# 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 , while the consensus queue, which represents the block currently undergoing consensus, has a capacity of . 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.

The blocking generation process is based on a **partial batch service mechanism**. When the consensus queue becomes idle, if there are more than customers in the customer queue, the first customers are selected and moved to the consensus queue. If there are 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 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 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 consensus queue is empty, at most 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 .

## 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 when the consensus queue is idle, and when it is not idle. On the other hand, for low-priority customers, the maximum capacity of the customer queue is always , regardless of whether the consensus queue is idle or not.

## 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 consensus queue, impatience is no longer considered. It is noted that if the consensus queue is empty, at most 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 .

## 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 consensus queue. Note that for high-priority customers, the maximum capacity of the customer queue is when the consensus queue is idle, and when it is not idle. On the other hand, for low-priority customers, the maximum capacity of the customer queue is always , regardless of whether the consensus queue is idle or not.