African University of Science and Technology

Election Algorithms and Distributed Processing

Prof. Ousmane THIARE

[www.ousmanethiare.com]

September 12, 2025

Outline

Outline

Need for a Coordinator

- Many algorithms used in distributed systems require a coordinator
  - For example, see the centralized mutual exclusion algorithm.
- In general, all processes in the distributed system are equally suitable for the role
- Election algorithms are designed to choose a coordinator.

Election Algorithms

- Any process can serve as coordinator
- Any process can "call an election" (initiate the algorithm to choose a new coordinator).
  - There is no harm (other than extra message traffic) in having multiple concurrent elections.
- Elections may be needed when the system is initialized, or if the coordinator crashes or retires.

Assumptions

- Every process/site has a unique ID; e.g.
  - the network address
  - a process number
- Every process in the system should know the values in the set of ID numbers, although not which processors are up or down.
- The process with the highest ID number will be the new coordinator.
- Process groups (as with ISIS toolkit or MPI) satisfy these requirements.

Requirements

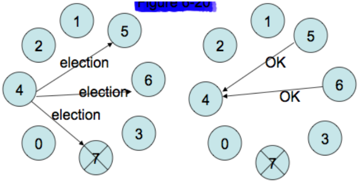
- When the election algorithm terminates a single process has been selected and every process knows its identity.
- Formalize: every process p_i has a variable e_i to hold the coordinator's process number.
  - $\forall i, e_i$=undefined or $e_i = P$, where P is the non-crashed process with highest id
  - All processes (that have not crashed) eventually set $e_i = P$.

Outline

The Bully Algorithm - Overview

- Process p calls an election when it notices that the coordinator is no longer responding.
- High-numbered processes "bully" low- numbered processes out of the election, until only one process remains.
- When a crashed process reboots, it holds an election. If it is now the highest- numbered live process, it will win.

The Bully Algorithm - Overview

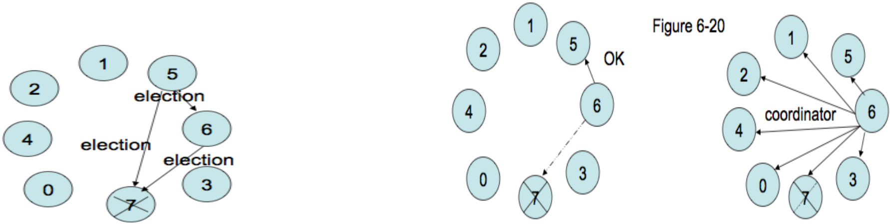


Process p sends an election message to all higher-numbered processes in the system. If no process responds, then p becomes the coordinator.

If a higher-level process (q) responds, it sends p a message that terminates p's role in the algorithm

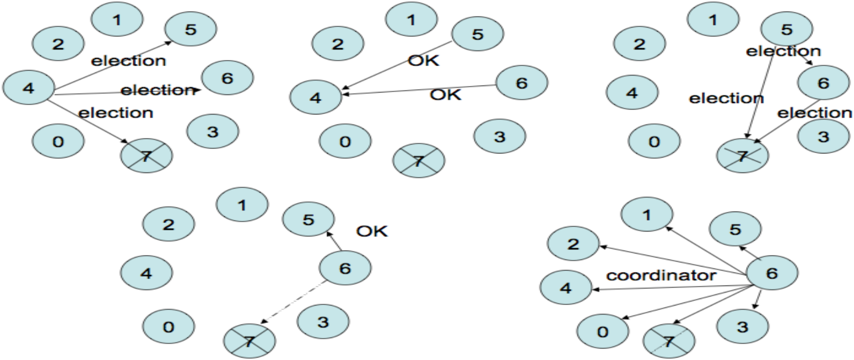
The Bully Algorithm - Overview

The process q now calls an election (if it has not already done so). Repeat until no higher-level process responds. The last process to call an election "wins" the election. election
The winner sends a message to other processes announcing itself as the new coordinator.



If 7 comes back on line, it will call an election

The Bully Algorithm - Overview



Analysis

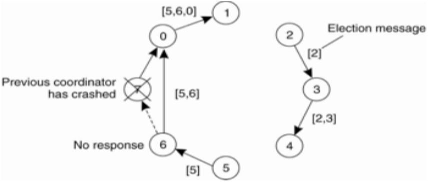
- Works best if communication in the system has bounded latency so processes can determine that a process has failed by knowing the upper bound (UB) on message transmission time (T) and message processing time (M).
  - UB = 2*T+M
- However, if a process calls an election when the coordinator is still active, the coordinator will win the election.

A Ring Algorithm - Overview

- The ring algorithm assumes that the processes are arranged in a logical ring and each process is knows the order of the ring of processes.
- Processes are able to "skip" faulty systems: instead of sending to process j, send to j+1.
- Faulty systems are those that don't respond in a fixed amount of time.

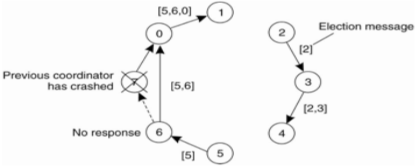
A Ring Algorithm

- P thinks the coordinator has crashed; builds an ELECTION message which contains its own ID number.
- Sends to first live successor
- Each process adds its own number and forwards to next.
- OK to have two elections at once.



A Ring Algorithm - Details

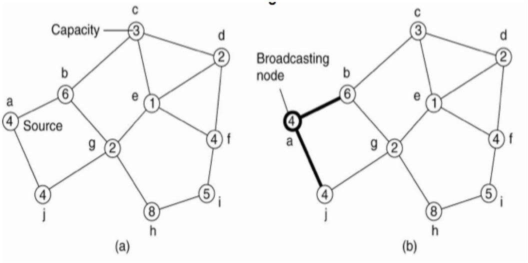
- When the message returns to p, it sees its own process ID in the list and knows that the circuit is complete.
- P circulates a COORDINATOR message with the new high number.
- Here, both 2 and 5 elect 6:
  $[5, \mathbf{6}, 0, 1, 2, 3, 4]$
  $[2, 3, 4, 5, \mathbf{6}, 0, 1]$



Elections in Wireless Environments

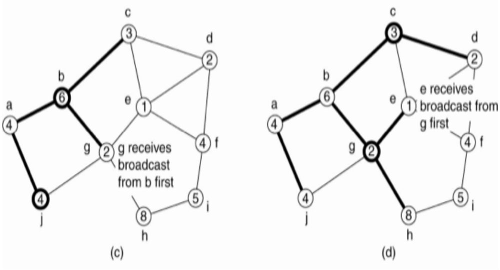
- Traditional algorithms aren't appropriate.
  - Can't assume reliable message passing or stable network configuration
- This discussion focuses on ad hoc wireless networks but ignores node mobility.
  - Nodes connect directly, no common access point, connection is short term
  - Often used in multiplayer gaming, on-the-fly file transfers, etc.

Assumptions



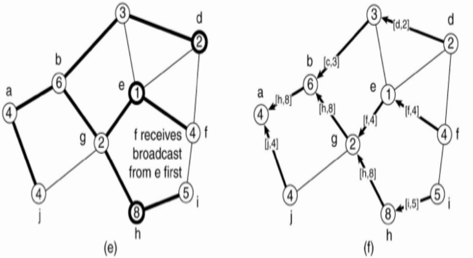
- Node a is the source.
  Messages have a unique ID to manage possible concurrent elections

Assumptions



- When a node R receives its first election message, it designates the source Q as its parent, and forwards the message to all neighbors except Q.
- When R receives an election message from a non-parent, it just acknowledges the message

Assumptions



- If R's neighbors have parents, R is a leaf; otherwise it waits for its children to forward the message to their neighbors.
- When R has collected acks from all its neighbors, it acknowledges the message from Q.

  Acknowledgements flow back up the tree to the original source.

Wireless Elections

- At each stage the "most eligible" or "best" node will be passed along from child to parent.
- Once the source node has received all the replies, it is in a position to choose the new coordinator.
- When the selection is made, it is broadcast to all nodes in the network.

Wireless Elections

- If more than one election is called (multiple source nodes), a node should participate in only one.
- Election messages are tagged with a process id.
- If a node has chosen a parent but gets an election message from a higher numbered node, it drops out of the current election and adopts the high numbered node as its parent. This ensures only one election makes a choice.

Summary

- "Synchronization is ... doing the right thing at the right time."
- Synchronization in distributed systems is related to communication.
- Complicated by lack of global clock, shared memory.
- Logical clocks support global event order.
- Distributed mutex: important class of synchronization algorithms.
- Leader election algorithms are sometimes needed.