Cloud computing assignment

Name: Urmil Vora
Roll No.: 23CS06018
Name: Rajnish Maurya
Roll No.: 23CS06009
Name: Bhushan Shrirame
Roll No.: 23CS06002

Implementation of Lamport's Mutual Exclusion Algorithm Using Logical Clocks and Socket Communication

Introduction:

The project aims to implement Lamport's Mutual Exclusion Algorithm utilizing logical clocks for synchronization and socket communication for inter-device interaction. This report outlines the design, implementation, and functionality of the system.

Design Overview:

The system consists of three devices, each running two threads: one for event generation and mutual exclusion request transmission, and the other for receiving messages and updating logical clocks accordingly. The critical section is assumed to be access to a file on any of the three machines.

Implementation Details:

- Main Class (Lamports_Mutual_Exclusion):
    1. Accepts user input for process ID, priority, and peer information.
    2. Initializes the EventThread with the provided parameters.
    3. Listens for incoming messages via sockets and triggers interrupt handling accordingly.

- EventThread Class:
    1. Manages the event generation, mutual exclusion request, and critical section execution.
    2. Utilizes logical clocks for synchronization.

3. Maintains a queue of processes requesting access to the critical section.
4. Sends and receives messages to/from peers using sockets.
5. Updates the queue and logical clocks based on incoming messages.

Functionality:

- Event Generation:
  1. Simulates local events and send events with appropriate mutual exclusion requests.
  2. Introduces randomness in event generation to emulate real-world scenarios.
- Mutual Exclusion:
  1. Processes maintain a queue to track requests for accessing the critical section. Each request includes the process ID and priority.
  2. When a process wants to enter the critical section, it adds its request to the queue and sends request messages to other processes.
  3. Upon receiving a request message, a process adds the requesting process to its queue and sends a reply back. The process waits until it receives replies from all other processes before entering the critical section.
  4. The while loop ensures that the current process waits until it becomes the head of the queue and receives replies from at least two other processes, indicating their consent for mutual exclusion.
  5. After completing the critical section, the process removes itself from the queue and broadcasts a completion message to other processes.
  6. Processes wait in the queue based on priority, ensuring mutual exclusion. Higher-priority processes are given precedence.
  7. Processes handle interrupt messages to update the queue and reply counts accordingly.
  8. Communication between processes is facilitated through socket communication, exchanging messages to coordinate access to the critical section.
- Synchronization:
  1. Achieves synchronization between sender and receiver threads by updating logical clocks upon message reception and transmission.
  2. Ensures consistency in logical clocks across all devices.

Challenges and Solutions:

- Synchronization:
    1. Ensuring accurate synchronization between sender and receiver threads was a key challenge. This was addressed by updating logical clocks upon message handling in the receiver thread.
- Queue Management:
    1. Managing the queue of processes requesting access to the critical section required careful handling to ensure fairness and correctness. This was achieved by maintaining a sorted queue based on process IDs and priorities.

Conclusion:

The implemented system successfully demonstrates Lamport's Mutual Exclusion Algorithm using logical clocks and socket communication. It effectively manages mutual exclusion requests, ensures synchronization between devices, and provides access to the critical section in a controlled and fair manner. Further optimizations and enhancements can be explored to improve the system's efficiency and scalability.

Note: Sample Output is uploaded in github.