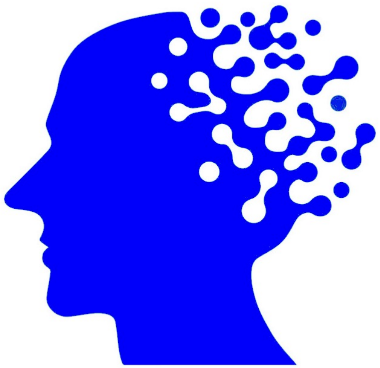
Spring 2023



Distributed Computing

- Non-Token-Based MutEx algorithms

Dr. Rajendra Prasath

Indian Institute of Information Technology Sri City, Chittoor

27th March 2023 (http://rajendra.2power3.com)

> Distributed Computing?

- ➤ How will you design a Distributed Algorithm?

- ➤ Learn to Solve using Distributed Algorithms

Overview

Recap: Distributed Systems

A Distributed System:

- A collection of independent systems that appears to its users as a single coherent system
- A system in which hardware and software components of networked computers communicate and coordinate their activity only by passing messages
- A computing platform built with many computers that:
  - Operate concurrently
  - Are physically distributed (have their own failure modes)
  - Are linked by a network
  - Have independent clocks

Recap: Characteristics

- Concurrent execution of processes:
  - Non-determinism, Race Conditions, Synchronization, Deadlocks, and so on
- No global clock
  - Coordination is done by message exchange
  - No Single Global notion of the correct time
- No global state
  - No Process has a knowledge of the current global state of the system
- Units may fail independently
  - Network Faults may isolate computers that are still running
  - System Failures may not be immediately known

What did you learn so far?

- Goals / Challenges in Message Passing systems
- Distributed Sorting / Space-Time diagram
- Partial Ordering / Total Ordering
- Concurrent Events / Causal Ordering
- Logical Clocks vs Physical Clocks
- Global Snapshot Detection
- Termination Detection Algorithm
- Leader Election in Rings
- Topology Abstraction and Overlays
- Message Ordering and Group Communication

- Mutual Exclusion Algorithm

[Now] → → →

Recap

> About this Lecture

What do we learn today?

- ➤ Mutual Exclusion Algorithms
  - ➤ Centralized Algorithm
  - ➤ Token-Based / Permission-Based Algorithms
  - ➤ Quorum-Based Algorithm
  - ➤ Tree-Based Algorithm

Let us explore these topics ➔ ➔ ➔

Today's Focus

Distributed Mutual Exclusion – Token / Non-Token-Based MutEx Algorithms

Recap: The need for MutEx?

- Mutual Exclusion
  - Operating systems: Semaphores
    - In a single machine, you could use semaphores to implement mutual exclusion
    - How to implement semaphores?
      - Inhibit interrupts
      - Use clever instructions (e.g. test-and-set)
    - On a multiprocessor shared memory machine, only the latter works

Characteristics

- Processes communicate only through messages – no shared memory or no global clocks
- Processes must expect unpredictable message but finite delays
- Processes coordinate access to shared resources that should only be used in a mutually exclusive manner.

Recap: Distributed MutEx

- **No Deadlocks** – no set of sites should be permanently blocked, waiting for messages from other sites in that set
- **No starvation** – no site should have to wait indefinitely to enter its critical section, while other sites are executing the CS more than once
- **Fairness** - requests honored in the order they are made. This means processes have to be able to agree on the order of events. (Fairness prevents starvation.)
- **Fault Tolerance** – the algorithm is able to survive a failure at one or more sites

Quorum Based algorithms

Why Quorum based algorithm?

- Lamports and Ricard-Agrawala' algorithm requires permission from all processes to enter into the critical section.

Modifications:

- Is it necessary to obtain permission from all processes before entering into the CS?
- How to reduce the message exchanges and increase the performance of MutEx algorithm?

Quorum Based algorithms

What is a Quorum?

- There are n requesting processes in a distributed system and any process may request for CS.
- Can we form such a subset of processes who request for Critical Section? YES !!
  - Such a set is said to be a Request Set or Quorum
  - In fact, we will have a separate Request set for each process P_i

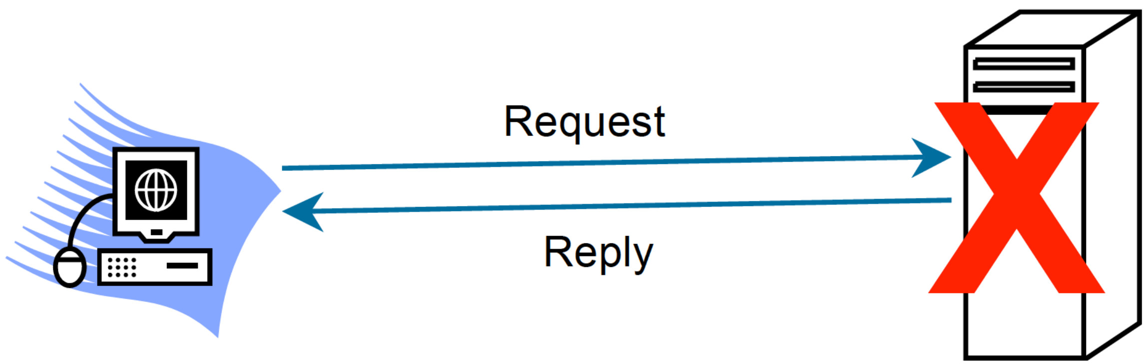
Quorum - Definition

- A quorum system is a collection of subsets of processes, called quorums, such that each pair of quorums have a non-empty intersection

- How do we formally define a quorum of processes in a distributed system?
- Let us look at some examples

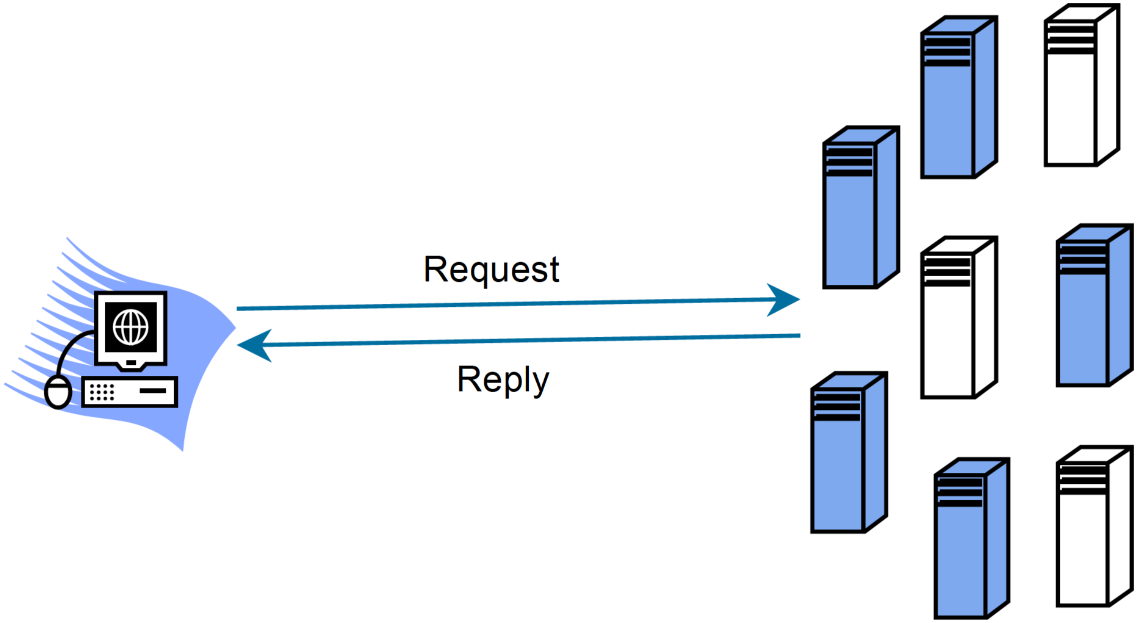
Quorum – Why?

- Process may not respond or may go down (any kind of failure)
- The requesting process can not get REPLY from all remaining processes
- It would infinitely wait for CS !!



Quorum – Why?

- Can the requesting process get permission from a quorum of processes to enter into CS?



Quorum - Definition

More Formally,

- Given a set of processes

$$P = \{P_1, P_2, \ldots, P_n\}$$

- A quorum system $Q \subseteq 2^P$ is a set of subsets of P such that

  for all Q_1, Q_2 in Q: $Q_1 \cap Q_2 \neq empty$

- Each Q_i in Q is called a quorum

Maekawa's Algorithm

- Permission obtained from only a subset of other processes, called the Request Set (or Quorum)

- Separate Request Set R_i, for each process i

Maekawa's Algorithm

Requirements

- **For all** i, j: $R_i \cap R_j \neq \Phi$
- **For all** i: $i \in R_i$
- **For all** i: $| R_i | = K$, **for some** K
- **Any node** i **is contained in exactly** D **Request Sets, for some Request set** D

- $K = D = sqrt(N)$ **for Maekawa's algorithm**

Maekawa's Algorithm - Steps

To Request Critical Section:

- ➔ P_i sends REQUEST message to all process in R_i

On receiving a REQUEST message:

- ➔ Send a REPLY message if no REPLY message has been sent since the last RELEASE message is received.
- ➔ Update status to indicate that a REPLY has been sent.
- ➔ Otherwise, queue up the REQUEST

To enter critical section:

- ➔ P_i enters critical section after receiving REPLY from all nodes in R_i

Maekawa’s Algorithm – Steps (contd)

To release critical section:

- Send RELEASE message to all nodes in R_i
- On receiving a RELEASE message, send REPLY to next node in queue and delete the node from the queue.
- If queue is empty, update status to indicate no REPLY message has been sent

Computation Complexity

- Message Complexity: 3 * sqrt (N)
- Synchronization delay
  - 2*(max message transmission time)
- Major problem: DEADLOCK possible
- Need three more types of messages (FAILED, INQUIRE, YIELD) to handle deadlock.
  - Message complexity can be 5* sqrt(N N)
- Important Issue:
  - How to build the request sets?

Raymond's Algorithm

- Forms a directed tree (logical) with the token token-holder as root
- Each node has variable "Holder" that points to its parent on the path to the root.
  - Root's Holder variable points to itself
- Each node P_i has a FIFO request queue Q_i

Raymond's Algorithm

- To request critical section:
  - Send REQUEST to parent on the tree, provided i does not hold the token currently and Q_i is empty. Then place is request in Q_i

- When a non-root node j receives a request from k
  - place request in Q_j
  - send REQUEST to parent if no previous REQUEST sent

Raymond's Algorithm (contd)

When the root receives a REQUEST:

- send the token to the requesting node
- set Holder variable to point to that node

When a node receives the token:

- delete first entry from the queue
- send token to that node
- set Holder variable to point to that node
- if queue is non non-empty, send a REQUEST message to the parent (node pointed at by Holder variable)

Raymond's Algorithm (contd)

- To execute critical section:
  - enter if token is received and own entry is at the top of the queue; delete the entry from the queue
- To release critical section:
  - if queue is non non-empty, delete first entry from the queue, send token to that node and make Holder variable point to that node
  - If queue is still non non-empty, send a REQUEST message to the parent (node pointed at by Holder variable)

Features of Raymond's Algo

- Average message complexity:
  - O(log n)
- Sync. Delay
  - (T log n)/2, where T = max. message delay

Summary

- Recap: Distributed Mutual Exclusion Algorithms
  - Mutual Exclusion Problem
    - Basics of MutEx algorithms
    - Types of MutEx algorithms
      - Token-based Algorithms
        - Raymond's Tree based algorithm
      - Non-Token based Algorithms
        - Quorum based algorithm
    - Performance Metrics

Many more to come up … ! Stay tuned in !!

Penalties

- Every Student is expected to strictly follow a fair Academic Code of Conduct to avoid penalties

- Penalties is heavy for those who involve in:
  - Copy and Pasting the code
  - Plagiarism (copied from your neighbor or friend – in this case, both will get "0" marks for that specific take home assignments)
  - If the candidate is unable to explain his own solution, it would be considered as a "copied case"!!
  - Any other unfair means of completing the assignments

Help among Yourselves?

- **Perspective Students** (having CGPA above 8.5 and above)
- **Promising Students** (having CGPA above 6.5 and less than 8.5)

- **Needy Students** (having CGPA less than 6.5)
  - Can the above group help these students? (Your work will also be rewarded)

- You may grow a culture of **collaborative learning** by helping the needy students

How to reach me?

- Please leave me an email:
  rajendra [DOT] prasath [AT] iiits [DOT] in

- Visit my homepage @
  - https://www.iiits.ac.in/people/regular-faculty/dr-rajendra-prasath/

(OR)

  - http://rajendra.2power3.com

Assistance

- You may post your questions to me at any time
- You may meet me in person on available time or with an appointment
- You may ask for one-to-one meeting

Best Approach

- You may leave me an email any time
  (email is the best way to reach me faster)

Questions

It's Your Time

Contact Information

Dr. Rajendra Prasath
IIIT Sri City, Chittoor