Chandy Lamport Algorithm

Need and Importance

Need

- Distributed debugging requires observing the entire system state at a specific moment.
- Lack of a global clock makes it hard to track event order.
- Messages traveling between processes can cause hidden errors.
- Stopping the system for debugging is impractical in real-time environments.

Importance

- Captures a consistent global snapshot without interrupting execution.
- Helps identify deadlocks, race conditions, message loss, and inconsistent states.
- Provides an accurate view of process and channel states.
- Improves reliability and simplifies debugging in complex distributed systems.

Algorithm

Marker Sending Rule for process *i*

1. Process *i* records its state.
2. For each outgoing channel C on which a marker has not been sent, *i* sends a marker along C before *i* sends further messages along C.

Marker Receiving Rule for process *j*

On receiving a marker along channel C:

if *j* has not recorded its state **then**

Record the state of C as the empty set

Follow the "Marker Sending Rule"

else

Record the state of C as the set of messages received along C after *j*'s state was recorded and before *j* received the marker along C

Real-Time Applications

- Monitoring system behavior during execution
- Detecting deadlocks and synchronization problems
- Analyzing message flow between processes
- Creating checkpoints for failure analysis
- Debugging large-scale systems like cloud platforms and microservices

Requirements to Implement

- Multiple processes communicating via message passing
- Reliable FIFO channels to maintain message order
- Ability to record local process states
- Mechanism to send and receive marker messages
- Storage to save snapshot data for later debugging
- Tools to reconstruct and analyze the global state

Thank You