



LEADER ELECTION

National Institute of Technology Rourkela

Leader Election Algorithms

- **Leader election is the simple idea of giving one thing (a process, host, thread, object, or human) in a distributed system some special powers such as:**
  - **the ability to assign work,**
  - **the ability to modify a piece of data, or**
  - **even the responsibility of handling all requests in the system.**

- **Advantages: a powerful tool for improving efficiency, reducing coordination, simplifying architectures, and reducing operations.**

- **Disadvantages: Can introduce new failure modes and scaling bottlenecks. It may make it more difficult for you to evaluate the correctness of a system.**

Requirement of Leader Election

Typically leader election is used:

To ensure exclusive access by a single node to shared data, or

To ensure a single node coordinates the work in a system.

For replicated database systems such as MySQL, Apache Zookeeper, or Cassandra, we need to make sure only one "leader" exists at any given time.

Applications of LEAs

Radio networks:

In radio network protocols, leader election is often used as a first step to approach more advanced communication primitives, such as message gathering or broadcasts.

When adjacent nodes transmit at the same time in wireless networks (very natural) induces collisions; electing a leader allows to better coordinate this process.

While the diameter D of a network is a natural lower bound for the time needed to elect a leader, upper and lower bounds for the leader election problem depend on the specific radio model.

RDBMS:

RDBMSs rely on leader election to pick a leader database which handles all writes, and sometimes, all reads where election may be automated, but it's frequently done manually by a human operator.

Election Algorithms

Many distributed algorithms need one process to act as a coordinator for coordinating all the activities in a distributed system

Election algorithms are to pick a unique coordinator/leader based on certain criteria such as largest identifier

Examples:
(1) Take over the role of a failed process (Fault Tolerance)

(2) Pick a master in Berkeley clock synchronization algorithm (Physical Clock Synchronization)

(3) A powerful tool used in systems across Amazon for fault-tolerance and easier to operate.

(4) Znode in Zookeeper is choosen as leader. All application processes watch the current smallest znode which is ephemeral (works for small duration of time), and check if they are the new leader when the smallest znode goes away

Leader Election Algorithms

Once the leader is elected, the nodes reach a particular state known as terminated state.

The states are partitioned into *elected states & non-elected states*.

When a node enters either state, it always remains in that state.

Safety and liveness condition for execution of Leader Election Algorithm:

Liveness condition: Every node will eventually enter an elected state or a non-elected state.

Safety condition: Only a single node enters the elected state & eventually become the leader.

Information is exchanged between nodes until an agreement is reached.

Once a decision is made, a node is elected as the leader and all the other nodes will acknowledge the role of that node as the leader

Validity of LEA

A LEA is valid if it meets the following conditions:

Termination: the algorithm should finish within a finite time once the leader is selected. In randomized approaches this condition is sometimes weakened (for example, requiring termination with probability 1).

Uniqueness: there is exactly one node that considers itself as leader.

Agreement: all other nodes know who the leader is.

Types of LEA

An algorithm for leader election may vary in the following aspects:

Communication mechanism: the nodes are either synchronous in which processes are synchronized by a clock signal or asynchronous where processes run at arbitrary speeds.

Process names: whether processes have a unique identity or are indistinguishable (anonymous).

Network topology: for instance, ring, acyclic graph or complete graph.

Size of the network: the algorithm may or may not use knowledge of the number of processes in the system.

Types of Leader Election Algorithms

The most prominent LE algorithms are:

a. Bully Algorithm presented by Gracia-Molina in 1982.

Improved Bully Election Algorithm by A. Arghavani in 2011.

Modified Bully Election Algorithm by M. S. Kordafshari and group.

b. Ring Algorithm

Modified Ring Algorithm

Bully Algorithm: Basic Assumptions

1. The system is synchronous
2. Each process has a unique numerical Id
3. Processes know the Ids and address of every other processes
4. Communication/Message Delivery b/n processes is reliable
5. The processes may fail at any time including during execution of algorithm
6. There is a failure detector
7. The process fails by stopping response message
8. Used to elect a coordinator dynamically from a set of distributed computing processes
9. The process with highest id among non-failed process is selected as coordinator

Bully Algorithm

Key idea:

- **Select process with highest Id**
- **Processes initiate election if just recovered from failure**
- **If coordinator failed several processes can initiate an election simultaneously**

Types of Messages:

- **Coordinator: For announcing the victory of election**
- **Election Message: To initiate election process**
- **Alive Message : To indicate the status of message**

O(n^2) messages are needed with n processes

Bully Algorithm

When a process P recovers from failure, or the failure detector indicates that the current coordinator has failed, P performs the following actions:

Step 1: If P has the highest process ID, it sends a Victory message to all other processes and becomes the new Coordinator. Otherwise, P broadcasts an Election message to all other processes with higher process IDs than itself.

Step 2: If P receives no Answer after sending an Election message, then it broadcasts a Victory message to all other processes and becomes the Coordinator.

Step 3: If P receives an Answer from a process with a higher ID, it sends no further messages for this election and waits for a Victory message. (If there is no Victory message after a period of time, it restarts the process at the beginning.)

Step 4: If P receives an Election message from another process with a lower ID it sends an Answer message back and starts the election process at the beginning, by sending an Election message to higher-numbered processes.

Step 5: If P receives a Coordinator message, it treats the sender as the coordinator.

Algorithm (Bully)

Step 1: Let process P sends a message to the coordinator

Step2: If coordinator does not respond to it within a time interval T, then it is assumed that coordinator has failed

Step 3: Now process P sends election message to every process with high priority number

Step 4: It waits for response, if no one responds for time interval T then process P elects itself as a coordinator

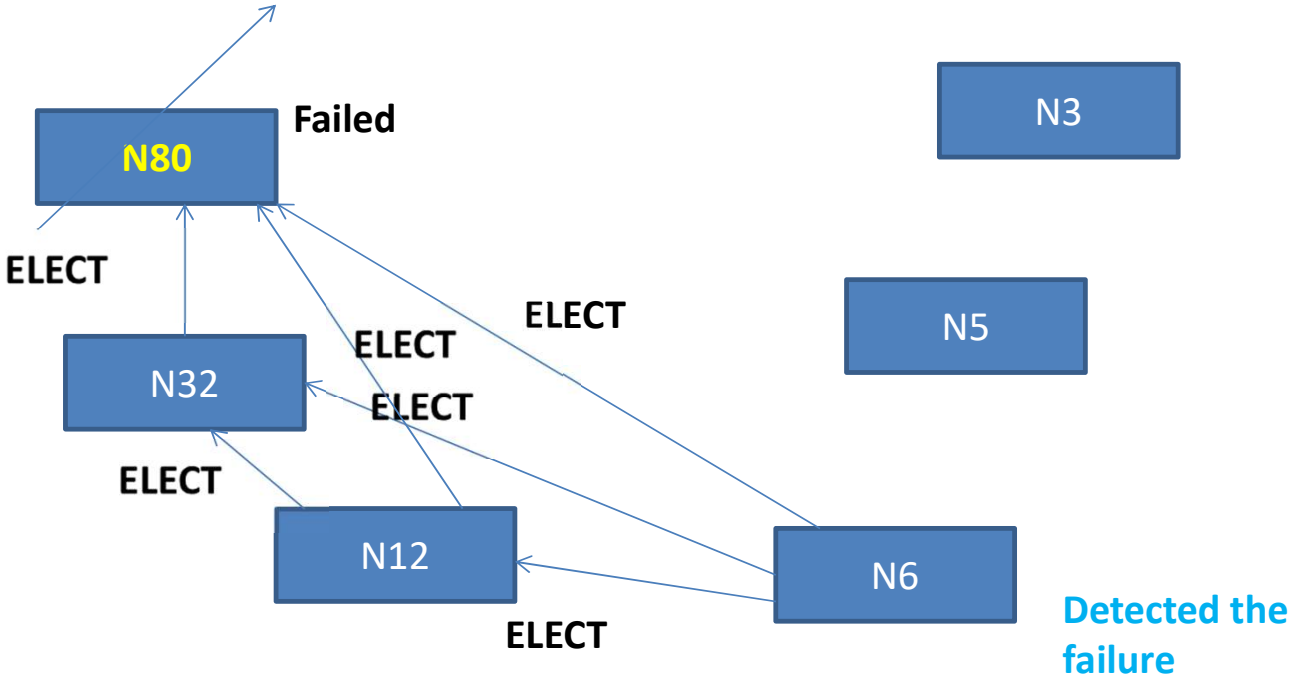
Step 5: Then it sends a message to all lower priority processes then it is elected as their new coordinator

Step 6: If an answer is received within time T from any other process Q

Process P again waits for time interval T to receive another message from Q that it has been elected as coordinator

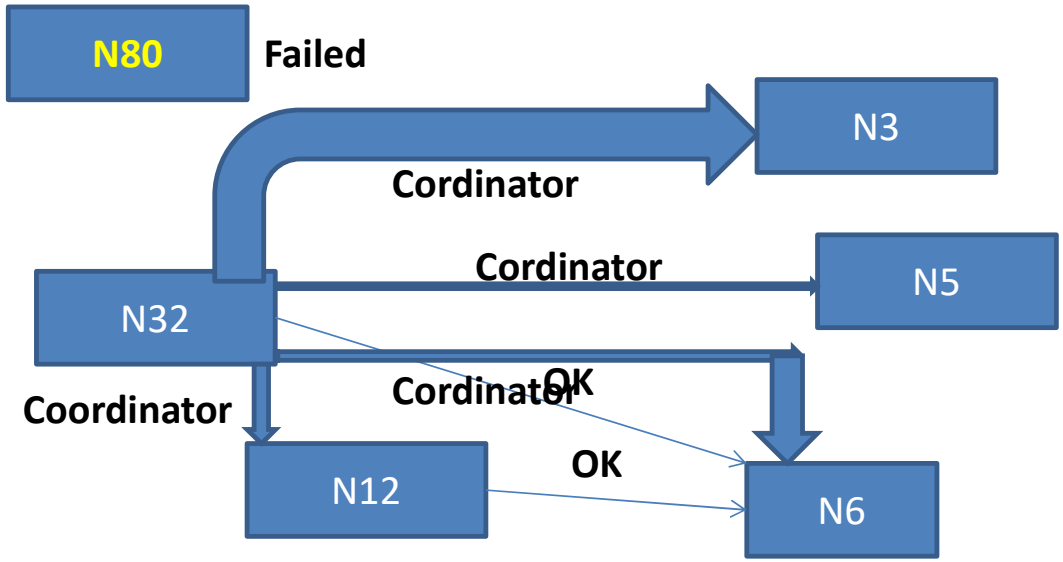
If Q does not respond within time interval T, then it is assumed to have failed and algorithm is reiterated

Example Bully Algorithm



Let process N80 fails and detected by node N6 with the help of a failure detector
N6 sends election messages to all processes having higher Ids i.e., N80, N32 & N12

N80 does not respond



N12 & N32 send OK or Alive messages to N6 being the higher ids than N6

N32 know the id of all other processes (Every process knows the ids of all other processes)

N32 sends Coordinator or Victory message to all lower Id processes & the Election is Complete

If failures is stop, eventually will elect a leader

How to set the timeouts ?

Answer: Based on worst case time to complete election

5 message transmissions time if there are no failures during the run

1. **Election from lowest id server in group**
2. **Answer to lowest id server from 2nd highest id process**
3. **Election from second highest id server to highest id**
4. **Timeout for answers @ 2nd highest id server**
5. **Coordinator from second highest id server**

Analysis

Worst case completion time: 5 message transmission times

When the process with lowest id in the system detects failure

N-1 processes altogether begin elections, each sending messages to processes with higher ids

ith highest id process sends (i-1) election messages

No. of election messages:

N-1 + N-2 + … +1 = (N-1)* N/2 = O(n^2)

Best case
Second highest id detects leader failure
Sends (N-2) coordinator messages
Completion time: 1 message transmission time

Impossibility

Since timeouts built into protocol, in asynchronous system model:

Protocol may never terminate -> liveness not guaranteed

But satisfy liveness in synchronous system model where

worst case one-way latency can be calculated = worst case process time + worst case process latency

Disadvantages of Bully Algorithm

(a) Space Complexity is very large since every process should know the identity of every other process in the system.

(b) High number of message passing during communication increases heavy traffic.

(c) The message complexity has order $O(n^2)$.

Improved Bully Algorithm

Presented by A.arghavani, E.ahmadi, A.T.haghighat in 2011.

Overcomes the disadvantages of the original bully.

The main concept: The algorithm declares the new coordinator before actual or current coordinator is crashed. (needs extra stages)

Before the coordinator is failed, the current coordinator tries to gather information about processes in the system and declares the next possible coordinator to the processes.

With increasing knowledge and get the id of all other process, a process with the bigger id attempts to execute the bully algorithm.

If the coordinator is failed, each process that notices this failure compares its id with the id which it has received via the coordinator.

And select the new coordinator

Disadvantages of Improved Bully Algorithm

It has complex structure.

Every time process updates its database.

Large database required to maintain the information of each process in database of every process.

MODIFIED ELECTION ALGORITHM

Presented by M.S. Kordafshari, M.Gholipour, M.jahanshahi, A.T.haghighat in 2005.

The algorithm resolve the disadvantages of the bully algorithm.

1. When any process p notices that coordinator is not responding, it initiates an election and send election message to all process with higher priority number.

2. If no process responds, process P wins the election and becomes new coordinator.

3. Process with the higher priority sends ok message with its priority number to process P.

4. When process p receive all the response it select the new coordinator with the highest priority number process and sends the grant message to it.

5. Now the coordinator process will broadcast a new coordinator message to all other process and informs itself as a coordinator.

Disadvantages of Modified Bully Algorithm

A modified algorithm is also time bounded.

It is better than bully but also has $O(n^2)$ complexity in worst case.

It is necessary for all process to know the priority of other.

Ring Algorithm

The algorithm applies to system organized as a ring (logically or physically)

Assumptions: The link between the processes are unidirectional and every process can manage to the processes on its right only (Clockwise)

Data Structures used in the algorithm: Active List
i.e., a list that has priority number of all active processes in the system

Highest priority number is the coordinator

Algorithm: Ring

Step1: If process P1 detects a coordinator failure, it creates a new active list which is empty initially.

It sends election message to its neighbor on right and adds number 1 to its active list.

Step 2: If process P2 receives a message elect from processes on left, it responds in 3 ways:

(i) If msg received does not contain 1 in active list then P1 adds 2 to its active list & forward the message

(ii) If this is the first election message it has received or sent, P1 creates new active list with numbers 1 and 2.

It then sends election message 1 followed by 2:

(iii) If process P1 receives its own election message 1 then active list for P1 now contains numbers of all the active processes in the system. Now process P1 detects highest priority number from list and elects it as the next coordinator.

Example: Ring Algorithm

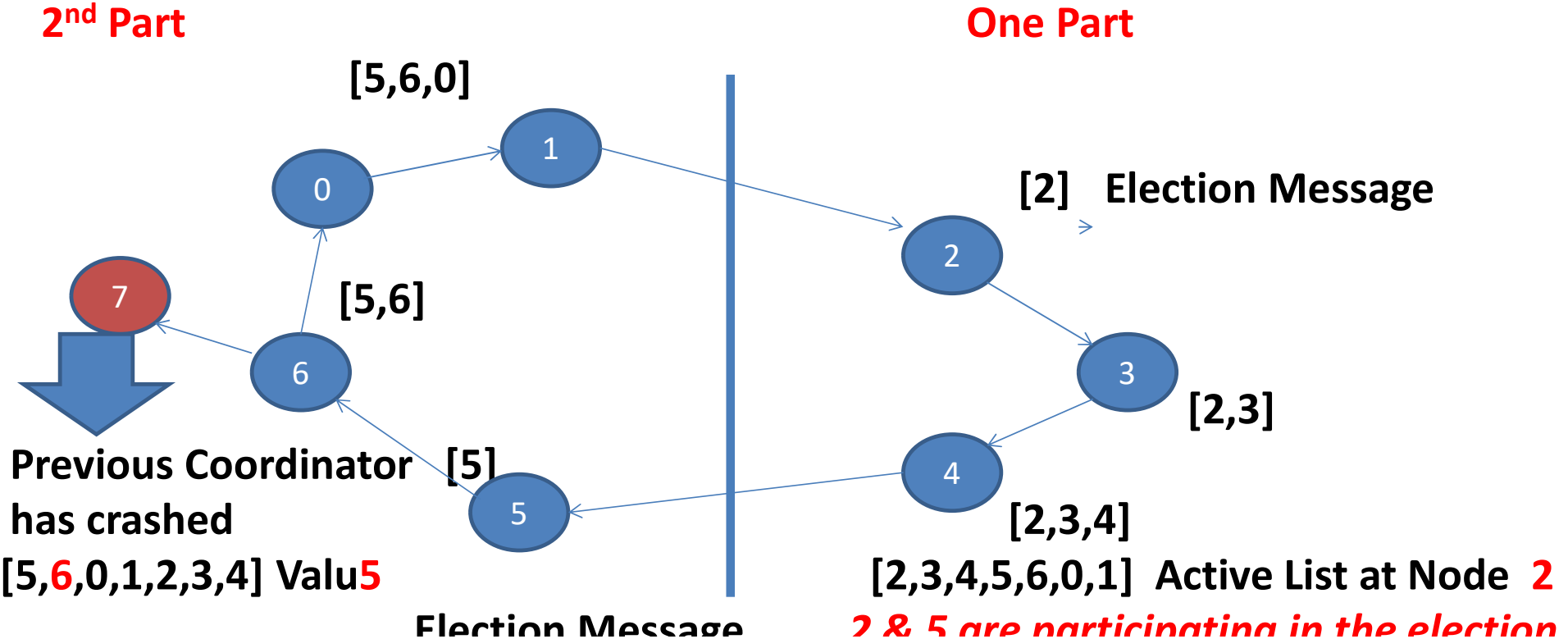
0-7 Processes are participating in the network

P thinks the coordinator has crashed, builds an election message which contains its own id number (Process 6)

Sends to first live successor; (Ex. Node 5 sends id 5 to node 6)

Each process adds its own number and forwards to next

O.K to have two elections at once



Example: Ring Algorithm

When the message returns to P, it sees its own process ID in the list & knows that the circuit is complete

P circulates a "COORDINATOR" message with the new high number

Here both 2 and 5 elected 6 as the leader

[5, 6, 0, 1, 2, 3, 4]
[2, 3, 4, 5, 6, 0, 1] 6 Coordinator

MODIFIED RING ALGORITHM

When a node notices that the leader has crashed, it sends its ID number to its neighboring node in the ring. Thus, it is not necessary for all nodes to send their IDs into the ring.

The receiving node compares the received ID with its own, and forwards whichever is the greatest. This comparison is done by all the nodes such that only the greatest ID remains in the ring.

Finally, the greatest ID returns back to the initial node.

If the received ID equals that of the initial sender, it declares itself as the leader by sending a coordinate message into the ring.

It can be observed that this method dramatically reduces the overhead involved in message passing.

Thus, if many nodes notice the absence of the leader at the same time, only the message of the node with the greatest ID circulates in the ring thus, preventing smaller IDs from being sent.

If n{i1,i2,⋯ ,im} is the number of nodes that concurrently detect the absence of the crashed coordinator and n is the number of nodes in the ring, then the total number of messages passed with an order of O(n2) is as follows:

$T = n\{i1i2,\cdots ,im\} \times n.$

- **Leader election is an important component of many cloud computing systems**
- **Classical leader election protocols: Ring and Bully**
- **But Failure Prone**
- **Paxos like protocols used by google Chubby, Apache Zookeeper**

Applications of Leader Election

In Wireless Networks:

- **Key distribution,**
- **Routing coordination,**
- **Sensor coordination, and**
- **General control.**

In Cloud Computing:

- **Resolving Conflicts During Resource sharing**

How Amazon elects a leader ?

There are many ways to elect a leader, ranging from algorithms like Paxos, to software like Apache ZooKeeper, to custom hardware, to leases.

Leases:

- **are the most widely used leader election mechanism at Amazon.**
- **are relatively straightforward to understand and implement**
- **offer built-in fault tolerance.**
- **work by having a single database that stores the current leader.**
- **requires that the leader heartbeat periodically to show that it's still the leader.**

If the existing leader fails to heartbeat after some time, other leader candidates can try to take over.

Examples of systems using leader election at Amazon

Leader election is a widely deployed pattern across Amazon.

For example:

RDBMSs rely on leader election to pick a leader database which handles all writes and sometimes all reads.

The election may be automated but it is frequently done manually by a human operator.

Examples of systems using leader election at Amazon

Amazon EBS (Elastic Block Store) distributes reads and writes for a volume (Solid State Drives/Hard Disk Drives) over many storage servers.

To ensure consistency, it uses leader election to elect primaries for each area of the volume which order the reads and writes.

If primary fails, follower copies steps in using the same leader election mechanism.

Leader election ensures consistency while improving performance by avoiding coordination on the data plane.

DynamoDB, Amazon Quantum Ledger Database (Amazon QLDB), and Amazon Kinesis (Kinesis) use similar approaches for the same reason.

Examples of systems using leader election at Amazon

DynamoDB:

Uses leader election protocol to elect a AWS Management Console to monitor resource utilization and performance metrics of various operations over data bases

Amazon Quantum Ledger Database (Amazon QLDB):

Elect a central trusted authority to provide a fully managed ledger database that provides a transparent, immutable, and cryptographically verifiable transaction log

Amazon Kinesis (Kinesis) :

The Kinesis Client Library (KCL) uses leases to ensure that each Kinesis shard is processed by one owner, making it easy to do scale-out processing of Kinesis streams.

What happens when leader fails?

Allows the new leader to confidently redrive work that the outgoing leader may have partially completed or completed but didn't tell others about.

To tolerate failures, Amazon distributed systems don’t have a single leader. Instead, leadership is a property that passes from server to server, or process to process.

In distributed systems, it’s not possible to guarantee that there is exactly one leader in the system. Instead, there can mostly be one leader, and there can be either zero leaders or two leaders during failures.

Idempotent can often tolerate two leaders with minimal loss of efficiency

The leader election system must always be correct and consistent.

- **Systems having at most one leader**
- **Systems having multiple leaders**

Characteristics of a Good Leader Election

Frequent Checkpointing: Frequent check of the remaining lease time (or lock status in general) especially before initiating any operation that has side-effects beyond the leader itself.

Network Latency: Consider that slow networking, timeouts, retries, and garbage collection pauses can cause the remaining lease time to expire before the code expects it to.

Correctness: Avoid heartbeating leases in a background thread. This can cause correctness issues if the thread can't interrupt the code when the lease expires or the heartbeating thread dies.

Availability: This issues can occur if the work thread dies or stops while the heartbeating thread holds on to the lease.

Characteristics of a Good Leader Election

Reliability: Have reliable metrics that show how much work a leader can do versus how much it is doing now.

Scalability: Review the metrics often and make sure that there are plans for scaling in advance of running out of capacity.

Flexibility: Make it easy to find which host is the current leader and which host was the leader at any given time. Keep an audit trail or log of leadership changes.

Formal Verification Tools: Model and formally verify the correctness of distributed algorithms using tools like TLA+.

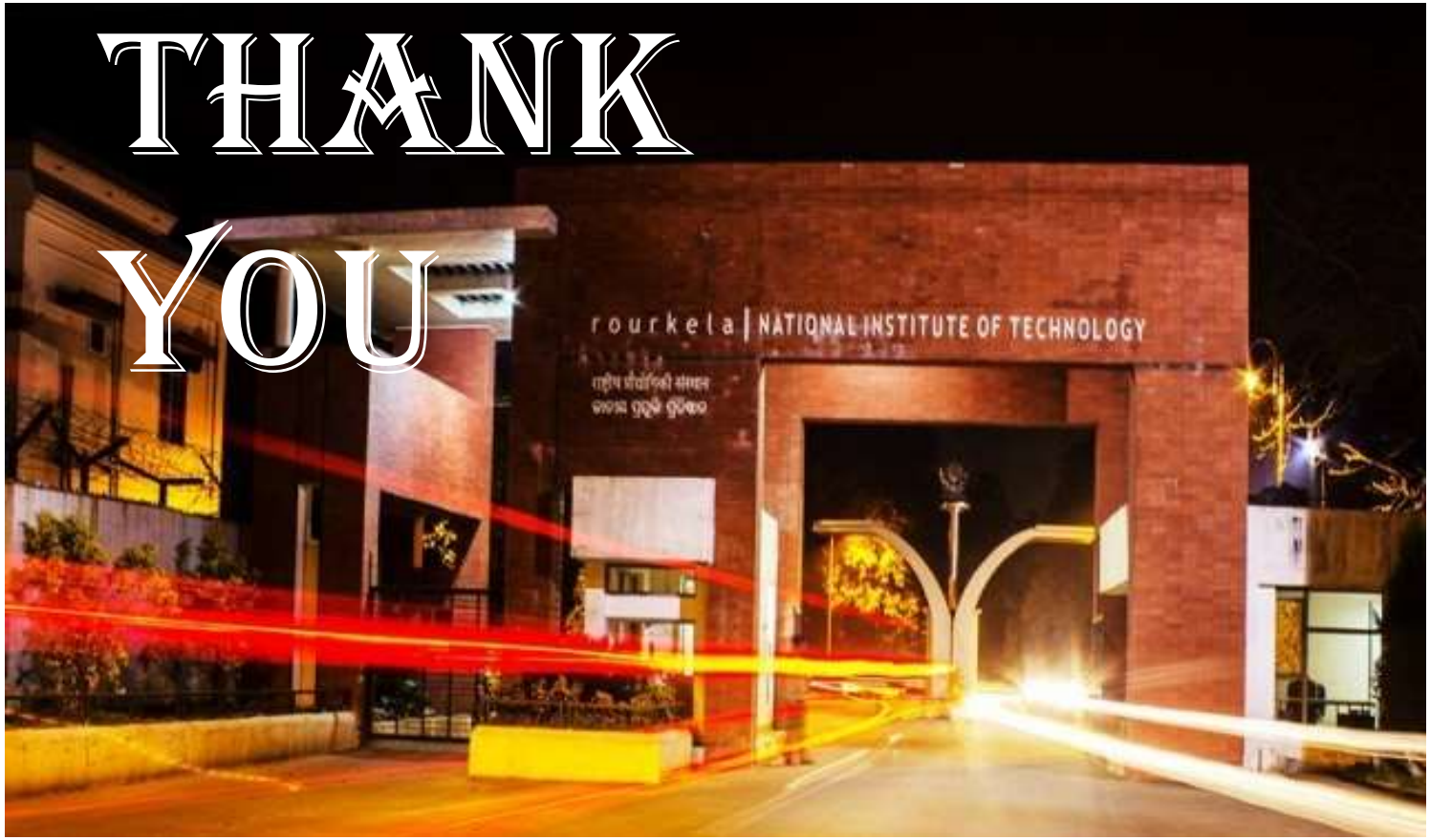
Bug Tolerance: This catches subtle, difficult to observe, and rare bugs that can creep in when an application assumes too much about the guarantees provided by the leader election protocol.

References

1. https://www.coursera.org/lecture/cloud-computing-2/1-4-bully-algorithm-K8QwJ

2. https://aws.amazon.com/builders-library/leader-election-in-distributed-systems/

3. Seema Balhara, Kavita Khanna, Leader Election Algorithms in Distributed Systems, International Journal of Computer Science and Mobile Computing, Vol. 3, Issue. 6, June 2014, pg.374 – 379


THANK
YOU
rourkela | NATIONAL INSTITUTE OF TECHNOLOGY