# 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 customer 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 Impatient

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

### State Balance Equations

The system under consideration is described as a three-dimensional Markov chain denoted by , where denotes the number of customers in the customer queue, denotes the number of customers in the consensus queue, denotes the system state. When , the maximum value of is the full capacity of the queue, denoted by . However, 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 are allowed to 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 number of feasible states is as follows:

For example, if is equals to 40, and is equals to 15, the number of feasible states is 862. In this scenario, the feasible states can be categorized into 13 distinct cases, as described below.

1. System off,

Case 1:

Case 2:

Case 3:

Case 4:

Case 5:

Case 6:

1. System on,

Case 7:

Case 8:

Case 9:

Case 10:

Case 11:

Case 12:

Case 13:

Case 14:

Case 15:

Case 16: