2. System Model

We aim to study four blockchain scenarios, each involving two queues of finite capacity: the **customer queue** and the **consensus queue**. The maximum capacity of the customer queue is denoted by N, while the consensus queue, which represents the block currently undergoing consensus, 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 consensus queue to undergo the consensus process.

This model is based on a **partial batch mechanism** derived from queuing theory. When the consensus queue becomes idle, if there are more than b customers in the customer queue, the first b customers are selected and moved to the consensus queue. If there are b or fewer customers waiting, all of them are transferred instead. After the consensus process concludes, regardless of whether the result is successful, all customers in the consensus queue leave the system.

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.

**Scenario 1: Single-Class Customers without Impatience**

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

**Scenario 2: Two-Class Customers without Impatience**

In the second scenario, we assumes there are two types 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-Serve (FCFS) discipline, and high-priority customers are always placed ahead of low-priority customers in the customer queue. Note that for high-priority customers, the maximum capacity of the customer queue is N when the consensus 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 consensus queue is idle or not.

**Scenario 3: Single-Class Customers with Impatience**

The third scenario considers a single type of customers as in Scenario 1, but introduces customer impatience. Customers still follow the First-Come-First-Serve (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 consensus queue, impatience is no longer considered. The capacity rule of the customer queue remains the same as in Scenario 1, depending on the status of the consensus queue.

**Scenario 4: Two-Class Customers with Impatience**

In the fourth scenario, we again consider two types of customers—high-priority and low-priority—similar to Scenario 2. Customers of the same priority follow the First-Come-First-Serve (FCFS) discipline, and high-priority customers are given precedence over low-priority customers in the queue. The key difference from Scenario 2 lies in the addition of customer impatience: customers from both priority classes may leave the queue if they wait too long. Each priority class may have its own impatience rate. As with the previous scenario, impatience is no longer relevant once customers enter the consensus queue. The queue capacity rules remain consistent with those in Scenario 2.

The comparison between models is shown in the table below. Queue capacity depends on the status of the consensus queue

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Model 1 | Model 2 | Model 3 | Model 4 |
| Customer Types | Single | High / Low | Single | High / Low |
| Priority | - | High before Low | - | High before Low |
| Impatient | No | No | Yes | Yes |
| Queue Capacity | Idle: N  Not idle: N-b | High: N or N-b  Low: always N-b | Same as Scenario 1 | Same as Scenario 2 |