Suzuki–Kasami's Broadcast Algorithm

Group 6

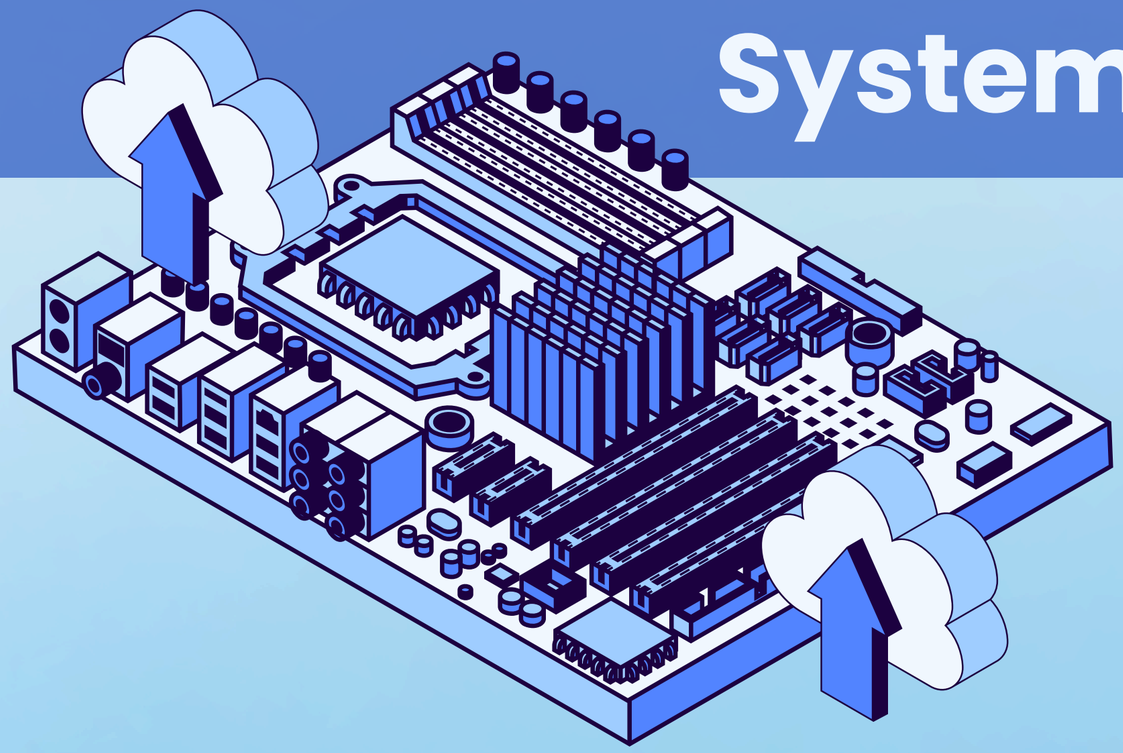
Introduction

- A distributed mutual exclusion algorithm
- Ensures only one site accesses the Critical Section (CS) at a time
- Uses a single token shared among all sites
- Designed for distributed systems and databases

Why Mutual Exclusion?

- To avoid data inconsistency
- To prevent simultaneous writes
- To maintain database integrity

System Model & Assumptions



Assumptions

- There are N sites (processes): S_1, S_2, ..., S_n
- Messages are delivered reliably
- Only one token exists in the system
- A site must possess the token to enter CS

Communication

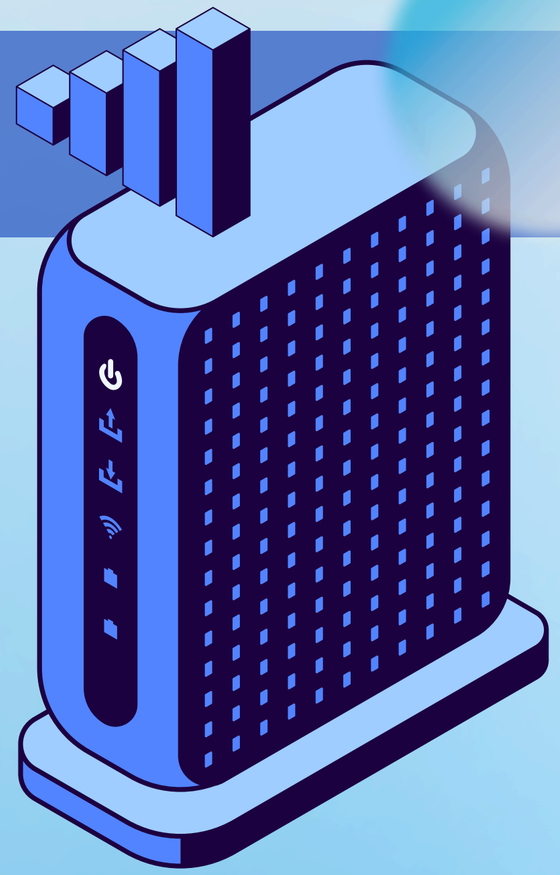
- Uses broadcast messages
- REQUEST messages are sent to all sites

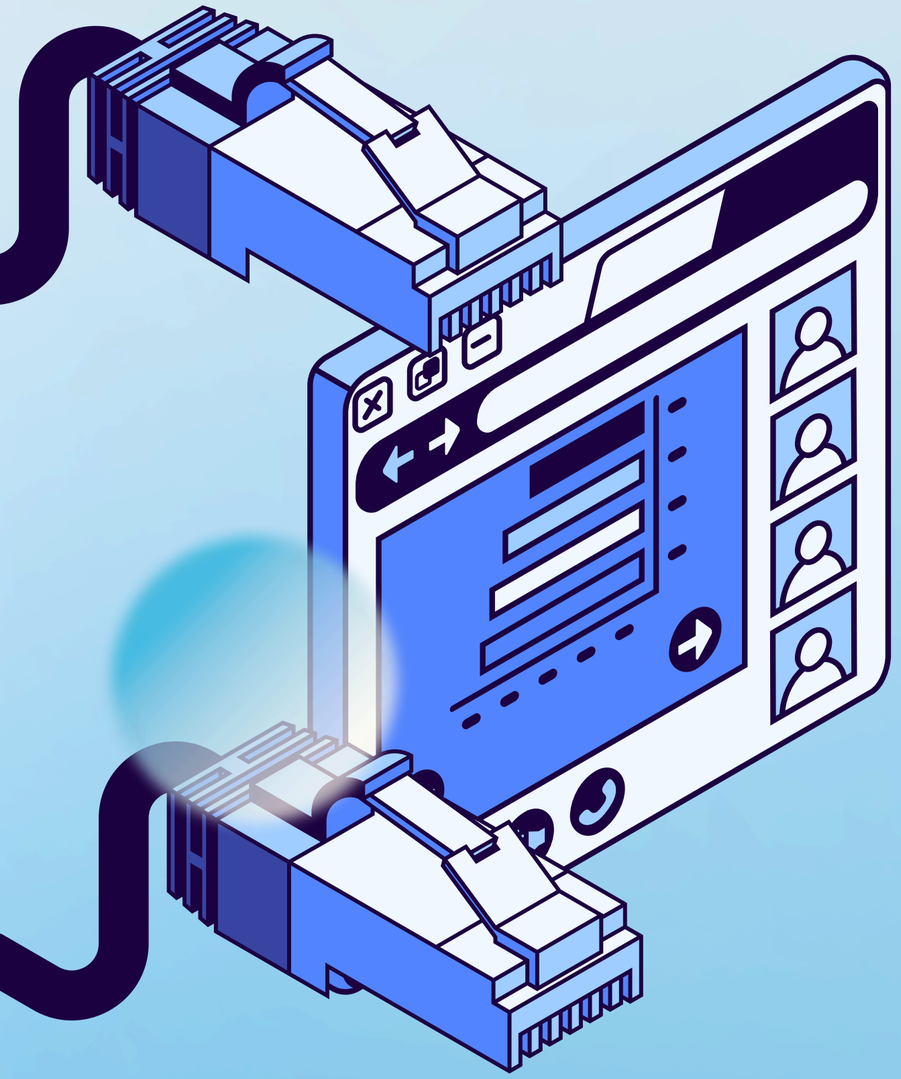
Basic Idea of the Algorithm

- If a site wants to enter CS:
- If it has the token → enter CS
- Else → broadcast REQUEST message
- The site holding the token sends it to the requester
- Token movement controls access

Problems to Be Addressed

- **How to distinguish old and new REQUEST messages?**
  - Use sequence numbers
- **How to identify which site is waiting for CS?**
  - Maintain request tracking arrayst





REQUEST Message Format

REQUEST(j, n)

- j → Site ID requesting CS
- n → Sequence number of the request

RN Array (Request Number Array)

RN Array

Each site S_i maintains:

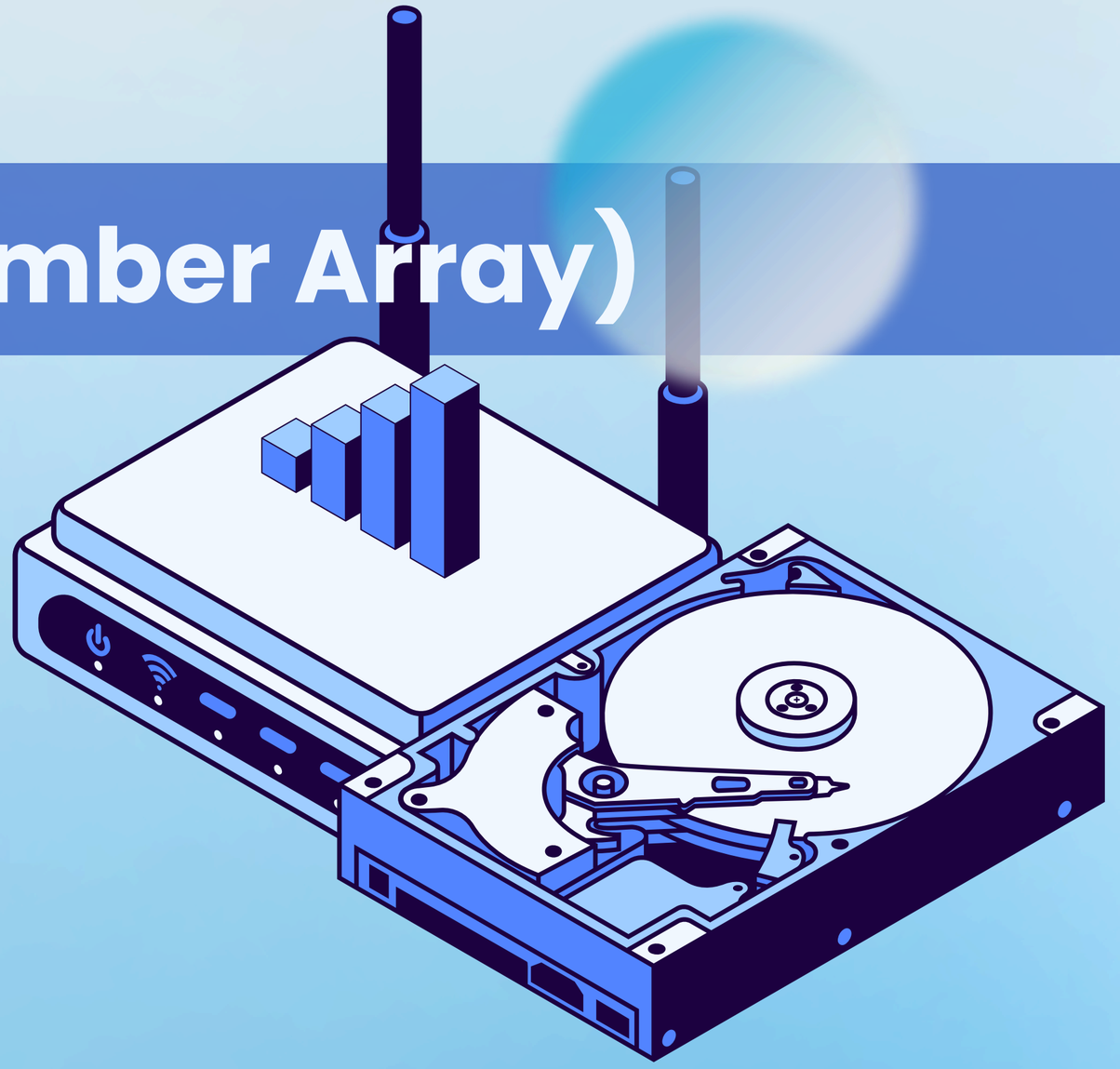
$$RN_i[1 \dots N]$$

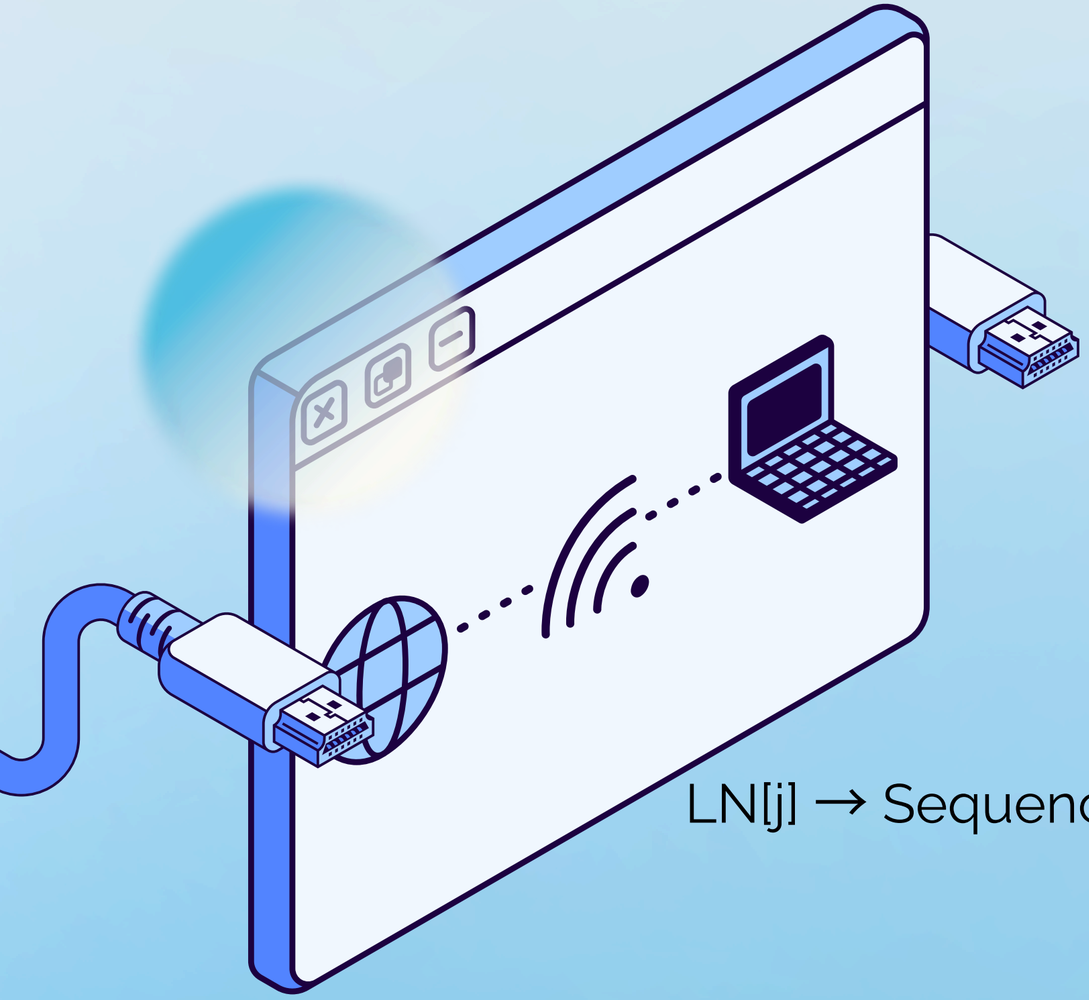
Update Rule

$$RN_i[j] = \max(RN_i[j], n)$$

Outdated Request

- REQUEST(j, n) is outdated if $RN_i[j] > n$





Token Structure

Token Contains

- **Queue Q**

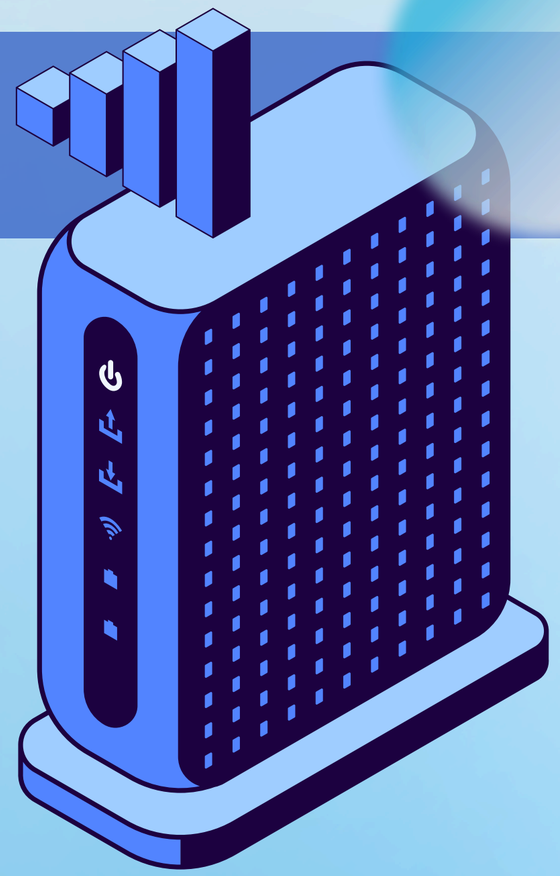
  Stores site IDs waiting for CS

- **LN Array**

LN[1 … N]

LN[j] → Sequence number of last request executed by site S_j

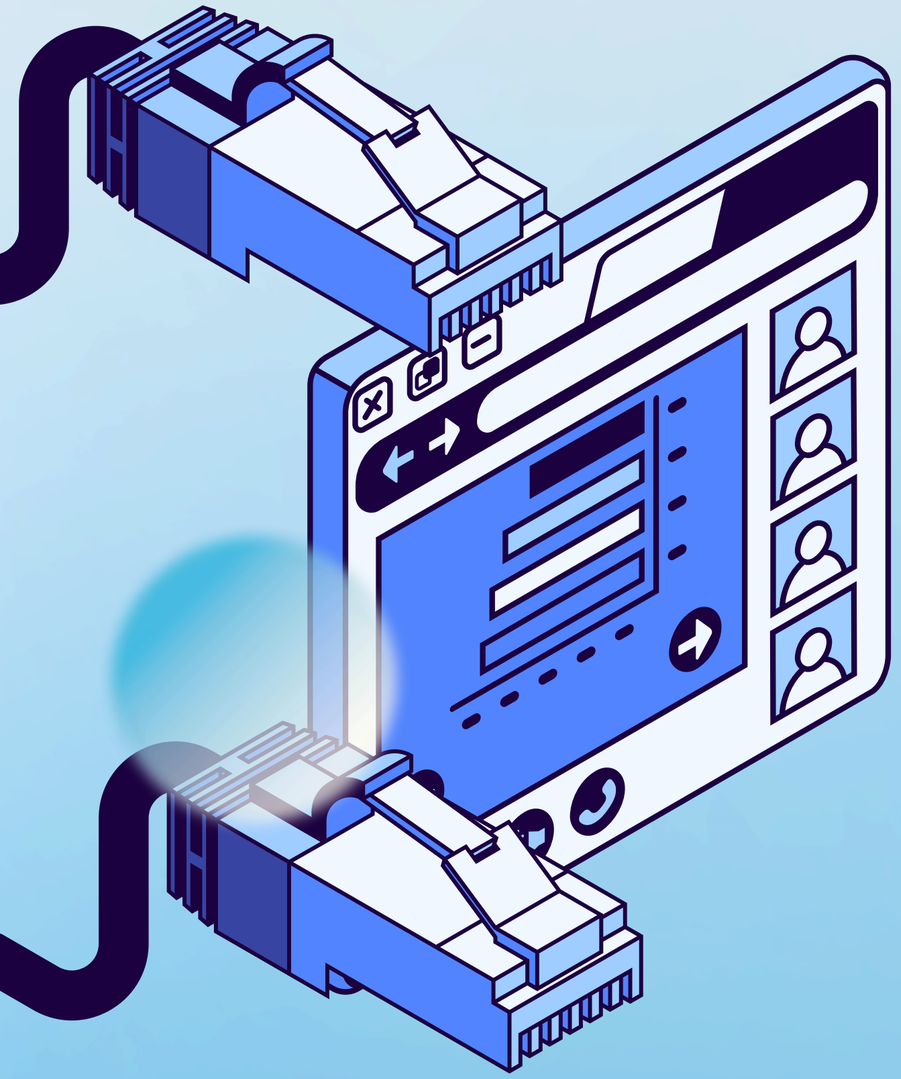
Requesting the Critical Section



1. If site S_i does not have the token:

    1.1 Increment $RN_i[i]$
    1.2 Broadcast REQUEST(i, $RN_i[i]$)

2. Receiving site updates RN
3. Token holder sends token if idle and request is valid



Executing the Critical Section

- Site enters CS only after receiving token
- Mutual exclusion is guaranteed
- No other site can enter CS simultaneously

Releasing the Critical Section

- **After CS Execution**

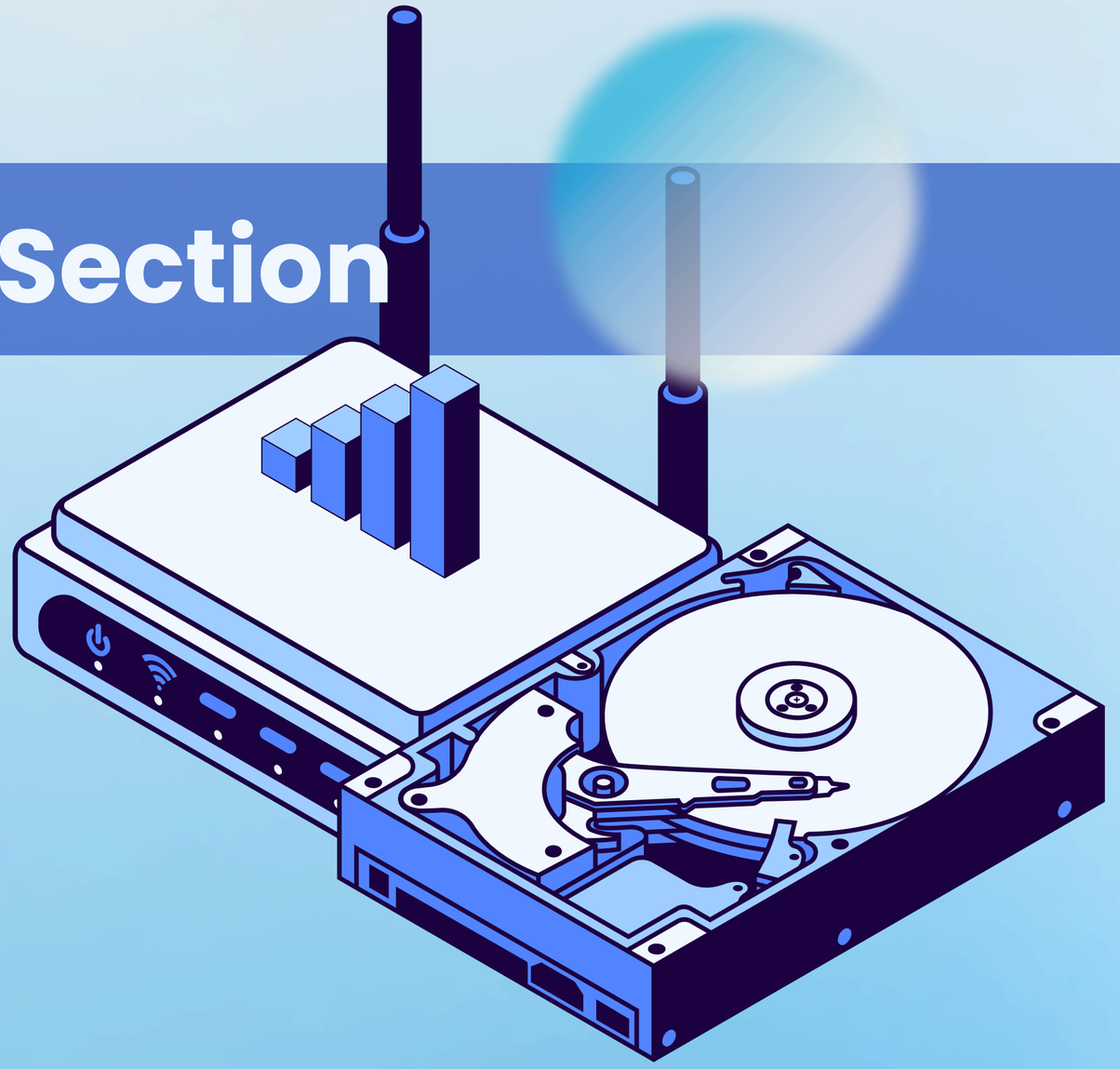
  Update:

  $LN[i] = RN[i]$

- **Check for pending requests**

  $RN[j] = LN[j] + 1$

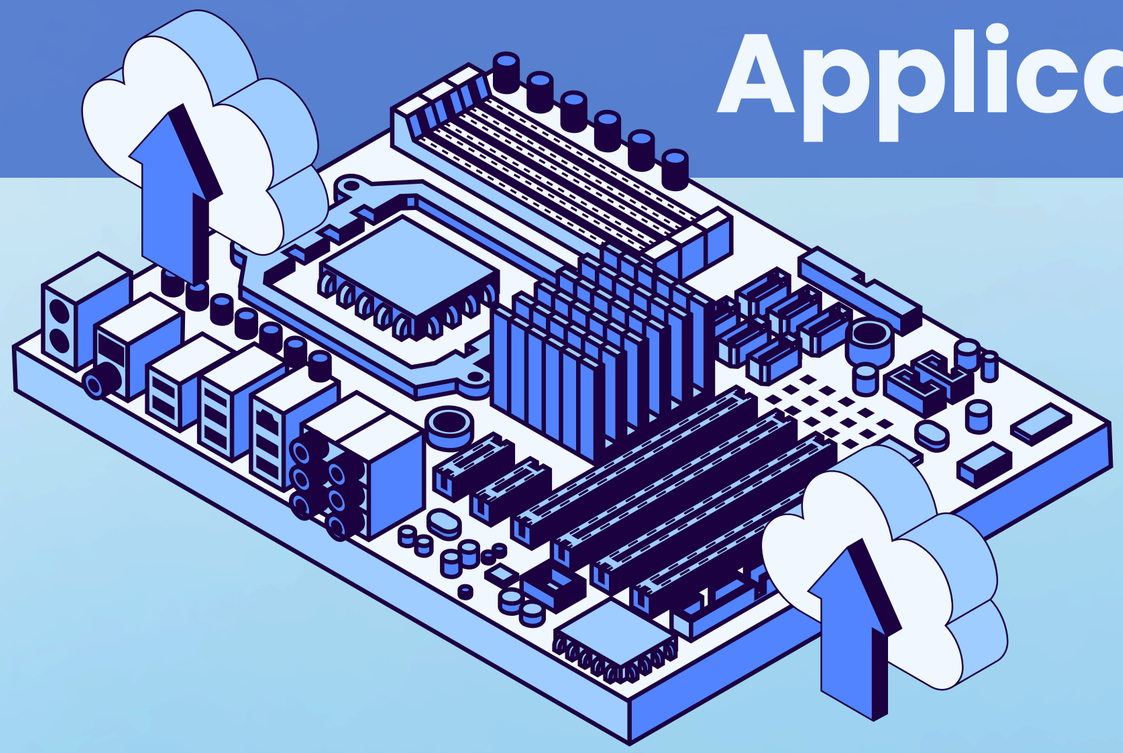
- **Add requesting sites to queue Q**
- **Send token to the head of the queue**



Algorithm Characteristics

- Token-based
- Fair (FIFO order)
- Avoids starvation
- Efficient under low contention

Applications



- Distributed databases
- Distributed file systems
- Resource sharing systems

Thank You!