BHARATIYA VIDYA BHAVANS
SARDAR PATEL INSTITUTE OF TECHNOLOGY
MUNSHI NAGAR, ANDHERI (WEST), MUMBAI - 400 058
AUTONOMOUS INSTITUTE AFFILIATED TO MUMBAI UNIVERSITY

Adwait Purao - 2021300101

TE Comps B - Batch B

DC LAB

DISTRIBUTED COMPUTING EXPERIMENT 6

Aim:

Implementation of Election Algorithms

Theory:

- Distributed computing involves the coordination and management of multiple processes running on various processors in a networked environment. In many distributed systems, it is essential to have a single process act as a coordinator or leader to ensure organized and efficient operations.
- The challenge lies in determining which process should assume this role, and this is where election algorithms come into play.
- Election algorithms are designed to elect a coordinator process from the currently running processes in a way that ensures there is always a single coordinator at any given time.

The primary goal of an election algorithm is to guarantee that when an election is initiated, it concludes with all the processes reaching a consensus on which process should be the coordinator. This theory section explores the

concept of election algorithms and focuses on two specific algorithms: the Bully Algorithm and the Ring Algorithm.

Election Algorithms

Coordinator Election Problem

The coordinator election problem is defined as the task of selecting one process from a set of processes running on various processors in a distributed system to serve as the central coordinator. The fundamental requirement is that this process should be elected in a distributed and coordinated manner. In this context, the following assumptions are made:

- Multiple processes are running on distinct machines within the distributed system.
- Peer-to-peer communication: Each process has the capability to communicate with all other processes.
- Each process has a unique ID, with the highest ID being associated with the highest priority.
- A process Pi is denoted with priority i.

Bully Algorithm

The Bully Algorithm is one of the approaches designed to solve the coordinator election problem. It operates as follows:

Election Process:

1. When a process Pi initiates communication with the current coordinator, it attempts to elect itself as the new leader if it receives no response from the current coordinator within a specified time (T units).

2. If Pi perceives that the coordinator is absent (no response is received), Pi notifies all processes with higher priorities of an "Election."

3. If no other process responds to the election message, Pi initiates the coordinator code and notifies all processes with lower priorities that "Elected Pi" has occurred.

4. If another process does respond, Pi gracefully steps down, allowing the higher-priority process to take on the role of coordinator. The coordinator change is communicated to all processes with lower priorities.

5. If Pi is not elected as the coordinator, and it receives a message from another process Pj indicating that it has been elected, Pi updates its internal state to acknowledge Pj as the new coordinator.

6. If Pi receives an "election" message from Pj (where i < j), Pi responds to Pj to confirm that it is still alive. Pi then initiates a vote and waits for the new coordinator.

The Bully Algorithm ensures that the process with the highest priority, among those aware of the election, assumes the coordinator role. If a higher-priority process comes into play, it will take over as the coordinator, ensuring the highest priority process is always the coordinator in the system.

Ring Algorithm

The Ring Algorithm is another approach for solving the coordinator election problem and operates in a fundamentally different manner. In the Ring Algorithm, processes are arranged in a logical ring, and the coordinator role is passed sequentially from one process to the next in the ring. The process with the highest priority is initially the coordinator.

The election process in the Ring Algorithm proceeds as follows:

1. When an election is initiated, a message is sent along the ring, and each process checks its own priority against the incoming message. If a

process with higher priority than the sender exists, it takes over as the new coordinator.

2. If no process with higher priority is found, the message circulates the ring until it reaches the sender again, and that process assumes the coordinator role.

In the Ring Algorithm, the coordinator changes in a cyclic manner, ensuring that the highest-priority process becomes the coordinator and maintains this role until another higher-priority process enters the system.

1. Bully Algorithm:

Code:

```
import time

class InventoryManagement:
    def __init__(self):
        self.inventory = {}

    def add_product(self, id, product, quantity):
        if id in self.inventory:
            self.inventory[id]['quantity'] += quantity
        else:
            self.inventory[id] = {'product': product, 'quantity':
quantity}
        print(f"\nAdded {quantity} {product} to the inventory.")

    def subtract_product(self, id, quantity):
        if id in self.inventory and self.inventory[id]['quantity'] >=
quantity:
            self.inventory[id]['quantity'] -= quantity
            print(f"\nSubtracted {quantity}
{self.inventory[id]['product']} from the inventory.")
        else:
            print(f"Unable to subtract {quantity} {id} from the
inventory.")
```

```
    def view_products(self):
        if not self.inventory:
            print("The inventory is empty.")
        else:
            print("ID\tName\tQuantity")
            for id in self.inventory:

print(f"{id}\t{self.inventory[id]['product']}\t{self.inventory[id]['quanti
ty']}\n")


class BullyElectionAlgorithm:
    def __init__(self, process_id, all_processes):
        self.process_id = process_id
        self.coordinator = max(all_processes)
        self.all_processes = all_processes
        self.election_in_progress = False

    def send_election_message(self, target_process_id):
        print(f"Process {self.process_id} sends election message to
Process {target_process_id}")

    def send_coordinator_message(self, target_process_id):
        print(f"Process {self.process_id} sends coordinator message to
Process {target_process_id}")

    def initiate_election(self):
        if not self.election_in_progress:
            self.election_in_progress = True
            for process in self.all_processes:
                if process > self.process_id:
                    self.send_election_message(process)

    def handle_election_message(self, sender_process_id):
        if sender_process_id > self.process_id:
            print(f"Process {self.process_id} acknowledges Process
{sender_process_id}")
            self.initiate_election()
        else:
            self.send_coordinator_message(sender_process_id)
```

```
            self.coordinator = sender_process_id
            print(f"Process {self.process_id} becomes the coordinator.")
            self.election_in_progress = False

    def election_result(self):
        return self.coordinator

if __name__ == "__main__":
    all_processes = [1, 2, 3, 4, 5]
    default_coordinator = max(all_processes)
    print("\n----------------------------------------------------\n")
    print("Process 1 started")
    print("Process 2 started")
    print("Process 3 started")
    print("Process 4 started")
    print("Process 5 started")
    print("\n----------------------------------------------------\n")
    print(f"Process {default_coordinator} chosen as Coordinator")
    election_algorithm = BullyElectionAlgorithm(default_coordinator,
all_processes)
    inventory_management = InventoryManagement()

while True:
    print("\n----------------------------------------------------\n")
    print("1. Get Election Result")
    print("2. Simulate Leader Failure and Election")
    print("3. Add Product")
    print("4. Subtract Product")
    print("5. View All Products")
    print("6. Quit\n")
    choice = int(input("Enter your choice: "))
    print("----------------------------------------------------")
    if choice == 1:
        current_coordinator = election_algorithm.election_result()
        print(f"Current Coordinator: Process {current_coordinator}")
    elif choice == 2:
        current_coordinator = election_algorithm.election_result()
        print(f"Current Coordinator: Process {current_coordinator}")
        leader_failure = int(input("Enter the process ID to simulate
leader failure:"))
```

```
        print(f"Process {current_coordinator} has failed.")
        for i in range(leader_failure+1, current_coordinator+1):
            print(f"Process {leader_failure} sends election message to
Process {i}")
        newProcessList = []
        for i in range(leader_failure+1, current_coordinator):
            print(f"Process {i} acknowledges Process {leader_failure}")
            newProcessList.append(i)
        new_coordinator = max(newProcessList)
        for i in range(current_coordinator+1):
            if i != new_coordinator:
                print(f"Process {new_coordinator} sends co-ordinator
message to Process {i}")
                election_algorithm.coordinator = new_coordinator
                print("New leader elected")
    elif choice == 3:
        id = input("Enter the product id: ")
        product = input("Enter the product name: ")
        quantity = int(input("Enter the quantity: "))
        inventory_management.add_product(id, product, quantity)
    elif choice == 4:
        id = input("Enter the product id: ")
        quantity = int(input("Enter the quantity: "))
        inventory_management.subtract_product(id, quantity)
    elif choice == 5:
        inventory_management.view_products()
    elif choice == 6:
        break
    else:
        print("Invalid choice. Please select a valid option.")
```

Output

```
------------------------------------------------

Process 1 started
Process 2 started
Process 3 started
Process 4 started
Process 5 started

------------------------------------------------

Process 5 chosen as Coordinator

------------------------------------------------

1. Get Election Result
2. Simulate Leader Failure and Election
3. Add Product
4. Subtract Product
5. View All Products
6. Quit

Enter your choice: 3
------------------------------------------------
Enter the product id: 1
Enter the product name: Pen
Enter the quantity: 20

Added 20 Pen to the inventory.

------------------------------------------------

1. Get Election Result
2. Simulate Leader Failure and Election
3. Add Product
4. Subtract Product
5. View All Products
6. Quit

Enter your choice: 3
------------------------------------------------
Enter the product id: 2
Enter the product name: Cake
Enter the quantity: 10

Added 10 Cake to the inventory.

------------------------------------------------
```

```
------------------------------------------------

1. Get Election Result
2. Simulate Leader Failure and Election
3. Add Product
4. Subtract Product
5. View All Products
6. Quit

Enter your choice: 4
------------------------------------------------
Enter the product id: 1
Enter the quantity: 5

Subtracted 5 Pen from the inventory.

------------------------------------------------

1. Get Election Result
2. Simulate Leader Failure and Election
3. Add Product
4. Subtract Product
5. View All Products
6. Quit

Enter your choice: 5
------------------------------------------------
ID      Name    Quantity
1       Pen     15

2       Cake    10


------------------------------------------------

1. Get Election Result
2. Simulate Leader Failure and Election
3. Add Product
4. Subtract Product
5. View All Products
6. Quit

Enter your choice: 1
------------------------------------------------
Current Coordinator: Process 5

------------------------------------------------
```

```
------------------------------------------------

1. Get Election Result
2. Simulate Leader Failure and Election
3. Add Product
4. Subtract Product
5. View All Products
6. Quit

Enter your choice: 2
------------------------------------------------
Current Coordinator: Process 5
Enter the process ID to simulate leader failure:3
Process 5 has failed.
Process 3 sends election message to Process 4
Process 3 sends election message to Process 5
Process 4 acknowledges Process 3
Process 4 sends co-ordinator message to Process 0
New leader elected
Process 4 sends co-ordinator message to Process 1
New leader elected
Process 4 sends co-ordinator message to Process 2
New leader elected
Process 4 sends co-ordinator message to Process 3
New leader elected
Process 4 sends co-ordinator message to Process 5
New leader elected

------------------------------------------------

1. Get Election Result
2. Simulate Leader Failure and Election
3. Add Product
4. Subtract Product
5. View All Products
6. Quit

Enter your choice: 1
------------------------------------------------
Current Coordinator: Process 4

------------------------------------------------

1. Get Election Result
2. Simulate Leader Failure and Election
3. Add Product
4. Subtract Product
5. View All Products
6. Quit

Enter your choice: 6
------------------------------------------------
```

2. Ring Algorithm:

Code:

```
class InventoryManagement:
    def __init__(self):
        self.inventory = {}

    def add_product(self, id, product, quantity):
        if id in self.inventory:
            self.inventory[id]['quantity'] += quantity
        else:
            self.inventory[id] = {'product': product, 'quantity':
quantity}
        print(f"\nAdded {quantity} {product} to the inventory.")

    def subtract_product(self, id, quantity):
        if id in self.inventory and self.inventory[id]['quantity'] >=
quantity:
            self.inventory[id]['quantity'] -= quantity
            print(f"\nSubtracted {quantity}
{self.inventory[id]['product']} from the inventory.")
        else:
            print(f"Unable to subtract {quantity} {id} from the
inventory.")

    def view_products(self):
        if not self.inventory:
            print("The inventory is empty.")
        else:
            print("ID\tName\tQuantity")
            for id in self.inventory:

print(f"{id}\t{self.inventory[id]['product']}\t{self.inventory[id]['quanti
ty']}\n")


class RingElectionAlgorithm:
    def __init__(self, process_id, all_processes):
        self.process_id = process_id
```

```
        self.coordinator = max(all_processes)
        self.all_processes = all_processes
        self.election_in_progress = False
        self.next_process_id = (process_id + 1) % len(all_processes)

    def send_election_message(self):
        print(f"Process {self.process_id} sends election message to
Process {self.next_process_id}")

    def receive_election_message(self, sender_process_id):
        if sender_process_id > self.process_id:
            print(f"Process {self.process_id} acknowledges Process
{sender_process_id}")
            self.coordinator = sender_process_id
            self.election_in_progress = False
            print(f"Process {self.process_id} becomes the coordinator.")
        else:
            print(f"Process {self.process_id} forwards election message to
Process {self.next_process_id}")
            self.send_election_message()
            self.election_in_progress = True

    def initiate_election(self):
        if not self.election_in_progress:
            self.election_in_progress = True
            self.send_election_message()

    def election_result(self):
        return self.coordinator

if __name__ == "__main__":
    all_processes = [1, 2, 3, 4, 5]
    default_coordinator = max(all_processes)
    print("\n-------------------------------------------------\n")
    print("Process 1 started")
    print("Process 2 started")
    print("Process 3 started")
    print("Process 4 started")
    print("Process 5 started")
    print("\n-------------------------------------------------\n")
```

```
    print(f"Process {default_coordinator} chosen as Coordinator")
    election_algorithm = RingElectionAlgorithm(default_coordinator,
all_processes)
    inventory_management = InventoryManagement()

def ring_algorithm(election_algorithm, leader_failure, all_processes):
    current_coordinator = election_algorithm.election_result()
    if current_coordinator==leader_failure:
        print("\n-------------------------------------------------\n")
        print("Same process ID of leader entered")
        print("\n-------------------------------------------------\n")
        return
    print(f"Current Coordinator: Process {current_coordinator}")

    print(f"Process {current_coordinator} has failed.")
    coordinator_list = []
    j=leader_failure
    for i in range(leader_failure+1,current_coordinator):
        print(f"Message sent from {j} to {i}")
        coordinator_list.append(j)
        j=i
    # if j<all_processes[len(all_processes)-1]:
    #     coordinator_list.append(j)
    if j<all_processes[len(all_processes)-1]:
        for i in range(current_coordinator+1,len(all_processes)):
            print(f"Message sent from {j} to {i}")
            coordinator_list.append(j)
            j=i

    if all_processes[0]==leader_failure:
        coordinator_list.append(j)
        print(f"Message sent from {j} to {leader_failure}")
    else:
        for i in range(all_processes[0],leader_failure+1):
            print(f"Message sent from {j} to {i}")
            coordinator_list.append(j)
            j=i
    print(f"Candidates for leader : ",coordinator_list)
    new_coo = max(coordinator_list)
    print("New leader elected")
```

```
    print(f"Process {new_coo} elected as new leader.")

    for i in range(all_processes[0], all_processes[len(all_processes)-1]):
        if i != leader_failure and i != current_coordinator:
            print(f"{leader_failure} send message to {i}, New leader
elected")

    election_algorithm.coordinator = new_coo

while True:
    print("\n----------------------------------------------------\n")
    print("1. Get Election Result")
    print("2. Simulate Leader Failure and Election")
    print("3. Add Product")
    print("4. Subtract Product")
    print("5. View All Products")
    print("6. Quit\n")

    choice = int(input("Enter your choice: "))
    print("----------------------------------------------")

    if choice == 1:
        current_coordinator = election_algorithm.election_result()
        print(f"Current Coordinator: Process {current_coordinator}")

    elif choice == 2:
        leader_failure = int(input("Enter the process ID to simulate
leader failure:"))
        ring_algorithm(election_algorithm, leader_failure, all_processes)

    elif choice == 3:
        id = input("Enter the product id: ")
        product = input("Enter the product name: ")
        quantity = int(input("Enter the quantity: "))

        inventory_management.add_product(id, product, quantity)

    elif choice == 4:
        id = input("Enter the product id: ")
        quantity = int(input("Enter the quantity: "))
```

```
        inventory_management.subtract_product(id, quantity)

    elif choice == 5:
        inventory_management.view_products()

    elif choice == 6:
        break

    else:
        print("Invalid choice. Please select a valid option.")
```

Output:

```
------------------------------------------------

Process 1 started
Process 2 started
Process 3 started
Process 4 started
Process 5 started

------------------------------------------------

Process 5 chosen as Coordinator

------------------------------------------------

1. Get Election Result
2. Simulate Leader Failure and Election
3. Add Product
4. Subtract Product
5. View All Products
6. Quit

Enter your choice: 3
------------------------------------------------
Enter the product id: 1
Enter the product name: Biscuit
Enter the quantity: 20

Added 20 Biscuit to the inventory.

------------------------------------------------

1. Get Election Result
2. Simulate Leader Failure and Election
3. Add Product
4. Subtract Product
5. View All Products
6. Quit

Enter your choice: 3
------------------------------------------------
Enter the product id: 2
Enter the product name: Cake
Enter the quantity: 10

Added 10 Cake to the inventory.

------------------------------------------------
```

```
------------------------------------------------

1. Get Election Result
2. Simulate Leader Failure and Election
3. Add Product
4. Subtract Product
5. View All Products
6. Quit

Enter your choice: 4
------------------------------------------------
Enter the product id: 1
Enter the quantity: 5

Subtracted 5 Biscuit from the inventory.

------------------------------------------------

1. Get Election Result
2. Simulate Leader Failure and Election
3. Add Product
4. Subtract Product
5. View All Products
6. Quit

Enter your choice: 5
------------------------------------------------
ID      Name    Quantity
1       Biscuit 15

2       Cake    10


------------------------------------------------

1. Get Election Result
2. Simulate Leader Failure and Election
3. Add Product
4. Subtract Product
5. View All Products
6. Quit

Enter your choice: 1
------------------------------------------------
Current Coordinator: Process 5

------------------------------------------------
```

```
------------------------------------------------

1. Get Election Result
2. Simulate Leader Failure and Election
3. Add Product
4. Subtract Product
5. View All Products
6. Quit

Enter your choice: 2
------------------------------------------------
Enter the process ID to simulate leader failure:2
Current Coordinator: Process 5
Process 5 has failed.
Message sent from 2 to 3
Message sent from 3 to 4
Message sent from 4 to 1
Message sent from 1 to 2
Candidates for leader :  [2, 3, 4, 1]
New leader elected
Process 4 elected as new leader.
2 send message to 1, New leader elected
2 send message to 3, New leader elected
2 send message to 4, New leader elected

------------------------------------------------

1. Get Election Result
2. Simulate Leader Failure and Election
3. Add Product
4. Subtract Product
5. View All Products
6. Quit

Enter your choice: 1
------------------------------------------------
Current Coordinator: Process 4

------------------------------------------------

1. Get Election Result
2. Simulate Leader Failure and Election
3. Add Product
4. Subtract Product
5. View All Products
6. Quit

Enter your choice: 6
------------------------------------------------
```

Conclusion:

In summary, our study has provided valuable insights into the significance of election algorithms in the context of distributed systems. Through our examination of the Bully Algorithm and the Ring Algorithm, we've gained an understanding of how these algorithms enable processes to independently select a coordinator in a distributed environment. As demonstrated, these algorithms guarantee the presence of a sole coordinator at all times, ultimately improving the efficiency of operational management in the distributed system.