**BHARATIYA VIDYA BHAVAN'S**
**SARDAR PATEL INSTITUTE OF TECHNOLOGY**
**(**Empowered Autonomous Institute Affiliated to Mumbai University)
**Department Of Computer Engineering**

| | |
|---|---|
| **Name** | Mayur Solankar, Manish Jadhav, Vishesh Savani, Shreyansh Salvi |
| **UID** | 2023301018, 2023301005, 2022300100, 2022300091 |
| **Subject** | Distributed Computing |
| **Experiment No.** | 7 |
| **Project title** | Social Media System |
| **Problem Statement** | To implement clock synchronization using Lamport's Logical Clock Algorithm |
| **Objectives** | The objective of Lamport's logical clock algorithm in a social media platform is to provide a consistent and ordered timestamping mechanism for posts and comments, ensuring accurate sequencing in a distributed environment. |
| **Theory** | **What are Lamport clocks ?**<br>Lamport clocks represent time logically in a distributed system. They are also known as logical clocks. The idea behind Lamport clocks is to disregard physical time and capture just a "happens-before" relationship between a pair of events.<br><br>**Why use Lamport clocks ?**<br>Time synchronization is a key problem in distributed systems. Time is used to order events across servers. Using physical clocks to order events is challenging because real synchronization is impossible and clocks experience skew. A clock skew is when different clocks run at different rates, so we cannot assume that time t on node a happened before time t + 1 on node b.<br>Instead of employing physical time, Leslie Lamport proposed logical clocks that capture events' orderings through a "happens-before" relationship.<br>Implementing Lamport's logical clock algorithm in a social media platform is necessary for maintaining order of events in a distributed environment. In the context of a social media platform, users constantly create posts and comments. Lamport's logical clocks allow these events to be time stamped consistently, ensuring that the order in which they occurred is accurate.<br>When a user makes a post or a comment, Lamport's algorithm assigns a logical timestamp to the event, representing its position in the sequence of events across the platform. This logical timestamp confirms that events are related in the correct order, regardless of the physical time they occurred. |

**CODE:**

**//SERVER SIDE**

```
import socket

class LamportClock:
    def _init_(self):
        self.timestamp = 0

    def tick(self):
        self.timestamp += 1

    def update(self, received_time):
        self.timestamp = max(self.timestamp, received_time) + 1

class Server:
    def _init_(self):
        self.clock = LamportClock()

    def handle_request(self, data):
        # Process the request and assign a Lamport timestamp to the event.
        self.clock.tick()
        event = f"{self.clock.timestamp}:{data}"
        return event

def main():
    server = Server()
    host = '127.0.0.1'
    port = 12345

    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind((host, port))
        s.listen()

        while True:
            conn, addr = s.accept()
            with conn:
                data = conn.recv(1024).decode('utf-8')
                if data:
                    event = server.handle_request(data)
                    print(f"Received Event: {event}")
```

```
                conn.send(event.encode('utf-8'))

if _name_ == "_main_":
    main()
```

**//CLIENT SIDE**

```
client
import socket
import time

class Client:
    def _init_(self):
        self.clock = 0
        self.host = '127.0.0.1'
        self.port = 12345

    def send_event(self, event):
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.connect((self.host, self.port))
            s.send(event.encode('utf-8'))
            data = s.recv(1024).decode('utf-8')
            print(f"Timestamp: {data}")

    def create_user(self, username):
        event = f"CREATE_USER:{username}"
        self.send_event(event)

    def post(self, username, content):
        event = f"POST:{username}:{content}"
        self.send_event(event)

    def comment(self, username, post_id, content):
        event = f"COMMENT:{username}:{post_id}:{content}"
        self.send_event(event)

def main():
    client = Client()

    # Simulate user actions
```

```
    user1 = input("\nEnter the username: ")
    client.create_user(user1)

    time.sleep(1)

    post = input("\nCreate a post: ")
    client.post(user1, post)

    time.sleep(1)

    print("\n\nYou are User2 and you want to comment on the post")
    comment = input("Your Comment: ")
    client.comment(user1, 1, comment)

if _name_ == "_main_":
    main()
```

**OUTPUT:**

```
PS C:\Users\vishe\OneDrive\Desktop\DC Codes> cd .\7_Experiment\
PS C:\Users\vishe\OneDrive\Desktop\DC Codes\7_Experiment> python server.py
Received Event: 1:CREATE_USER:vishesh
Received Event: 2:POST:vishesh:Hi guys ! I implemented clock synchronizationn algorithm
Received Event: 3:COMMENT:vishesh:1:congrats vishesh!
```

```
PS C:\Users\vishe\OneDrive\Desktop\DC Codes> cd .\7_Experiment\
PS C:\Users\vishe\OneDrive\Desktop\DC Codes\7_Experiment> python client.py

Enter the username: vishesh
Timestamp: 1:CREATE_USER:vishesh

Create a post: Hi guys ! I implemented clock synchronizationn algorithm
Timestamp: 2:POST:vishesh:Hi guys ! I implemented clock synchronizationn algorithm


You are User2 and you want to comment on the post
Your Comment: congrats vishesh!
Timestamp: 3:COMMENT:vishesh:1:congrats vishesh!
PS C:\Users\vishe\OneDrive\Desktop\DC Codes\7_Experiment>
```

| | |
|---|---|
| **Conclusion:** | Hence by completing we came to about implementation of clock synchronization using Lamport's Logical Clock Algorithm |