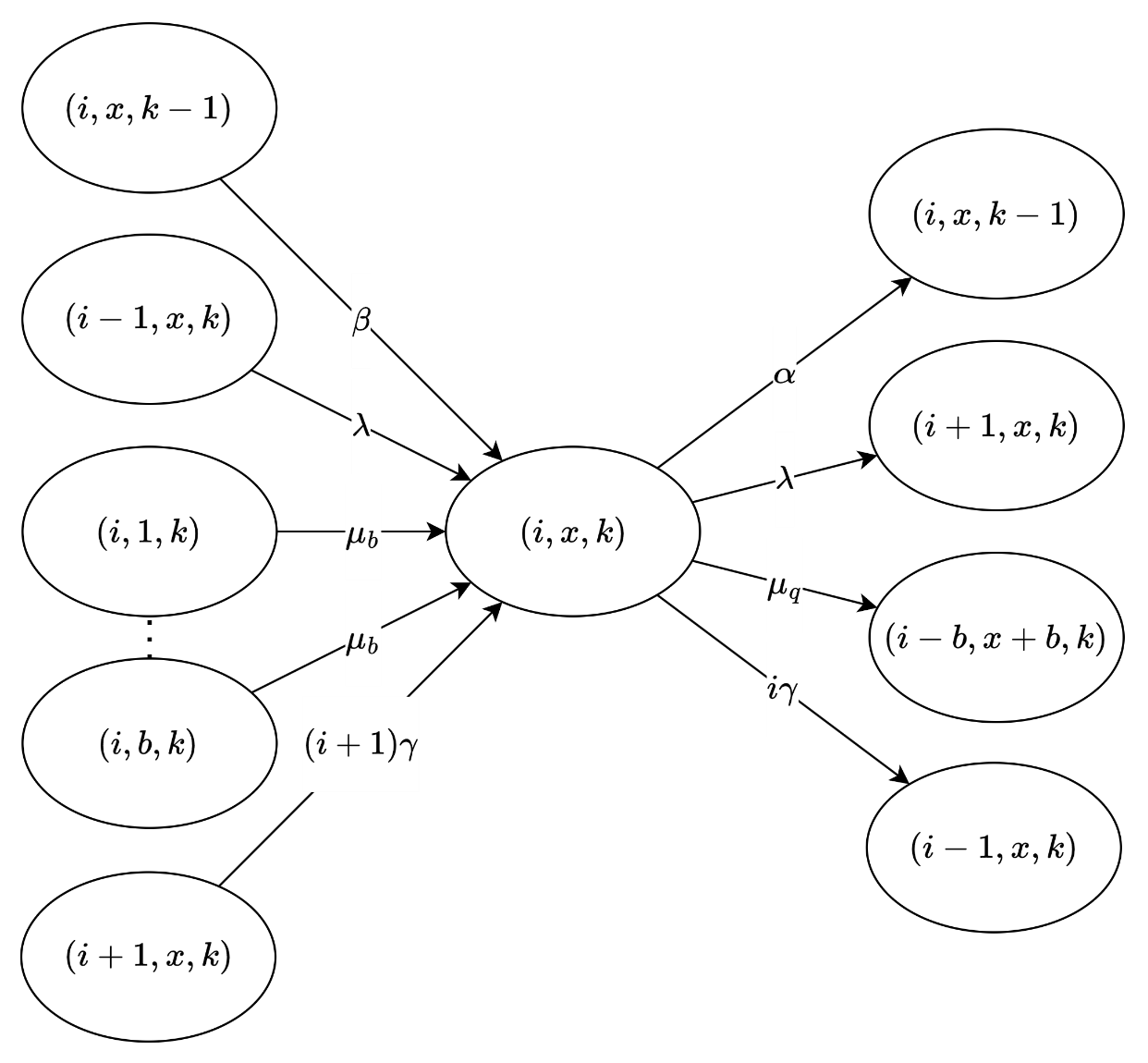


Figure 3‑4 The state transition diagram of Case 12:

### Iterative Algorithm

We use the iterative algorithm provided below, we 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 for all , where is the total number of feasible states.

**Step 2**: Substitute into the balance equations from Case 1 to Case 16 to find , .

**Step 3**: Normalize , .

**Step 4**: If , then stop the iteration. Otherwise, set , , and return to **Step 2**.

In our analysis, the convergence threshold is set to , and the algorithm typically converges after about 72 iterations.

### Performance Measure

After obtaining the steady-state probabilities 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, denote by , is given by:

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

Third, the average number of customers in consensus queue, denoted by , is given by:

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

Fifth, the impatient probability of the system, denoted by , is given by:

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

Seventh, the average waiting time in the system is given by:

Eighth, the average waiting time in the customer queue, denoted by , is given by:

Ninth, the average waiting time in the consensus queue, denoted by , is given by:

Finally, the average number of blocks participating in the consensus process, denoted by , is given below.

## 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 and , 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, users wait for the block generation process, which occurs at a rate of and , respectively. Each block is generated according to the partial batch policy, i.e., each block can contain 1 to customers of the same policy class. Once a block is formed, it is transferred to the consensus queue, where the consensus process is carried out at the service rate denoted by and , 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​ and ​ for high-priority and low-priority customers, respectively. Customers in the consensus 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 and , respectively.

### State Balance Equations

The system under consideration is described as a five-dimensional Markov chain denoted by , where and represent the number of high-priority and low-priority customers in the customer queue, respectively. and represent the number of high-priority and low-priority customers in the consensus queue, respectively. And denotes the system state. When the consensus queue is empty (i.e., and ), the maximum number of customers allowed in the customer queue is , implying that . However, when the consensus queue is occupied (i.e., or ), the maximum number of customers in the customer queue is reduced to , and thus .

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 consensus queue as a batch for processing without preemption.

The system state 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 , 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:

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

For example, if and , the number of feasible states is 22524. The steady state probability of state is denoted as . In this scenario, the feasible states can be categorized into 99 distinct cases, as described below.

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

### Iterative Algorithm

We use the iterative algorithm provided below, we 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 for all , where is the total number of feasible states.

**Step 2**: Substitute into the balance equations from Case 1 to Case 16 to find , .

**Step 3**: Normalize , .

**Step 4**: If , then stop the iteration. Otherwise, set , , and return to **Step 2**.

In our analysis, the convergence threshold is set to , and the algorithm typically converges after about 80 iterations.

### Performance Measure