Election Algorithm in Distributed Systems

Bully Algorithm

Bully Algorithm Main Characteristics

- **Operating Assumptions**
  - Synchronous system
    - All messages arrive within T_M units transmission of time.
    - A reply is dispatched within P_p units of processing time after the receipt of a message.
    - If no response is received in $2 \times T_M + P_P$, the node is assumed to be faulty
      - Node crashed
  - Attribute = Process ID
  - Each process knows all the other processes in the system
    - Therefore, processes know each others' IDs

The Bully Algorithm

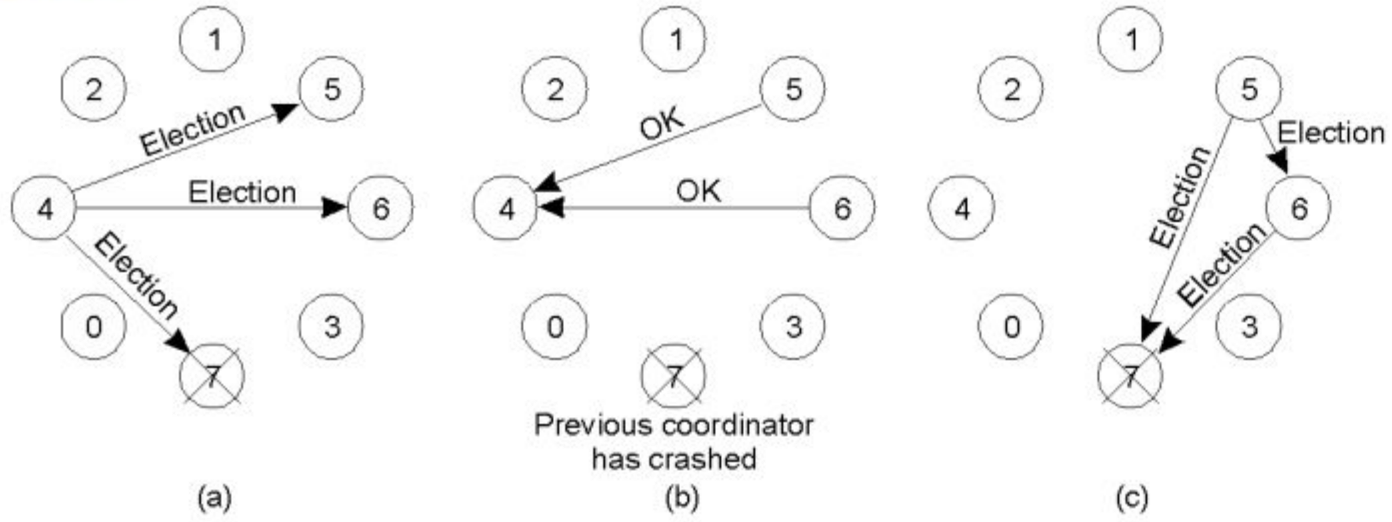
When any process, P, notices that the coordinator is no longer responding it initiates an election:

1. P sends an ***election*** message to all processes with higher id numbers.
2. If no one responds, P wins the election and becomes coordinator.
3. If a higher process responds, it takes over.
   - Process P's job is done.

The Bully Algorithm

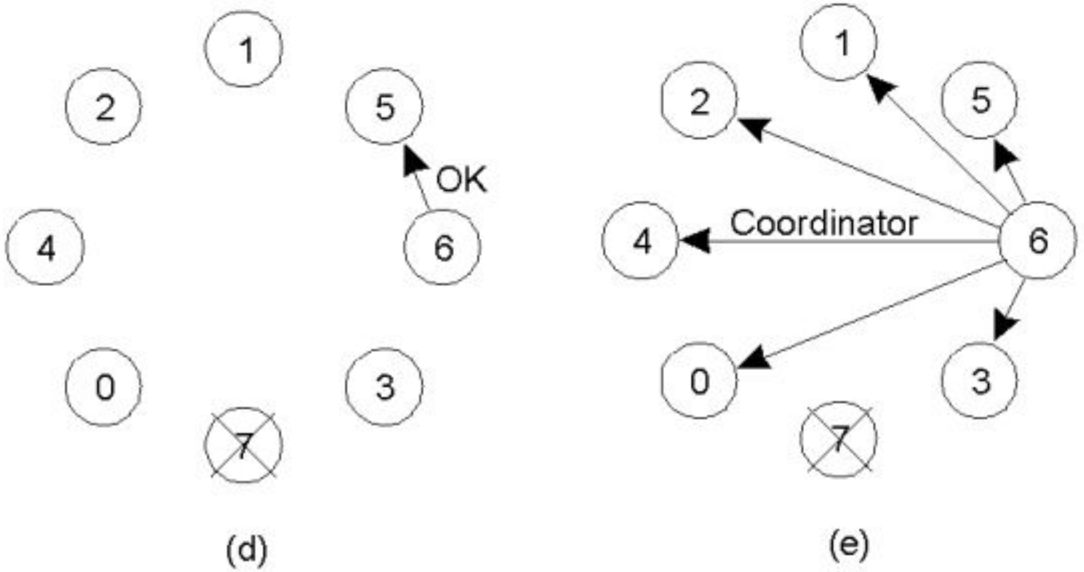
- At any moment, a process can receive an **election** message from one of its lower-numbered colleagues.
- The receiver sends an OK back to the sender and conducts its own election.
- Eventually only the **bully process** remains.
  - Bully process becomes the new cooridinator
  - The bully announces victory to all processes in the distributed group.

Bully Algorithm Example



- Process 4 notices 7 down.
- Process 4 holds an election.
- Process 5 and 6 respond, telling 4 to stop.
- Now 5 and 6 each hold an election.

Bully Algorithm Example



- Process 6 tells process 5 to stop
- Process 6 (the bully) wins and tells everyone
- If processes 7 recovers, it restarts election process

Performance of 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.
- Worst case scenario: When the process with the least id detects the failure.
  - N-1 processes altogether begin elections, each sending messages to processes with higher ids.
  - The message overhead is $O(N^2)$.
  - Turnaround time is approximately 5 message transmission times if there are no failures during the run: election, answer, election, answer, coordinator

Ring Algorithm

Ring Algorithm – Basic Operation

- RA assumes that the processes are logically ordered in a ring **{implies a successor pointer and an active process list}** that is unidirectional.
- When any process, P, notices that the coordinator is no longer responding it initiates an election:

1. P sends message containing **P's process ID** to the next available successor.

Ring Algorithm – Basic Operation

2. At each active process, the receiving process adds its process number to the list of processes in the message and forwards it to its successor.
   - Eventually, the message gets back to the sender.

3. The initial sender sends out a second message letting everyone know who the coordinator is **{the process with the highest number}** and indicating the current members of the active list of processes.