Spring 2023

Distributed Computing

- Leader Election Algorithm

Dr. Rajendra Prasath

Indian Institute of Information Technology Sri City, Chittoor

14th 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**
➔ **Causal Precedence Relation**
   ➔ **Happens Before**

➔ **Concurrent Events**
   ➔ **How to define Concurrent Events**
   ➔ **Logical vs Physical Concurrency**

➔ **Causal Ordering**
➔ **Logical Clocks vs Physical Clocks**
➔ **Global Snapshot Detection**
➔ **Termination Detection Algorithm**

➔ **[Now] Leader Election in Rings …**

Recap

> About this Lecture

What do we learn today?

➤ This covers a model of distributed computations that every algorithm designer needs to know

   ➤ **Termination Detection algorithm**
      ➤ **Using Distributed Snapshots**

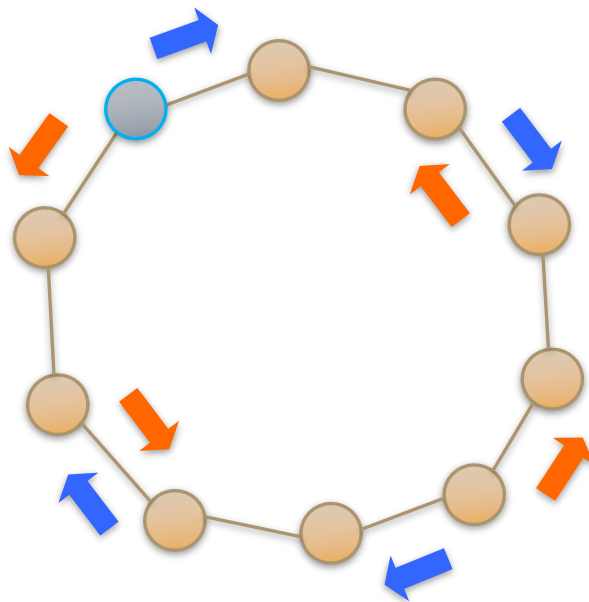   ➤ **Leader Election in Rings**

Let us explore these topics ➔ ➔ ➔

Leader Election in Distributed Systems

Ring Networks

➔ **In an oriented ring, processes have a consistent notion of left and right**

Clockwise (right)

Anti-clockwise (left)

➔ **For example, if messages are forwarded on right channel, they will cycle clockwise around the ring**

Why Study Rings?

- Simple starting point, easy to analyze

- Abstraction of a token ring

- Lower bounds and impossibility results for ring topology also apply to arbitrary topologies

Leader Election - Definition

- Each processor has a set of **elected** (won) and **not-elected** (lost) states.

- Once an **elected** state is entered, processor is always in an elected state (and similarly for not-elected): i.e., **irreversible** decision

- In every **admissible** execution:
  - every processor eventually enters **either** an elected or a not-elected state
  - exactly **one** processor (the **leader**) enters an elected state

Uses of Leader Election

- A leader can be used to coordinate activities of the system:
  - find a spanning tree using the leader as the root
  - reconstruct a lost token in a token-ring network

- We will study leader election in rings.

Anonymous Rings

- How to model situation when processes do not have unique identifiers?

- First attempt: Does each process require to be in the same state machine?

- Subtle point: Does the algorithm rely on knowing the ring size (number of processes)?

Uniform (Anonymous) Algorithms

- A **uniform** algorithm does not use the ring size (same algorithm for each size ring)
  - Formally, every processor in every size ring is modeled with the same state machine

- A **non-uniform** algorithm uses the ring size (different algorithm for each size ring)
  - Formally, for each value of n, every processor in a ring of size n is modeled with the same state machine A_n.

- Note the lack of unique ids.

Leader Election in Anonymous Rings

- **Theorem:** There is *no* leader election algorithm for anonymous rings, even if
  - algorithm knows the ring size (non-uniform)
  - synchronous model

- **Proof Sketch:**
  - Every processor begins in same state with same outgoing messages (since anonymous)
  - Every processor receives same messages, does same state transition, and sends same messages in round 1
  - Do the same for rounds 2, 3, …
  - Eventually some processor is supposed to enter an elected state.

Leader Election in Anonymous Rings

- Proof sketch shows that either ==safety== (never elect more than one leader) or ==liveness== (eventually elect at least one leader) is violated

- Since the theorem was proved for non-uniform and synchronous rings, the same result holds for weaker (less well-behaved) models:
  - Uniform
  - Asynchronous

Rings with Identifiers

- Assume each processor has a unique ID.

- Don't confuse indices and IDs:

  - **indices** are 0 to $n - 1$; used only for analysis, not available to the processors

  - **IDs** are arbitrary nonnegative integers; are available to the processors through local variable *ID*

Specifying a Ring

- Start with the smallest ID and list IDs in clockwise order.
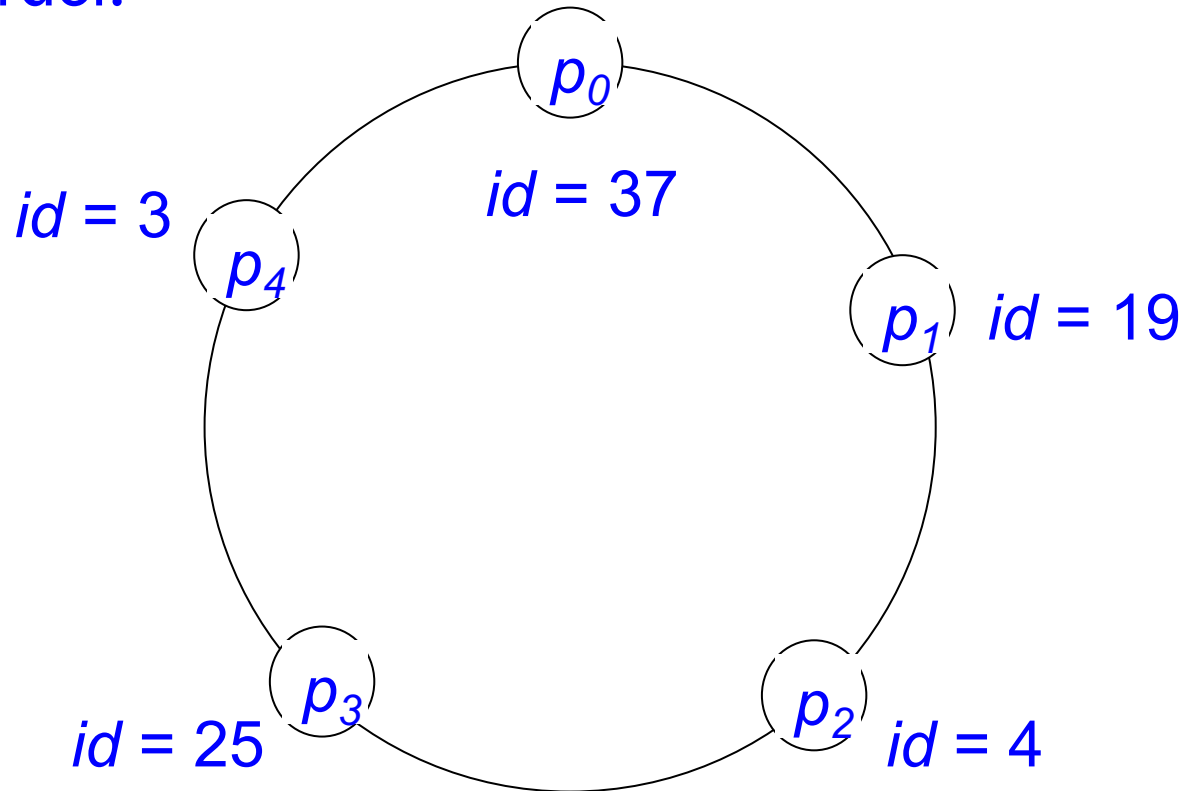
- Example: 3, 37, 19, 4, 25

Uniform (Non-anonymous) Algorithms

- **Uniform** algorithm: there is one state machine for every id, no matter what size ring

- **Non-uniform** algorithm: there is one state machine for every id and every different ring size

- These definitions are tailored for leader election in a ring.

Overview of LE in Rings with IDs

- There exist algorithms when nodes have unique IDs.

- We will evaluate them according to their *message complexity*.

- asynchronous ring:
  - $\Theta(n \log n)$ messages

- synchronous ring:
  - $\Theta(n)$ messages under certain conditions
  - otherwise $\Theta(n \log n)$ messages

- All bounds are asymptotically tight.

The LCR algorithm

- Lelann-Chang-Robert (LCR) Algorithm, 1979

- The network graph is a directed ring (uni-directed or bi-directed) consisting of n nodes (n may be unknown to the processes … Does this condition really require for rings?)

- Processes run the same deterministic algorithm
- The only piece of information supplied to the processes is a unique identifier (ID).
- IDs may be used

- In comparisons only (comparison-based algorithms)
- In comparisons and other calculations (non-comparison-based)

LCR algorithm - Description

- Each process sends its ID around the ring.

- When a process receives a ID, it compares this one to its own

- If the incoming ID is greater, then it passes this ID to the next process.

- If the incoming ID is smaller, then it discards it.

- If it is equal, then the process declares itself the leader.

# LCR Algorithm for Leader Election

Alternative Algorithm:

- send value of own id to the left
- when receive an ID j (from the right):
- if j > id then
    - forward j to the left (this processor has lost)
  - if j < id then
    - do nothing
  - if j = id then
    - elect self as leader
      (this processor has won)

Analysis of O(n²) Algorithm

- **Correctness:** Elects processor with largest id.
  - message containing largest id passes through every processor
- **Time:** *O(n)*

- **Message complexity:** Depends how the ids are arranged.
  - largest id travels all around the ring (n messages)
  - 2nd largest id travels until reaching largest
  - 3rd largest id travels until reaching largest or second largest
  - And so on

Analysis of O(n²) Algorithm

- Worst way to arrange the ids is in decreasing order:
  - 2nd largest causes *n - 1* messages
  - 3rd largest causes *n - 2* messages and so on
- Total number of messages is *n + (n-1) + (n-2) + … + 1* = Θ*(n²)*.
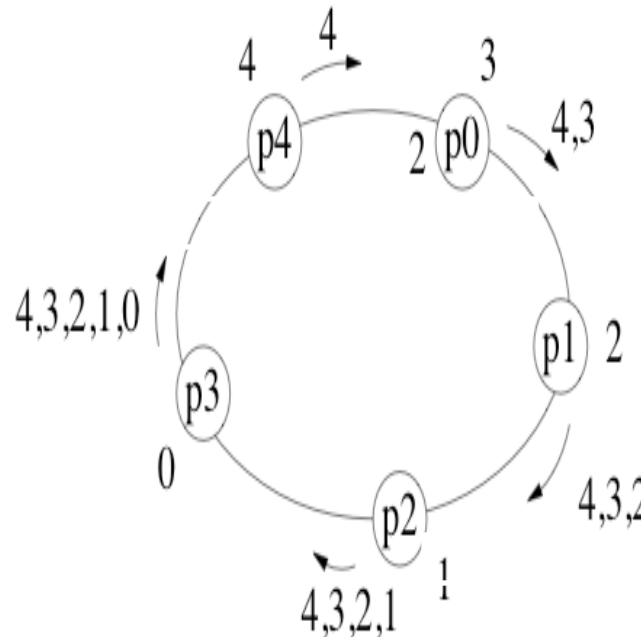
Can We Use Fewer Messages?

- The *O(n²)* algorithm is simple and works in both <mark>synchronous and asynchronous</mark> model

- But can we solve the problem with fewer messages?

- Idea:
  - Try to have messages containing smaller ids travel smaller distance in the ring

O(n log n) Leader Election

- Each process tries to probe successively larger neighborhoods in both directions

  - size of neighborhood *doubles* in each phase

- If probe reaches a node with a larger id, the probe stops

- If probe reaches end of its neighborhood, then a reply is sent back to initiator

- If initiator gets back replies from both directions, then go to next phase

- If process receives a probe with its own id, it elects itself

O(n log n) Leader Election
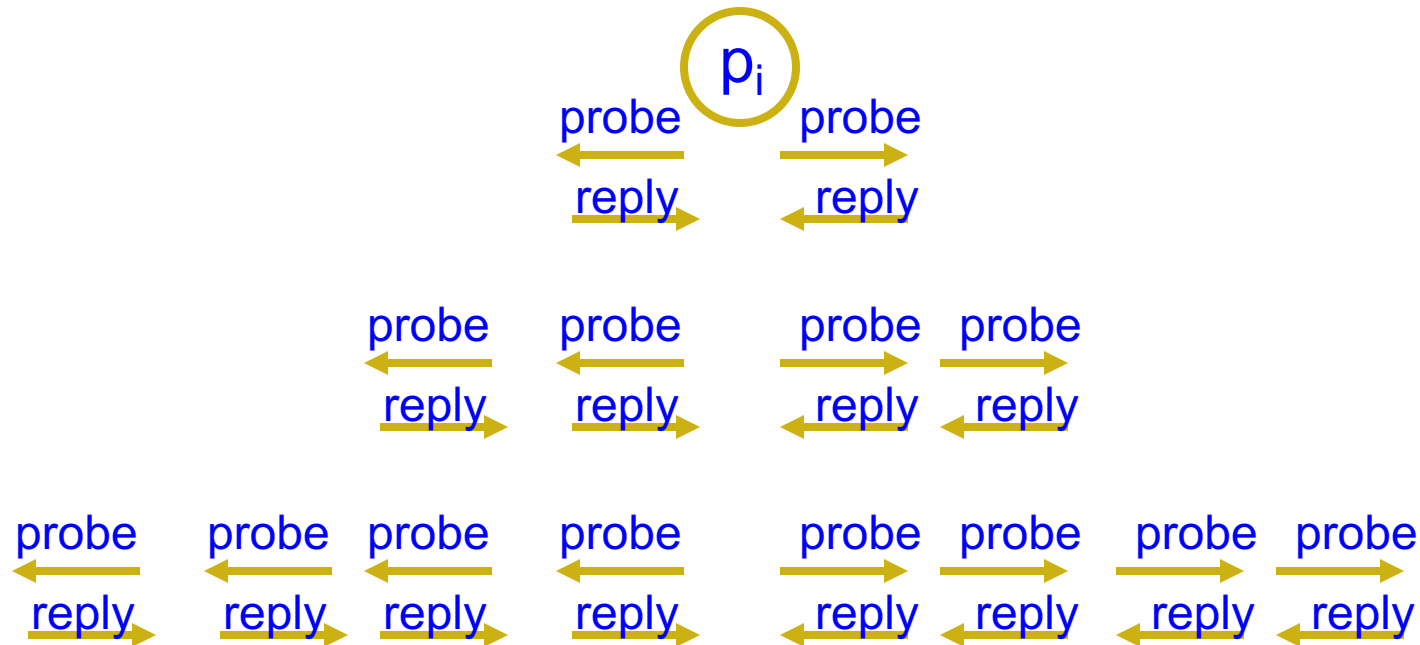
Analysis of O(nlogn) Algorithm

- **Correctness:** Similar to $O(n^2)$ algorithm.

- **Message Complexity:**

  - Each message belongs to a particular phase and is initiated by a particular proc.

  - Probe distance in phase k is 2^k

  - Number of messages initiated by a proc. in phase k is at most $4*2^k$ (probes and replies in both directions)

Analysis of O(n log n) Algo

- How many processes initiate probes in phase k?

  - For $k = 0$, every process does

  - For $k > 0$, every process that is a "winner" in phase $k - 1$ does

  - "winner" means has largest id in its 2^{k-1} neighborhood

Analysis of O(n log n) Algo

- Maximum number of phase *k - 1* winners occurs when they are packed as densely as possible:

2^{k-1} processors
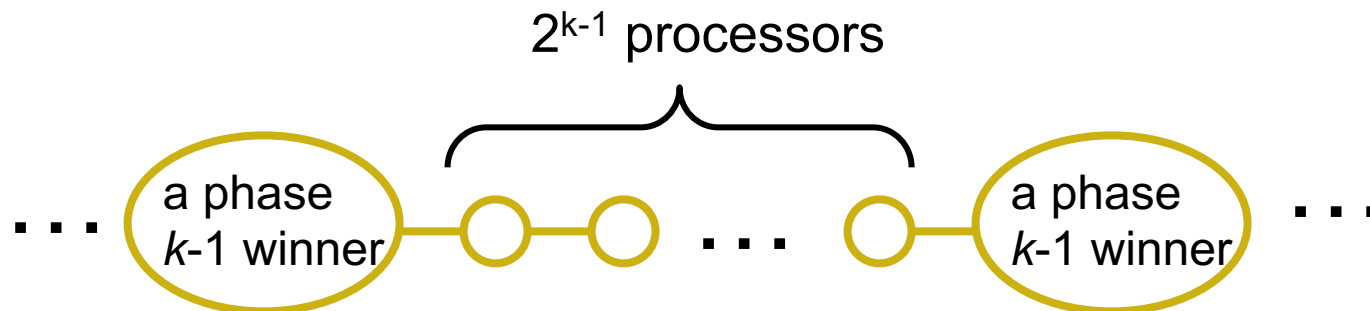
- The total number of phase *k - 1* winners is at most

$$n/(2^{k-1} + 1)$$

Analysis of O(n log n) Algo

- How many phases are there?

- At each phase the number of (phase) winners is cut approx. in half
  - from $n/(2^{k-1} + 1)$ to $n/(2^k + 1)$

- So after approx. $\log_2 n$ phases, only one winner is left.
  - more precisely, max phase is $\lceil \log(n-1) \rceil + 1$

Analysis of O(n log n) Algo

- Total number of messages is sum, over all phases, of number of winners at that phase times number of messages originated by that winner:

$$\leq 4n + n + \sum_{k=1}^{\lceil \log(n-1) \rceil + 1} 4 \bullet 2^k \bullet n / (2^{k-1} + 1)$$

phase 0 msgs

termination msgs

msgs for phases 1 to $\lceil \log(n-1) \rceil + 1$

$$< 8n(\log n + 2) + 5n$$

$$= O(n \log n)$$

Can We Do Better?

- The *O(n* log *n)* algorithm is more complicated than the *O(n²)* algorithm but uses fewer messages in the worst case.

- Works in both synchronous and asynchronous case.

- Can we reduce the number of messages even more?

- Not in the asynchronous model … !!

Summary

➔ **Termination Detection**
- ➔ Definition
- ➔ TD using Global Snapshots

➔ **Leader Election in Rings**
- ➔ Leader Election Problem
- ➔ Formulation of the problem
- ➔ LCR algorithm
- ➔ O(nlogn) algorithm using probes
- ➔ Complexity Analysis

➔ 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

How may I assist you?

Contact Information

Dr. Rajendra Prasath
IIIT Sri City, Chittoor