Leader Election Algorithms

- Many distributed algorithms need one process to act as coordinator
  - Doesn’t matter which process does the job, just need to pick one
  - Example: pick a master in Berkeley clock synchronization algorithm
- Election algorithms: technique to pick a unique coordinator (aka leader election)

Bully Algorithm

- Each process has a unique numerical ID
- Processes know the ID and address of every other process
- Communication is assumed reliable
- **Key Idea:** select process with highest ID
- Process initiates election if it just recovered from failure or if coordinator failed
- 3 message types: Election, OK, Coordinator
- Several processes can initiate an election simultaneously
  - Need consistent result
- \( O(n^2) \) messages required with \( n \) processes

Bully Algorithm Details

- Any process \( P \) can initiate an election
- \( P \) sends Election messages to all process with higher IDs and awaits OK messages
- If no OK messages, \( P \) becomes coordinator and sends Coordinator messages to all process with lower IDs
- If it receives an OK, it drops out and waits for an Coordinator message
- If a process receives an Election message, it returns an OK and starts an election
- If a process receives a Coordinator message, it treats sender an coordinator

Bully Algorithm Example

- Process 4 holds an election.
- Process 5 and 6 respond, telling 4 to stop.
- 5 and 6 each hold an election.
- Process 6 tells 5 to stop.
- Process 6 wins and tells everyone.

Ring-based Election

- Processes have unique IDs and arranged in a logical ring
  - Select process with highest ID
  - Each process knows its neighbors
  - Begin election if just recovered or coordinator has failed
  - Send Election message to closest downstream node that is alive
    - Sequentially poll each successor until a live node is found
  - Each process tags its ID on the message
  - When Election message comes back, initiator picks node with highest ID and sends a coordinator message
    - Coordinator message is removed when it has circulated once.
  - Multiple elections can be in progress
    - Wastes network bandwidth but does no harm
A Ring Algorithm

Comparison

• Assume n processes and one election in progress
  • Bully algorithm
    – Worst case: initiator is node with lowest ID
      • Triggers n-2 elections at higher ranked nodes: O(n^2) messages
    – Best case: immediate election
      • n-2 messages
  • Ring
    – 2(n-1) messages always

Election in Wireless Environments

• Election algorithm in a wireless network, with node a as the source. (a) Initial network, (b)-(e) The build-tree phase. (f) Reporting of best node to source.

Distributed Synchronization

• Processes in a distributed system may need to simultaneously access the same resources
  – Processes may be running on different machines
• Need to grant mutual exclusive access to shared resources by processes
• Solution: lock mechanism
  – Can be centralized or distributed

Centralized Mutual Exclusion

• Assume processes are numbered
• One process is elected coordinator
• Every process needs to check with coordinator before entering the critical section
• To obtain exclusive access: send request, await reply
• To release: send release message
• Coordinator:
  – Receive request: if resource is available and queue empty, send grant; if not, queue request
  – Receive release: remove next request from queue and send grant
Centralized Mutual Exclusion

- Process 1 asks the coordinator for permission to access a shared resource. Permission is granted.
- Process 2 then asks permission to access the same resource. The coordinator does not reply.
- When process 1 releases the resource, it tells the coordinator, which then replies to 2.

Properties

- Simulate centralized locking with blocking calls
- Fair: requests are granted in the order they were received
- No starvation
- Simple: three messages per use of a resource (request, grant, release)
- Shortcomings:
  - Single point of failure
  - How do you detect a dead coordinator?
  - No response from coordinator in either case
  - Performance bottleneck in large distributed systems

Decentralized Algorithm

- Use voting
- Each resource is replicated n times. Each replica has its own coordinator.
- To acquire lock, need majority vote from m > n/2 coordinators.
  - Nonblocking: coordinators return OK or "no"
- Coordinator crash => forgets previous votes
  - Probability that k coordinators reset p(k) = C(m,k)p^k(1-p)^m-k
  - p is the probability that a coordinator resets during a time interval
  - At least 2m-n coordinators need to reset to violate correctness
  - Probability of such a violation is \[ \sum_{k=2m-n}^{m} p(k) \]

Decentralized Algorithm

- If request is denied, process will back off for a randomly-chosen time, and try later
- Drawbacks
  - Low resource utilization when many nodes want to access the same resource
  - Starvation