2. Token Ring Algorithm Report

Aim:

To implement a token ring algorithm for achieving mutual exclusion in a distributed system, allowing processes to access a critical section by passing a unique token in a logical ring structure.

Theory:

The Token Ring algorithm is a simple yet effective method for achieving mutual exclusion in a distributed system. In this algorithm, processes are arranged in a logical ring, and a unique token is passed around the ring. Only the process that holds the token can enter the critical section. When the process exits the critical section, it passes the token to the next process in the ring.

How the Token Ring Algorithm Works:

1. **Initialization:**
    - A single token is created and given to the first process in the ring.
2. **Requesting Entry to the Critical Section:**
    - A process that holds the token checks if it wants to enter the critical section.
    - If yes, it enters the critical section; otherwise, it immediately passes the token to the next process in the ring.
3. **Passing the Token:**
    - After completing its critical section, the process passes the token to the next process in the logical ring.
    - The next process in turn checks if it needs to enter the critical section and repeats the steps.
4. **Handling Token Loss:**
    - If the token is lost or a process fails, a new token must be generated by one of the processes after detecting the loss.

Performance Evaluation:

1. **Message Complexity:**
    - The message complexity of the Token Ring algorithm is `O(1)` per critical section entry, as only one message (token passing) is required.
    - The overall complexity depends on the number of processes, as the token must circulate the entire ring.
2. **Response Time:**
    - The response time is generally low in small rings but increases linearly with the number of processes in the ring.
    - Each process must wait for the token to traverse the entire ring in the worst-case scenario.
3. **Scalability:**
    - The algorithm scales linearly with the number of processes. As the number of processes increases, the time taken for the token to circulate around the ring also increases.
    - Suitable for small to medium-sized distributed systems but can become less efficient with a large number of processes.
4. **Robustness:**
    - The algorithm is moderately robust against process failures. If a process fails while holding the token, the token is lost, and recovery mechanisms must be implemented.
    - If a process fails without holding the token, the system can continue to function normally.
5. **Handling Dynamic Process Joining:**
    - Dynamic process joining is straightforward; a new process can be added to the ring by updating the pointers of neighboring processes.
    - The new process will be included in the next cycle of token passing.
6. **Advanced Fault Tolerance:**
    - For enhanced fault tolerance, the system can implement a token regeneration mechanism where one of the processes is designated to recreate the token if it detects that the token is lost.
    - Additional checks such as heartbeat messages or acknowledgments can be used to detect failures and take corrective actions.

Output Analysis:

- The output confirms that the token is correctly passed around the ring and that only one process at a time can enter the critical section.
- Logs demonstrate the sequence of token passing and the order in which processes access the critical section, indicating that the algorithm maintains mutual exclusion effectively.

Bonus:

1. **Handling Dynamic Process Joining:**
    a. Dynamic process joining is straightforward; a new process can be added to the ring by updating the pointers of neighboring processes.
    b. The new process will be included in the next cycle of token passing.
2. **Advanced Fault Tolerance:**
    a. For enhanced fault tolerance, the system can implement a token regeneration mechanism where one of the processes is designated to recreate the token if it detects that the token is lost.
    b. Additional checks such as heartbeat messages or acknowledgments can be used to detect failures and take corrective actions.

Conclusion:

The Token Ring algorithm is a simple and efficient solution for achieving mutual exclusion in small to medium-sized distributed systems. Its low message complexity and straightforward implementation make it ideal for scenarios with a moderate number of processes. However, the linear scalability and need for additional mechanisms to handle token loss and process failures limit its suitability for larger systems. Implementing advanced fault tolerance techniques can improve the robustness and reliability of the algorithm.