Visvesvaraya National Institute of Technology, Nagpur

Department of Computer Science & Engineering

Distributed System

Assignment 2

1.)Program to implement Lamport's Distributed Mutual Exclusion Algorithm

Lamport's Distributed Mutual Exclusion Algorithm is a contention-based algorithm for mutual exclusion on a distributed system.

Algorithm

Nodal properties

1. Every process maintains a queue of pending requests for entering critical section order. The queues are ordered by virtual timestamps derived from Lamport timestamps.

Algorithm

Requesting process

1. Enters its request in its own queue (ordered by time stamps)
2. Sends a request to every node.
3. Wait for replies from all other nodes.
4. If own request is at the head of the queue and all replies have been received, enter critical section.
5. Upon exiting the critical section, send a release message to every process.

Other processes

1. After receiving a request, send a reply and enter the request in the request queue (ordered by time stamps)
2. After receiving release message, remove the corresponding request from the request queue.
3. If own request is at the head of the queue and all replies have been received, enter critical section.

2.) Program to implement Lamport's Logical Clock Algorithm

Lamport logical clock

→: happen before

1) a → b two events occur at the same process

2) a → b for sending event and receiving event of a

3) if a →b and b →c then a →c

→ is a transitive relation.

→ may be referred to as *causally affect*.

Concurrent events

a || b if (not a → b) and (not b → a)

For any two events: either a → b or b → a or a || b.

General assumption: no two events occur at the same time.

Reason: no referencing global time to prove two events occurred at the same time.

Logical clock Algorithm:

Each process I maintains an integer variable X.i.

[IR1] An event occurs at Pi: X.i = X.i + 1

[IR2] When Pi receives a message with time stamp (TS,j) from Pj:

X.i = max(X.i, TS) + 1

Each event on Pi has an associated time stamp (Tsi,I).

Tsi is the X.i right after the event (after applying [IR1] or [IR2].

Total ordering: Any two time stamps of different events can be totally ordered.

(Tsi,I) < (TSj,j) if (Tsi < TSj) or ((Tsi = TSj) and (I < j))

Logical time is a discrete virtual time.

Pi waits for a specific logical time (e.g. X.i = 5) is risky because X.i may jump over 5 (from X.i < 5 to X.i > 5.

3.) Program to implement Edge Chasing distributed deadlock detection algorithm

Edge-chasing is an algorithm for deadlock detection in distributed systems.
Whenever a process *A* is blocked for some resource, a probe message is sent to all processes *A* may depend on. The probe message contains the process id of *A* along with the path that the message has followed through the distributed system. If a blocked process receives the probe it will update the path information and forward the probe to all the processes it depends on. Non-blocked processes may discard the probe.
If eventually, the probe returns to process *A*, there is a circular waiting loop of blocked processes, and a deadlock is detected. Efficiently detecting such cycles in the “wait-for graph” of blocked processes is an important implementation problem.