St. Francis Institute of Technology

Department of Computer Engineering

Academic Year: 2021-2022 **Semester: VIII**

Subject: DCL **Class/Branch/: BE/CMPNA**

Name :- Nithin Menezes **Roll Number: 56**

EXPERIMENT-4

Aim: To implement Clock Synchronization using Lamport Timestamp Algorithm

Theory:

- **What is clock synchronization?**
  Clock synchronization is a method of synchronizing clock values of any two nodes in a distributed system with the use of external reference clock or internal clock value of the node. During the synchronization, many factors affect on a network. Even when initially set accurately, real clocks will differ after some amount of time due to clock drift, caused by clocks counting time at slightly different rates.

- **Types of Clock Synchronization**
  The clock synchronization can be achieved by 2 ways: External and Internal Clock Synchronization. ○ **External clock synchronization** is the one in which an external reference clock is present. It is used as a reference and the nodes in the system can set and adjust their time accordingly. ○ **Internal clock synchronization** is the one in which each node shares its time with other nodes and all the nodes set and adjust their times accordingly.

- **Types of Clock Synchronization Algorithms**

  Physical clock synchronization algorithm

  Every computer contains a clock which is an electronic device that counts the oscillations in a crystal at a particular frequency. Synchronization of these physical clocks to some known high degree of accuracy is needed. This helps to measure the time relative to each local clock to determine order between events.

  Physical clock synchronization algorithms can be classified as centralized and distributed.

1. **Centralized clock synchronization algorithms**

   These have one node with a real-time receiver and are called time server nodes. The clock time of this node is regarded as correct and used as reference time.

   The goal of this algorithm is to keep the clocks of all other nodes synchronized with the time server node.

   Examples of Centralized clock synchronization algorithms:

   a. Cristian's Algorithm

   b. The Berkeley Algorithm

2. **Distributed algorithms**

   Distributed algorithms overcome the problems of centralized by internally synchronizing for better accuracy. One of the two approaches can be used:

   a. Global Averaging Distributed Algorithms

   b. Localized Averaging Distributed Algorithms

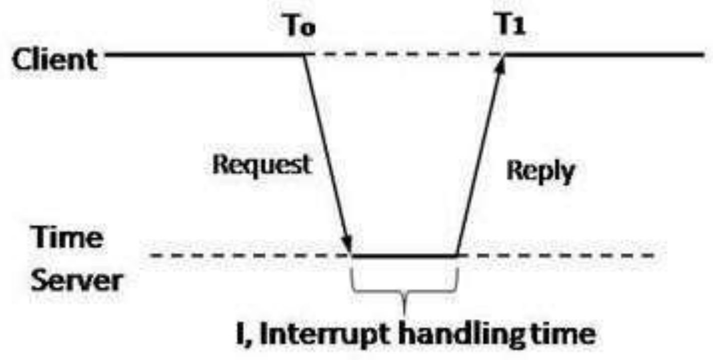
Logical clock synchronization algorithm

Logical Clocks refer to implementing a protocol on all machines within your distributed system, so that the machines are able to maintain consistent ordering of events within some virtual timespan. A logical clock is a mechanism for capturing chronological and causal relationships in a distributed system. Distributed systems may have no physically synchronous global clock, so a logical clock allows global ordering on events from different processes in such systems. Examples of this are:

a) Lamport Timestamp Algorithm
b) Vector Timestamp Algorithm

Cristian Algorithm

Cristian's Algorithm is a clock synchronization algorithm is used to synchronize time with a time server by client processes. This algorithm works well with low-latency networks where Round Trip Time is short as compared to accuracy while redundancy-prone distributed systems/applications do not go hand in hand with this algorithm.



- In this method each node periodically sends a message to the server. When the time server receives the message it responds with a message T, where T is the current time of the server node.
- Assume the clock time of client be To when it sends the message and T1 when it receives the message from server. To and T1 are measured using the same clock so the best estimate of time for propagation is (T1-To)/2.
- When the reply is received at the client's node, its clock is readjusted to T+(T1-T0)/2. There can be unpredictable variation in the message propagation time between the nodes hence (T1-T0)/2 is not good to be added to T for calculating current time.
- For this several measurements of T1-To are made and if these measurements exceed some threshold value then they are unreliable and discarded. The average of the remaining measurements is calculated and the minimum value is considered accurate and half of the calculated value is added to T.
- Advantage-It assumes that no additional information is available.
- Disadvantage- It restricts the number of measurements for estimating the value.

Logical Clock Synchronization Algorithm:

- **Lamport Timestramp Algoritm:**

The Lamport timestamp algorithm is a simple logical clock algorithm used to determine the order of events in a distributed computer system. As different nodes or processes will typically not be perfectly synchronized, this algorithm is used to provide a partial ordering of events with minimal overhead, and conceptually provide a starting point for the more advanced vector clock method. The algorithm is named after its creator, Leslie Lamport.

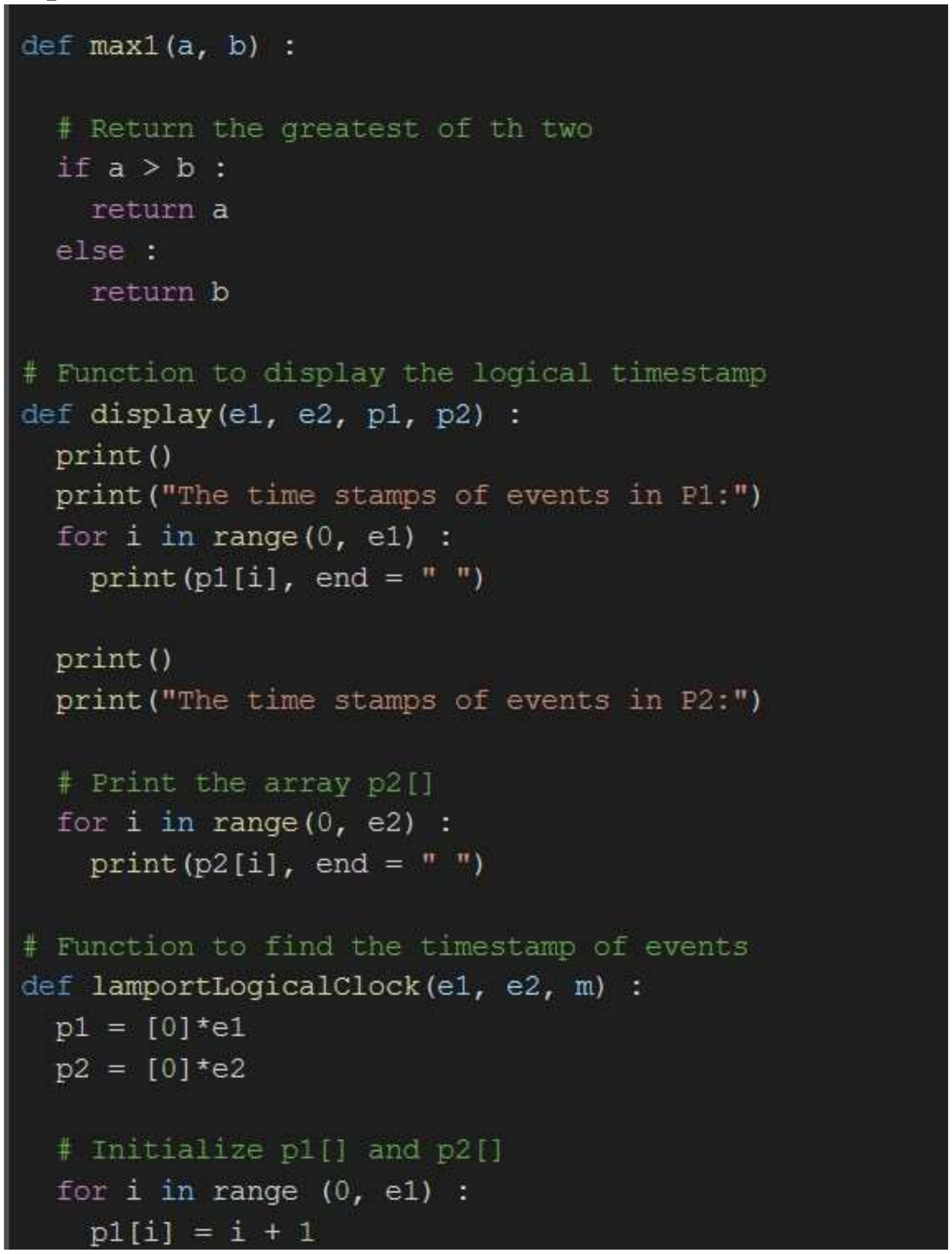
The algorithm follows some simple rules:

1. A process increments its counter before each local event (e.g., message sending event);

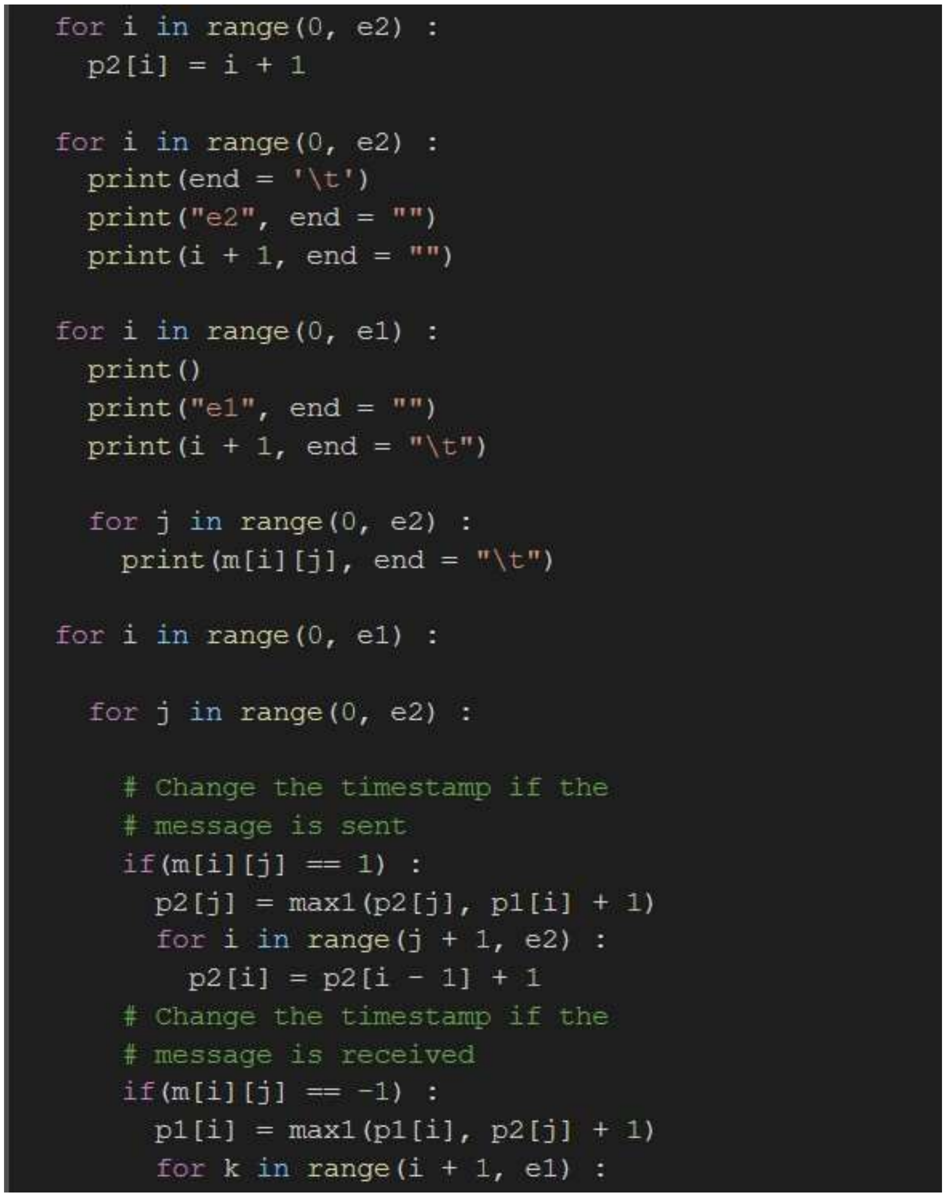
2. When a process sends a message, it includes its counter value with the message after executing step 1;

3. On receiving a message, the counter of the recipient is updated, if necessary, to the greater of its current counter and the timestamp in the received message. The counter is then incremented by 1 before the message is considered received.

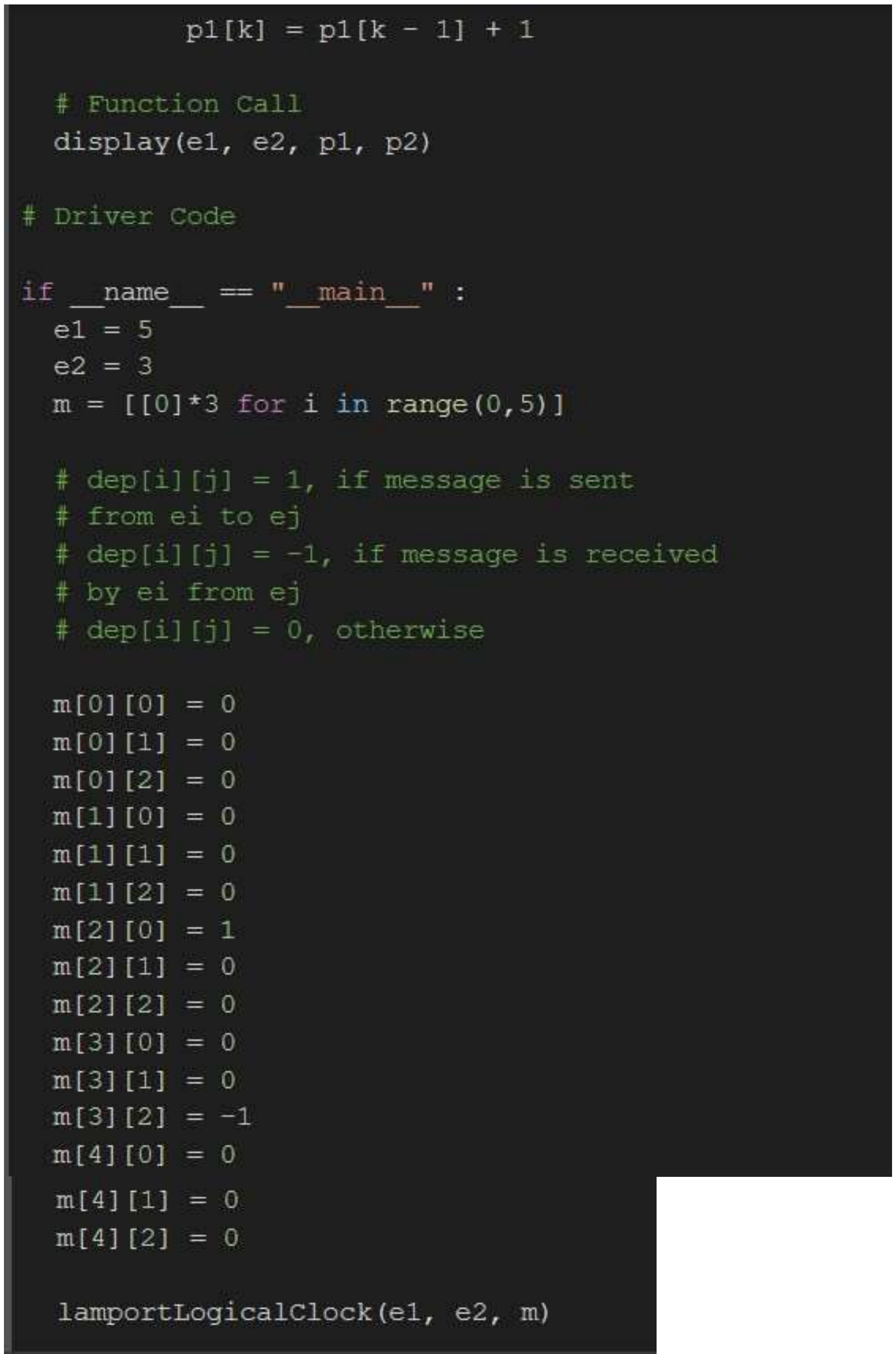
Implementation :



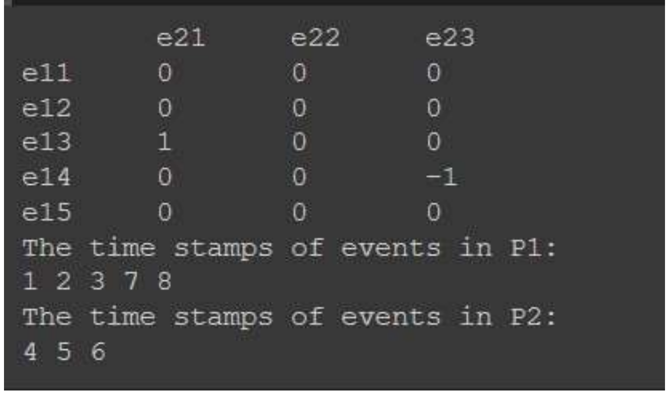
```
def max1(a, b) :

  # Return the greatest of th two
  if a > b :
    return a
  else :
    return b

# Function to display the logical timestamp
def display(e1, e2, p1, p2) :
  print()
  print("The time stamps of events in P1:")
  for i in range(0, e1) :
    print(p1[i], end = " ")

  print()
  print("The time stamps of events in P2:")

  # Print the array p2[]
  for i in range(0, e2) :
    print(p2[i], end = " ")

# Function to find the timestamp of events
def lamportLogicalClock(e1, e2, m) :
  p1 = [0]*e1
  p2 = [0]*e2

  # Initialize p1[] and p2[]
  for i in range (0, e1) :
    p1[i] = i + 1
```



```
for i in range(0, e2) :
  p2[i] = i + 1

for i in range(0, e2) :
  print(end = '\t')
  print("e2", end = "")
  print(i + 1, end = "")

for i in range(0, e1) :
  print()
  print("e1", end = "")
  print(i + 1, end = "\t")

  for j in range(0, e2) :
    print(m[i][j], end = "\t")

for i in range(0, e1) :

  for j in range(0, e2) :

    # Change the timestamp if the
    # message is sent
    if(m[i][j] == 1) :
      p2[j] = max1(p2[j], p1[i] + 1)
      for i in range(j + 1, e2) :
        p2[i] = p2[i - 1] + 1
    # Change the timestamp if the
    # message is received
    if(m[i][j] == -1) :
      p1[i] = max1(p1[i], p2[j] + 1)
      for k in range(i + 1, e1) :
```



```
          p1[k] = p1[k - 1] + 1

  # Function Call
  display(e1, e2, p1, p2)

# Driver Code

if __name__ == "__main__" :
  e1 = 5
  e2 = 3
  m = [[0]*3 for i in range(0,5)]

  # dep[i][j] = 1, if message is sent
  # from ei to ej
  # dep[i][j] = -1, if message is received
  # by ei from ej
  # dep[i][j] = 0, otherwise

  m[0][0] = 0
  m[0][1] = 0
  m[0][2] = 0
  m[1][0] = 0
  m[1][1] = 0
  m[1][2] = 0
  m[2][0] = 1
  m[2][1] = 0
  m[2][2] = 0
  m[3][0] = 0
  m[3][1] = 0
  m[3][2] = -1
  m[4][0] = 0
  m[4][1] = 0
  m[4][2] = 0

  lamportLogicalClock(e1, e2, m)
```



```
        e21     e22     e23
e11     0       0       0
e12     0       0       0
e13     1       0       0
e14     0       0       -1
e15     0       0       0
The time stamps of events in P1:
1 2 3 7 8
The time stamps of events in P2:
4 5 6
```

Conclusion:

In the above experiment we implemented clock synchronization. Clock synchronization is a method of synchronizing clock values of any two nodes in a distributed system with the use of external reference clock or internal clock value of the node. Clock

Synchronization was performed using the Lamport’s Timestamp algorithm. It is an internal clock synchronization algorithm. This algorithm works well with low-latency networks where Round Trip Time is short as compared to accuracy while redundancy-prone, It is a procedure to determine the order of events occurring. It provides a basis for the more advanced Vector Clock Algorithm. Due to the absence of a Global Clock in a Distributed Operating System Lamport Logical Clock is needed.

.