CSE 486/586 Distributed Systems
Leader Election

Steve Ko
Computer Sciences and Engineering
University at Buffalo

Recap: Maekawa’s Algorithm
• Deadlock?

Why Election?
• Example 1: sequencer for TO multicast
• Example 2: leader for mutual exclusion
• Example 3: group of NTP servers: who is the root server?

What is Election?
• In a group of processes, elect a leader to undertake special tasks.
• What happens when a leader fails (crashes)
  – Some process detects this (how?)
  – Then what?
• Focus of this lecture: election algorithms
  – 1. Elect one leader only among the non-faulty processes
  – 2. All non-faulty processes agree on who is the leader
• We’ll look at 3 algorithms
  – 2 for asynchronous systems
  – 1 for synchronous systems

Assumptions
• Any process can call for an election.
• A process can call for at most one election at a time.
• Multiple processes can call an election simultaneously:
  – All of them together must yield a single leader only
  – The result of an election should not depend on which process calls for it.
• Messages are eventually delivered.

Problem Specification
• At the end of the election protocol, the non-faulty process with the best (highest) election attribute value is elected.
  – Attribute examples: CPU speed, load, disk space, ID
  – Must be unique
• Each process has a variable elected.
• A run (execution) of the election algorithm must always guarantee at the end:
  – Safety: ∀ non-faulty p: (p’s elected = (q: a particular non-faulty process with the best attribute value) or ⊥)
  – Liveness: ∀ election: (election terminates) & ∀ p: non-faulty process, p’s elected is eventually not ⊥.
Algorithm 1: Ring Election
[Chang & Roberts’79]
- N Processes are organized in a logical ring
  - \( p_i \) has a communication channel to \( p_{(i+1) \mod N} \)
  - All messages are sent clockwise around the ring.
- To start election
  - Send \( \text{election} \) message with my ID
- When receiving message \( (\text{election}, \text{id}) \)
  - If \( \text{id} > \text{my ID} \): forward message
  - If \( \text{id} < \text{my ID} \): send \( (\text{election}, \text{my ID}) \)
  - Skip if already participating
  - Set state to \( \text{participating} \)
- If \( \text{id} = \text{my ID} \): I am elected (why?) send \( \text{elected} \) message
  - \( \text{elected} \) message forwarded until it reaches leader

Ring-Based Election: Example
- The worst-case scenario occurs when?
  - the counter-clockwise neighbor (@ the initiator) has the highest attr.
- In the example:
  - The election was started by process 17.
  - The highest process identifier encountered so far is 24
  - (final leader will be 33)

Ring-Based Election: Analysis
- In a ring of \( N \) processes, in the worst case:
  - \( N-1 \) \( \text{election} \) messages to reach the new coordinator
  - Another \( N \) \( \text{election} \) messages before coordinator decides it’s elected
  - Another \( N \) \( \text{elected} \) messages to announce winner
- Total Message Complexity = \( 3N-1 \)
- Turnaround time = \( 3N-1 \)

Correctness?
- Safety: highest process elected
- Liveness: complete after \( 3N-1 \) messages
  - What if there are failures during the election run?

Example: Ring Election
- PA2 due this Friday
- Midterm: 3/6 (Wednesday) in class
  - 45 minutes
  - Everything up to today
  - 1-page cheat sheet is allowed.
- Best way to prepare
  - Read the textbook & go over the slides
  - Go over the problems in the textbook
  - Will add more problems for the lectures this week & next
- PA3 will be out this weekend
- Anonymous feedback form still available.
- Please come to me!
Algorithm 2: Modified Ring Election

- **election** message tracks all IDs of nodes that forwarded it, not just the highest
  - Each node appends its ID to the list
- Once message goes all the way around a circle, new **coordinator** message is sent out
  - Coordinator chosen by highest ID in **election** message
  - Each node appends its own ID to **coordinator** message
- When **coordinator** message returns to initiator
  - Election a success if coordinator among ID list
  - Otherwise, start election anew

Example: Ring Election

1. **P2** initiates election
2. **P2** receives "election"; **P4** dies
3. **P2** selects **4** and announces the result
4. **P2** receives "Coordinator", but **P4** is not included
5. **P2** re-initiates election
6. **P3** is finally elected

Modified Ring Election

- How many messages?
  - \(2N\)
- Is this better than original ring protocol?
  - Messages are larger
- Reconfiguration of ring upon failures
  - Can be done if all processes "know" about all other processes in the system
- What if initiator fails?
  - Successor notices a message that went all the way around (how?)
  - Starts new election
- What if two people initiate at once
  - Discard initiators with lower IDs

What about that Impossibility?

- Can we have a **totally correct** election algorithm in a fully asynchronous system (no bounds)?
  - No! Election can solve consensus
- Where might you run into problems with the modified ring algorithm?
  - Detect leader failures
  - Ring reorganization

Algorithm 3: Bully Algorithm

- Assumptions:
  - Synchronous system
  - attr=id
  - Each process knows all the other processes in the system (and thus their id’s)

Algorithm 3: Bully Algorithm

- 3 message types
  - **election** – starts an election
  - **answer** – acknowledges a message
  - **coordinator** – declares a winner
- Start an election
  - Send election messages only to processes with higher IDs than self
  - If no one replies after timeout: declare self winner
  - If someone replies, wait for **coordinator** message
    » Restart election after timeout
- When receiving **election** message
  - Send **answer**
  - Start an election yourself
    » If not already running
Example: Bully Election

1. P2 initiates election
2. P2 receives replies
   - P1
   - P2
   - P3
   - P4
   - P0
   - P5
3. P3 & P4 initiate election
   - P1
   - P2
   - P3
   - P4
   - P0
   - P5
4. P3 receives reply
   - OK
   - Election
   - Election
   - Election
   - Election
   - Election
5. P4 receives no reply
6. P4 announces itself

Analysis of The Bully Algorithm

• Best case scenario?
  • The process with the second highest id notices the failure of the coordinator and elects itself.
    – N-2 coordinator messages are sent.
    – Turnaround time is one message transmission time.

Turnaround time

• All messages arrive within T units of time (synchronous)
• Turnaround time:
  – election message from lowest process (T)
  – Timeout at 2nd highest process (X)
  – coordinator message from 2nd highest process (T)
• How long should the timeout be?
  – X = 2T + T_{process}
  – Total turnaround time: 4T + 3T_{process}

Summary

• Coordination in distributed systems sometimes requires a leader process
• Leader process might fail
• Need to (re-) elect leader process
• Three Algorithms
  – Ring algorithm
  – Modified Ring algorithm
  – Bully Algorithm
Acknowledgements

- These slides contain material developed and copyrighted by Indranil Gupta (UIUC).