MIT-WPU

॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

BTech - CSE [2022 - 23]

Distributed Computing Lab

Assignment - 5

Title : Election Algorithms

Aim : To study and implement the following :-

1. Ring Algorithm
2. Bully AlgorithmElection Algorithms

Election Algorithms

- There is a collection of processes/nodes that are identified by a unique number
- A connection is established amongst the processes/nodes,hence these can communicate with each other.Sending and Receiving happens between these processes.
- A coordinator(leader) process needs to be elected through the election algorithms
- And The coordinator election is done through election algorithms

Why elect a coordinator process ?

- A coordinator process can be a decision maker eg In banking,a coordinator can decide which server shall respond to transactions
- A coordinator can be a resource allocator
- A coordinator can be responsible for task divisions for distributed computing

Types of messages

There are basically 3 types of messages :

1. An election message to initiate the election

2. A reply/response message given in response to the election message

3. A coordinator message sent to inform other processes, the id of the coordinator process.

Election Algorithm : Assumptions/Requirements

1. Any process can call for an election.

2. A process can call for at most one election at a time.

3. Multiple processes can call an election simultaneously.

4. The result of an election should not depend on which process calls for it.

5.Each process has

  Variable called *elected*

  An attribute value called *attr,* e.g., id, MAC address, CPU

6.The non-faulty process with the best (highest) election attribute value (e.g., highest id or address, or fastest cpu, etc.) is elected.

7. Requirement: A *run* (execution) of the election algorithm must always guarantee at the end: a) safety and b) liveliness [non-faulty process]

Ring algorithm

1.N Processes are organized in a logical ring.

p_i has a communication channel to $p_{(i+1) \bmod N}$.

All messages are sent clockwise around the ring.

2.Any process p_i that discovers a coordinator has failed initiates an “election” message $<i, p_i.attr>$

3.When a process p_j receives an *election* message $<i, p_i.attr>$, it compares the *attr* in the message with its own.

If the arrived $p_i.attr > p_j.attr$, then receiver p_j forwards the message $<i, p_i.attr>$.

If the arrived $p_i.attr < p_j.attr$ and the receiver p_j has not forwarded an election message earlier, it substitutes its own $<j, p_j.attr>$ in the message and forwards it.

If the arrived $p_i.attr = p_j.attr$, then this process’s $p_j.attr$ must be the greatest, and it becomes the new coordinator. This process then sends an “elected” message to its neighbor announcing the election result.

4.When a process p_i receives an elected message, it

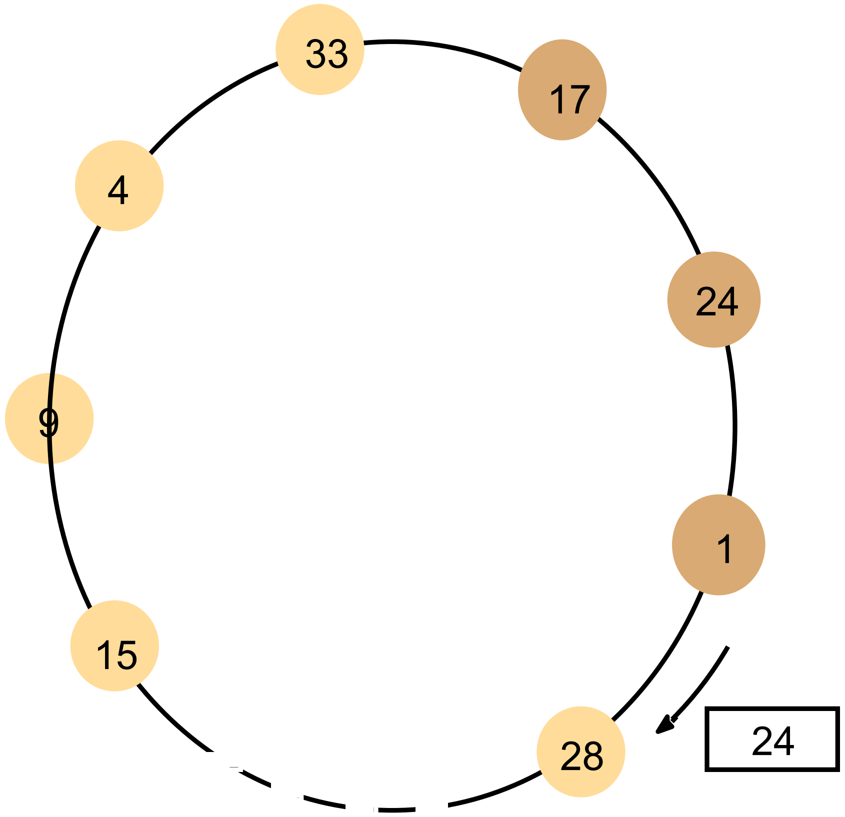
sets its variable $elected_i$ □ id of the message.

forwards the message if it is not the new coordinator.

A participant participates in election and becomes non participant when election is over and elected message reaches it

Ring Algorithm : Progress

Ring of N processes



Note: The election was started by process 17.
The highest process identifier encountered so far is 24.
(final leader will be 33)

Ring Algorithm : Time Complexity

(attr:=id)

1.The worst-case scenario occurs when the counter-clockwise neighbor has the highest *attr/id*

2. A total of *N-1* messages is required to reach the new coordinator-to-be.

3.Another *N* messages are required until the new coordinator-to-be ensures it is the new coordinator.

4.Another *N* messages are required to circulate the elected messages.

worst case : 3N-1

best : 2N

assume : one process began election

ALTERNATIVELY :-

In ring algorithm, on the contrary, irrespective of which process detects the failure of coordinator and initiates an election, an election always requires 2(n-1) messages. (n-1) messages needed for one round rotation of the ELECTION message, and another (n-1) messages for the COORDINATOR message.

Modified Ring Algorithm

1. Processes are organized in a logical ring.
2. Any process that discovers the coordinator (leader) has failed initiates an “election” message. This is the *initiator* of the election.
3. The message is circulated around the ring, bypassing failed nodes.
4. Each node adds (appends) its *id:attr* to the message as it passes it to the next node.
5. Once the message gets to the initiator, it elects the node with the best election attribute value.
6. It then sends a “coordinator” message with the id of the newly-elected coordinator. Again, each node adds (appends) its *id* to the end of the message.

Once “coordinator” message gets back to initiator,

- election is over if “coordinator” is in id-list.
- else the algorithm is repeated (handles election failure).

Bully Algorithm

1.A node initiates election by sending an “election” message to only nodes that have a higher id than itself.

If no answer, announce itself to lower nodes as coordinator.

if any answer, then there is some higher node active; wait for coordinator message. If none received after time out, start a new election.

2.A node that receives an “election” message replies with answer, & starts an election – unless it has already.

3. When a process finds the coordinator has failed, if it knows its id is the highest, it elects itself as coordinator, then sends a *coordinator* message to all processes with lower identifiers.

Bully Algorithm : Progress

1.P sends an *ELECTION* message to all processes with higher numbers.

2.If no one responds, *P* wins the election and becomes coordinator.

3.If one of the higher-ups answers, it takes over. *P*’s job is done.

The Bully Algorithm (1)

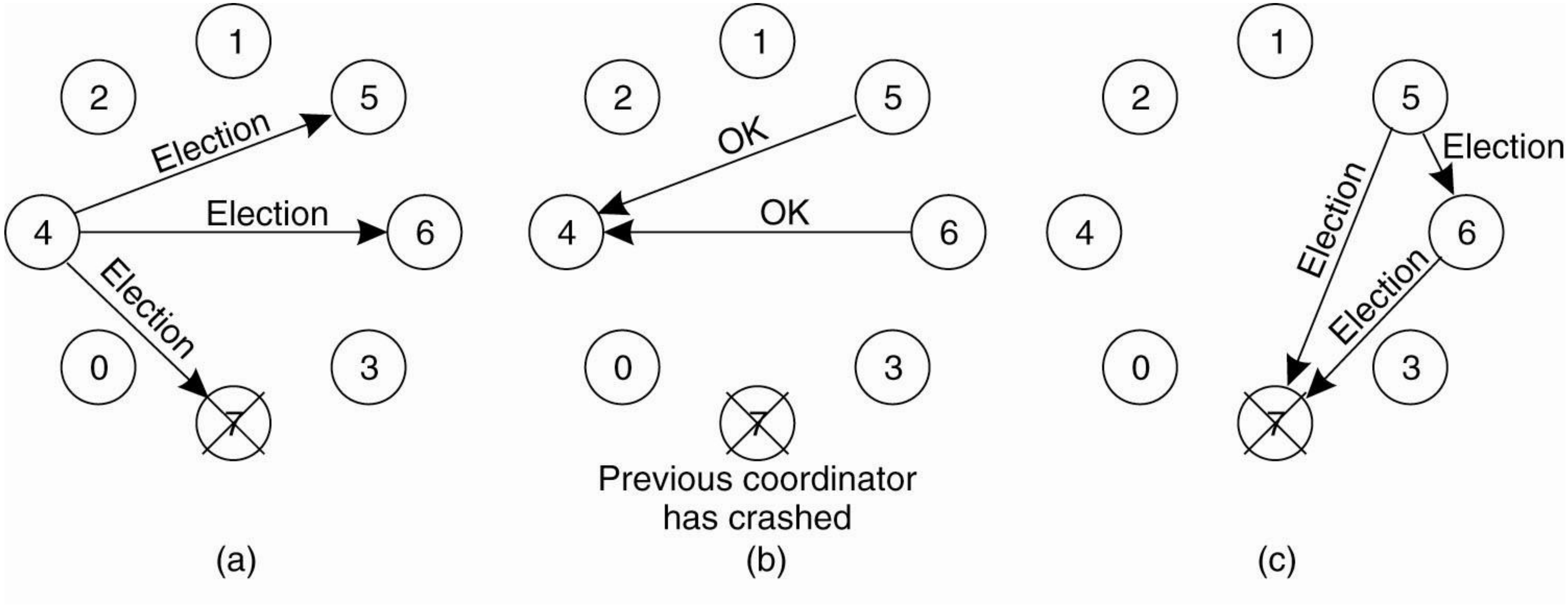


Figure 6-20. The bully election algorithm. (a) Process 4 holds an election. (b) Processes 5 and 6 respond, telling 4 to stop. (c) Now 5 and 6 each hold an election.

The Bully Algorithm (2)

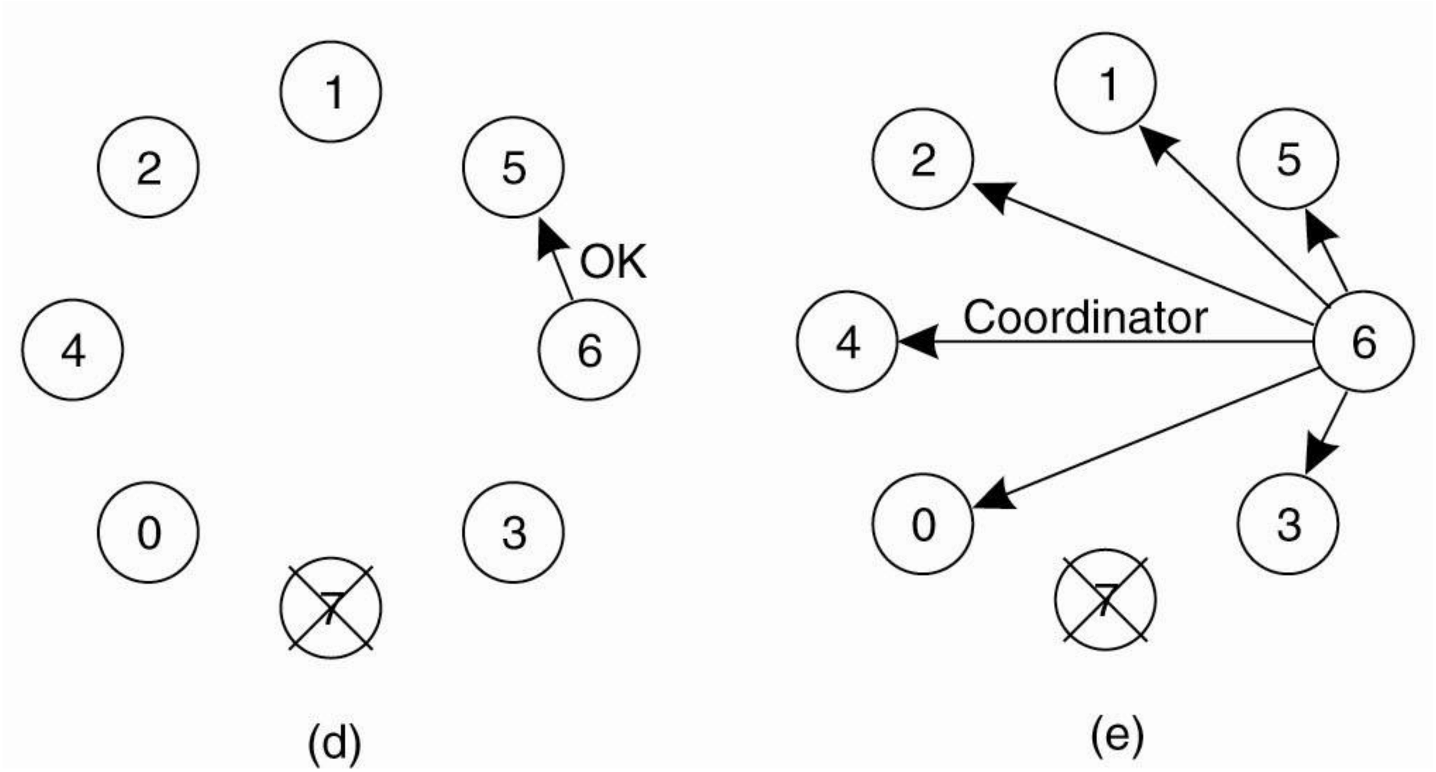


Figure 6-20. The bully election algorithm. (d) Process 6 tells 5 to stop. (e) Process 6 wins and tells everyone.

Bully Algorithm : Time Complexity

- In the worst case,the bully algorithm requires O(n^2) messages. When the process having the priority number just below the failed coordinator detects failure of coordinator, it immediately elects itself as the coordinator and sends n-2 coordinator messages.

- (n-1) messages in the best case.