Department of Computer Science & Engineering, Jorhat Engineering College

Distributed Systems Assignment

Submitted By
Kaushik Barkataki
Roll No: 210710007023
Batch: 2021-2025

Problem: *Implement Lamport's Logical Clock using Python*

Literature Review:

Lamport's Logical Clock, introduced by Leslie Lamport, orders events in distributed systems without a global clock. It ensures consistent sequencing of events across processes and forms the basis of advanced algorithms like Vector Clocks.

Algorithm:

- Happened-before (→): If a → b, then $C_i(a) < C_i(b)$.
- Logical Clock Rules:
    - IR1: For same process events, $C_i(b) = C_i(a) + d$.
    - IR2: On message receipt, $C_j = \max(C_j, t_m) + d$.

Explanation of the Algorithm:

1. Each process maintains its own logical clock, initialized to zero.
2. For an internal event, the process increments its clock by 1.
3. When a message is sent, the current clock value is included.
4. On receiving a message, the process updates its clock to `max(current clock, received clock) + 1`.
5. This ensures the logical order of events is preserved across all processes.

Implementation in Python:

```
class LamportClock:
    def __init__(self, process_id):
        self.process_id = process_id
        self.clock = 0

    def increment(self):
        """Increment the clock for an internal event."""
        self.clock += 1
            print(f"Process {self.process_id}: Clock incremented to
{self.clock}")
```

```
    def send(self):
        """Simulate sending a message, increasing the clock value."""
        self.clock += 1
          print(f"Process {self.process_id}: Sent message with clock
{self.clock}")
        return self.clock

    def receive(self, received_clock):
        """
        Update the clock upon receiving a message.
        The clock becomes the maximum of the current clock or received
clock + 1.
        """
        self.clock = max(self.clock, received_clock) + 1
        print(f"Process {self.process_id}: Received message and updated
clock to {self.clock}")

# Example Usage
if __name__ == "__main__":
    # Instantiate clocks for two processes
    process_A = LamportClock("A")
    process_B = LamportClock("B")

    # Simulate events in process A
    process_A.increment()       # Internal event in A
    sent_clock = process_A.send()  # A sends a message

    # Simulate events in process B
    process_B.increment()       # Internal event in B
    process_B.receive(sent_clock)  # B receives the message from A

    # Another event in process A
    process_A.increment()

    # Process B sends a message to process A
    sent_clock = process_B.send()
    process_A.receive(sent_clock)  # A receives the message from B
```

Output:

```
Process A: Clock incremented to 1
Process A: Sent message with clock 2
Process B: Clock incremented to 1
Process B: Received message and updated clock to 3
Process A: Clock incremented to 3
Process B: Sent message with clock 4
Process A: Received message and updated clock to 5
```