St. Francis Institute of Technology
Department of Computer Engineering

Academic Year: 2021-2022 **Semester: VIII**
Subject: Distributed Computing **Class / Branch / Division: BE/CMPN/A**
Name: Rebecca Dias Roll Number:18

Experiment No: 04

Aim: Implementation of Clock Synchronization algorithm.

Theory:

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 synchronization, many factors affect 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.

Lamport's Algorithm

- 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.
- Each process maintains a single Lamport timestamp counter. Each event on the process is tagged with a timestamp from this counter.
- The counter is incremented before the event timestamp is assigned.
- If a process has four events, a, b, c, d, they would get Lamport timestamps of 1, 2, 3, 4.
- If an event is the sending of a message then the timestamp is sent along with the message.
- If an event is the receipt of a message then the algorithm instructs you to compare the current value of the process' timestamp counter (which was just incremented before this event) with the timestamp in the received message.
- If the timestamp of the received message is greater than that of the current system, the system timestamp is updated with that of the timestamp in the received message plus one.
- This ensures that the timestamp of the received event and all further timestamps will be greater than that of the timestamp of sending the message as well as all previous messages.
- Lamport invented a simple mechanism by which the happened-before ordering can be captured numerically.
- A Lamport logical clock is an incrementing software counter maintained in each process.
- It follows some simple rules:

1. A process increments its counter before each event in that process;

2. When a process sends a message, it includes its counter value with the message;
3. On receiving a message, the receiver process sets its counter to be the maximum of the message counter and its own counter incremented, before it considers the message received.

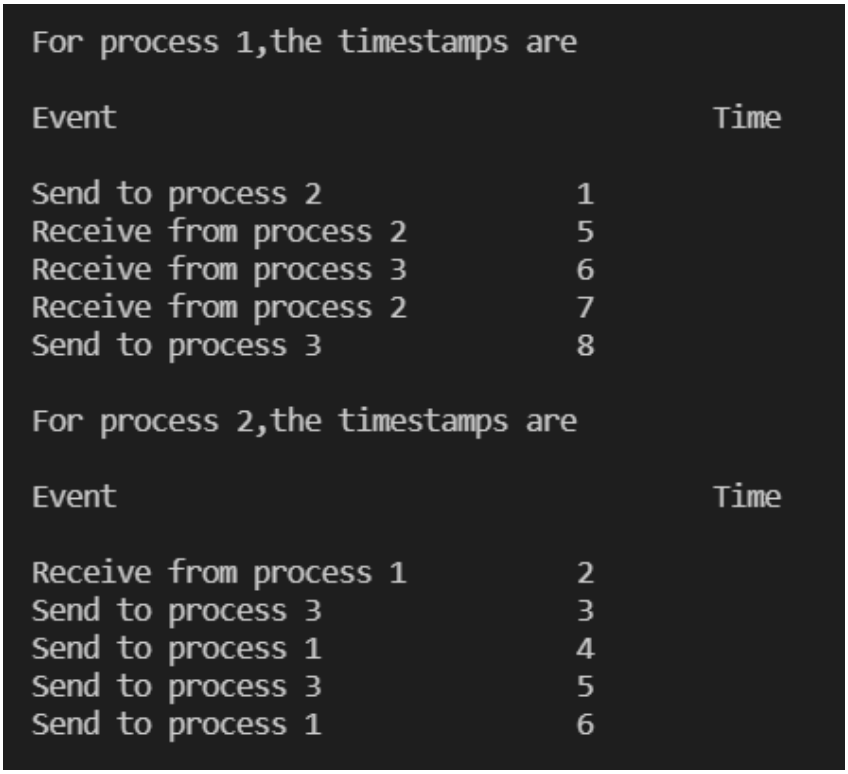
Program

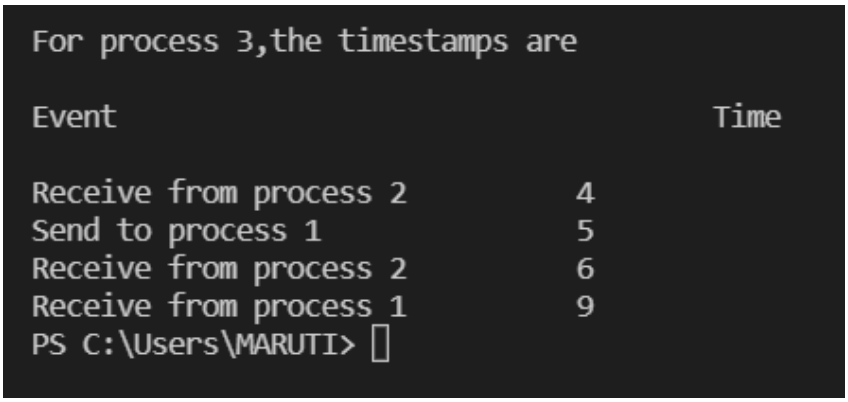
```
import java.io.*;
class Lamport
{
	public static void main(String args[]) throws IOException
	{
		BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
		int p[],n,st,en,tlog[][],cnt[];
		String t[][];
		System.out.println("Enter the number of processes");
		n=Integer.parseInt(br.readLine());
		p=new int[n];
		tlog=new int[n][10];
		cnt=new int[n];
		t=new String[n][10];
		for(int i=0;i<n;i++)
		{
				p[i]=0;
				cnt[i]=0;
		}
		System.out.println("Enter the messages sent(0 0 to exit)");
		while(true)
		{
				System.out.println("From process:");
				st=Integer.parseInt(br.readLine());
				System.out.println("To process:");
				en=Integer.parseInt(br.readLine());
				if(st==0 && en==0)
					break;
				p[st-1]+=1;
				tlog[st-1][cnt[st-1]]=p[st-1];
				t[st-1][cnt[st-1]++]="Send to process "+en;
				p[en-1]=((p[en-1]+1)>(p[st-1]+1))?(p[en-1]+1):(p[st-1]+1);
				tlog[en-1][cnt[en-1]]=p[en-1];
				t[en-1][cnt[en-1]++]="Receive from process "+st;
		}
		for(int i=0;i<n;i++)
		{
				System.out.println("\nFor process "+(i+1)+",the timestamps are\n");
				System.out.println("Event\t\t\t\t\tTime\n");
				for(int j=0;j<cnt[i];j++)
					System.out.println(t[i][j]+"\t\t"+tlog[i][j]);
		}
	}
}
```

Output:

```
Enter the number of processes
3
Enter the messages sent(0 0 to exit)
From process:
1
To process:
2
From process:
2
To process:
3
From process:
2
To process:
1
```

```
From process:
3
To process:
1
From process:
2
To process:
3
From process:
2
To process:
1
From process:
1
To process:
3
From process:
0
To process:
0
```





Conclusion:

In the above experiment we implemented clock synchronization. We also learnt about Lamport Algorithm which is used to implement clock synchronization which is a simple logical clock algorithm used to determine the order of events in a distributed computer system. Thus the experiment was successfully implemented.