2. Ring Algorithm for Distributed File System.

How the Ring Algorithm Works in a Distributed File System

1. **Logical Ring Formation**
   All file server nodes in the distributed file system are logically arranged in a **ring structure**.
   Each node knows only its **next node** in the ring.
2. **Token Circulation**
   A special control message called a **token** continuously circulates around the ring.
   At any moment, **only one token exists** in the system.
3. **File Access Control**
   - When a node wants to **read or write a shared file**, it must wait until it receives the token.
   - The node holding the token is allowed to enter the **critical section**, i.e., access or modify the file.
   - Other nodes must wait, ensuring **mutual exclusion**.
4. **Releasing the File**
   After completing the file operation, the node releases the file and **passes the token** to the next node in the ring.
5. **Failure Handling**
   If a node fails and the token is lost, the system can **regenerate the token** using a predefined recovery mechanism to continue file operations.

Why Ring Algorithm is Suitable for Distributed File Systems

- Prevents **simultaneous modification** of files
- Ensures **fair access** to shared files
- Eliminates the need for a **central coordinator**
- Reduces message overhead compared to broadcast-based algorithms

---

Example Scenario

In a distributed file system with 5 file servers:

- Only the server holding the token can update a shared configuration file.

- Once done, it passes the token to the next server.
- This guarantees that **no two servers write to the file at the same time**.

Need and Importance of the Algorithm

In a distributed file system, multiple nodes share files and resources. Proper coordination is required to avoid conflicts such as simultaneous access to the same file or resource.
The **Ring Algorithm** is important because it provides a **simple and fair mechanism** for coordination, leader election, or mutual exclusion among distributed nodes.
It ensures **ordered communication**, **fault tolerance**, and **equal opportunity** for all nodes in the system.

Description of the Algorithm

In the Ring Algorithm, all nodes in the distributed system are logically arranged in a **ring topology**. Each node has a unique identifier and knows only its **next neighbor** in the ring.

- A special message called a **token** or control message circulates around the ring.
- Only the node holding the token is allowed to perform a critical operation such as accessing or updating a file.
- After completing its operation, the node passes the token to the next node.
- In case of leader election, the node with the **highest ID** is selected as the coordinator after one full round of message passing.

This approach avoids centralized control and reduces message collisions.

Real-Time Applications of the Algorithm

- Distributed file systems for **controlled file access**
- **Leader election** in distributed databases
- **Mutual exclusion** in distributed operating systems
- Token-based **resource sharing systems**
- Distributed systems requiring **fair scheduling**

Requirements to Implement the Algorithm

- Logical **ring topology** among nodes
- Unique **node identifiers**
- Reliable **message passing mechanism**
- Ability to detect **node failures**
- Mechanism for **token generation and recovery**