Synchronization

COMP750 Distributed Systems

Textbook

Read chapter 5 of the textbook

What Time is It?

- Many algorithms, particularly distributed algorithms, are expected to do thing in time order. Knowing the exact time is necessary to do this.
- Ordering events on different computers requires a consistent time system.
- Computer clocks usually do not keep accurate time.
- One of the defining concepts of a distributed system is that it does not have a central clock.

What is the ***real*** time of day?

- When you measure time accurately, what is the basis for correctness?
  - Earth rotation
  - Solar rotation
  - Atomic cesium clock

- Coordinated Universal Time (UTC) is the world’s standard time (used to be Grenidge Mean Time)

Time Servers

- There are many time servers on the Internet.
- Primary time servers have an accurate clock.
- Secondary time servers synchronize with a primary time server.

Time Server Equipment

- Time servers can be purchased in the range of $2,200 to $3,200.

Network Time Protocol

- The Network Time Protocol (NTP) can be used to get the time from a time server.
- The old fashion method was to send a packet to the time server and it would send a packet with the time.
- To get an accurate time you must account for the network delay.

Lamport Time Algorithm

- The time algorithm developed by Leslie Lamport creates a non-linear virtual time.
- Time is represented as an integer. When significant events occur, the integer is incremented.
- On a single CPU system, events that:
  - Occur later have larger time numbers.
  - Have larger numbers occur later

Lamport Time Algorithm

- When any message is sent to another system, the time value is sent with it.
- When a message arrives from another computer, the local time number is incremented by one.
- If the time value in the message > local time, the local time is incremented to the time value + 1.

Happens Before Relationship

- An event ***x*** can be said to *Happen Before* an event ***y*** if there is a path from ***x*** to ***y*** in the time diagram.
- Not all events that actually happen before another event will have the Happen Before relationship.
- Two events are said to be concurrent if neither happens before the other.

Distributed Mutual Exclusion

- Distributed mutual exclusion coordinates software on different computers so that they agree upon assigning a resource or section of code to one particular task.

Requirements

- Mutual Exclusion
- Freedom from deadlock
- Eventual entry (freedom from starvation)
- *All processes must participate equally*. Only interested processes participate.

Assumptions

- N nodes randomly request access.
- Messages are not lost or corrupted.
- Message transmissions take a finite, variable, non-zero time.
- Messages arrive in order.
- Transmission time might not be transitive.
- Network is fully connected.

Mutual Exclusion Goals

- Minimize the number of messages sent.
- Grant permission in order of request.
- Fault tolerant
- Fair to all systems
- Scalable

Distributed Mutual Exclusion Algorithms

- Assertion Based
  - Lamport algorithm
  - Ricart-Agrawala algorithm
  - Maekawa algorithm

- Token Based
  - Raymond’s Tree-based algorithm
  - Simple 2 process algorithm

Lamport’s DME Algorithm

- Processes are granted mutual exclusion in the order in which they make the request. Each process maintains a request queue sorted in timestamp order.
- Assertion based algorithm.
- Uses the Lamport time numbers.

Lamport Algorithm (cont)

1. To request a resource, process *Pi* sends a timestamped request message to every other process and puts the request on its own request queue.
2. When process *Pk* receives a request message, it sends a timestamped reply and puts the request on its request queue.

Lamport Algorithm (cont)

3. Process *Pi* can access the resource when:

- Its own request is at the top of the request queue.
- It has received a reply from every other process.

4. To release a resource, *Pi* removes its request from its queue and sends a release message to all other processes.

Lamport Algorithm (cont)

5. When process *Pk* receives a release message, it removes *Pi*’s request from its queue.