# Aim:

Perform Implementation of Clock Synchronization Using Lamports' Algorithm

# Theory:

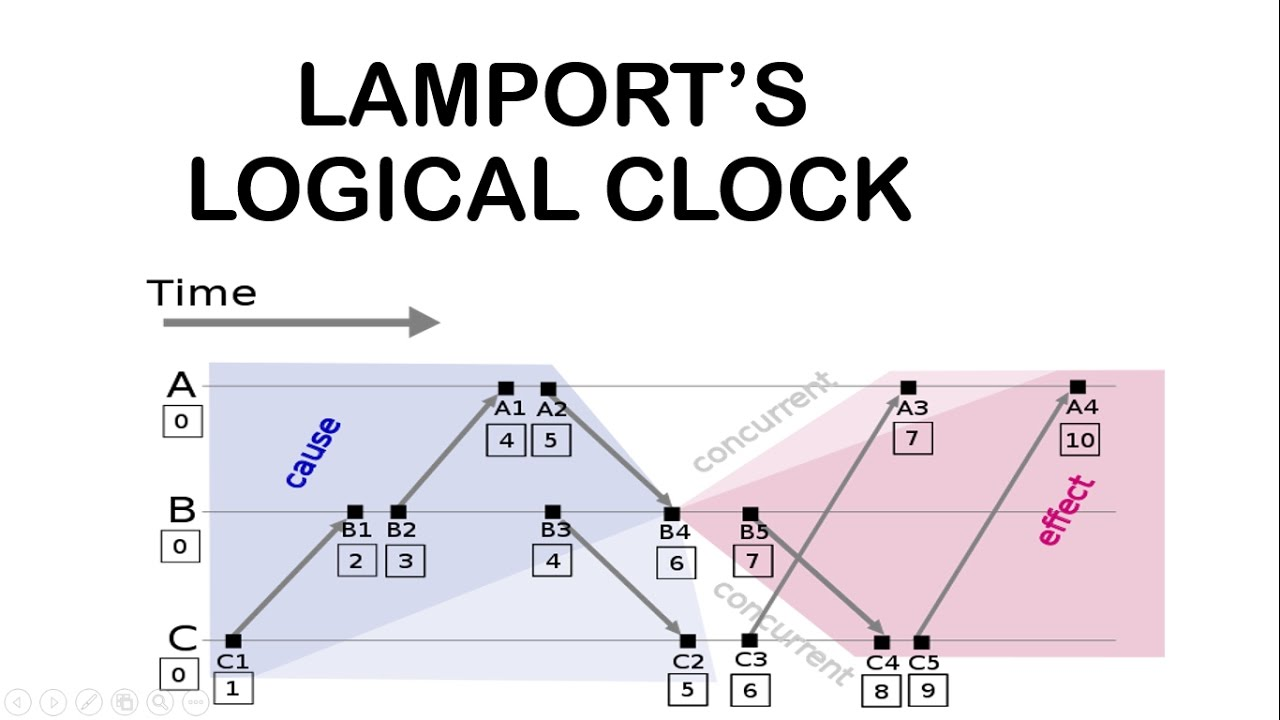
Clock synchronization is a problem from computer science and engineering which deals with the

idea that internal clocks of several computers may differ. 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



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 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;

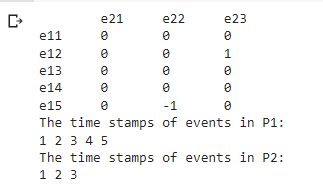
3. On receiving a message, the receiver process sets its counter to be the maximum of the

message counter and its own counter incremented, before it considers the message received.

# Program:

| # Python program to illustrate the Lamport's Logical Clock  # Function to find the maximum timestamp between 2 events  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[k] = p2[k - 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)  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] = 1  m[2][0] = 0  m[2][1] = 0  m[2][2] = 0  m[3][0] = 0  m[3][1] = 0  m[3][2] = 0  m[4][0] = 0  m[4][1] = -1  m[4][2] = 0  # Function Call  lamportLogicalClock(e1, e2, m) |
| --- |

# Output:



# Conclusion:

Thus we have performed the Implementation of Clock Synchronization Using Lamports' Algorithm.