Week 5: Part 2
Leader Election
Leader Election

• Purpose
  – Need to pick one process to act as coordinator

• Assumptions
  – Processes have no distinguishing characteristics
  – Each process has a unique ID to identify itself
  – Reliable message delivery
Goal: Select the process with the largest ID as the leader

• Holding an election: when process $P_i$ detects a dead leader:
  – Send *election* message to all processes with higher IDs
    • If nobody responds, $P_i$ wins and takes over
    • If any process responds, $P$’s job is done
  – Optional: Let all nodes with lower IDs know an election is taking place

• If a process receives an *election* message
  – Send an *OK* message back
  – Hold an election (unless it is already holding one)
Bully algorithm

• A process announces victory:
  – Sends all processes a message telling them that it is the new leader

• If a dead process recovers
  – It holds an election to find the leader
Bully algorithm

Rule: highest # process is the leader

Suppose $P_5$ dies

$P_2$ detects $P_5$ is not responding
**Bully algorithm**

P₂ starts an election

Contacts all higher-numbered systems
Everyone who receives an *election* message responds
... and holds their own election, contacting higher # processes

Example: $P_3$ receives the message from $P_2$
  Responds to $P_2$
  Sends *election* messages to $P_4$ and $P_5$
Bully algorithm

P₄ responds to P₃ and P₂'s messages
... and holds an election
Bully algorithm

Nobody responds to $P_4$

After a timeout, $P_4$ declares itself the leader
Ring election algorithm

Ring arrangement of processes

• **Holding an election**: if any process $P_i$ detects failure of leader
  – Construct an *election* message containing the ID of $P_i$ and send it to the clockwise neighbor (the successor)
  – If the successor is down, skip over it and try the process after that
  – Repeat until a running process is located

• Upon receiving an *election* message
  – Process forwards the message, adding its process ID to the body

Eventually message returns to originator

• The process receives an *election* message and sees its ID is at the head of the list
• Multicast a *leader* message announcing the new leader
  – E.g., highest numbered process
Ring algorithm

Assume $P_1$ discovers that the leader, $P_5$, is dead
$P_1$ starts an election

Election: $\{P_1\}$
Ring algorithm

Election: \{P_1, P_2\}
Ring algorithm

Election: \{P_1, P_2, P_3\}
Ring algorithm

Election: \{P_1, P_2, P_3, P_4\}

Fails: \(P_5\) is dead
Ring algorithm

Election: \{P_1, P_2, P_3, P_4\}

*Skip to P₀*
Ring algorithm

Election: \{P_1, P_2, P_3, P_4, P_0\}
P₂ receives the election message that it initiated
P₂ now picks a leader (e.g., highest ID)

Because P₁ sees its ID at the head of the list, it knows that this is the election that it started.

We might have multiple concurrent elections. Everyone must choose the same leader. Here, we agree to pick the highest ID in the list.
Ring algorithm

P₁ announces that P₄ is the new leader to the group

Many other election algorithms that target other topologies: mesh, torus, hypercube, trees, …
Chang & Roberts Ring Algorithm

**Optimize the ring election algorithm**

- The message always contains *one* process ID (PID)
- Try to avoid multiple circulating elections

- If a process sends an *election* message, it marks its state as a *participant*
  - This allows it to cut off extra elections

- Assume highest # PID is the winner
Chang & Roberts Ring Algorithm

Upon receiving an election message:

- If \( \text{PID(message)} > \text{PID(process)} \) – *higher ID will always win over a lower one*
  forward the message

- If \( \text{PID(message)} < \text{PID(process)} \) AND process is not a participant – *we are a higher ID number; use ours*
  replace PID in message with \( \text{PID(process)} \) and forward the new message
  set the process state to *participant*

- If \( \text{PID(message)} < \text{PID(process)} \) AND process is participant
  discard the message
  – *we’re already circulating our ID and it’s a higher number than this one*

- If \( \text{PID(message)} == \text{PID(process)} \)
  the process is now the leader
  – *message fully circulated to the one who started: announce winner*
Elections & Network Partitions

• Network **partitions** (segmentation)
  – Multiple nodes may decide they’re the leader
  – Leads to multiple groups, each with a leader & diverging data among them → **split brain**

• Dealing with partitions
  – Insist on a majority → if no majority, the system will not function
    • Quorum = minimum # of participants required for a system to function
  – Rely on alternate communication mechanism to validate failure
    • Redundant network, shared disk (but that can also fail)
The End