# Synchronization

# Event Ordering

**Happened Before Relation (denoted by --> )**

1. **If a  and b are events in same process, and a occurs before b then a->b**
2. **If a is an event of sending a message by one process and b is the event of receipt of same message by another process,then a->b…receiver cannot receive unless sender sends it.Time taken for the message propogation from receiver to sender is always positive.**

3. **If a -> b and b -> c,then a -> c,thus happened before is a transitive relation.**

# Logical clocks concept

The concept of a logical clock is a way to associate a timestamp (which maybe a simple number independent of any clock time)

Eg : Pi has a clock Ci associated with it that assigns a number Ci(a) to any event a in that process.

# Lamport's Logical Clocks (2)

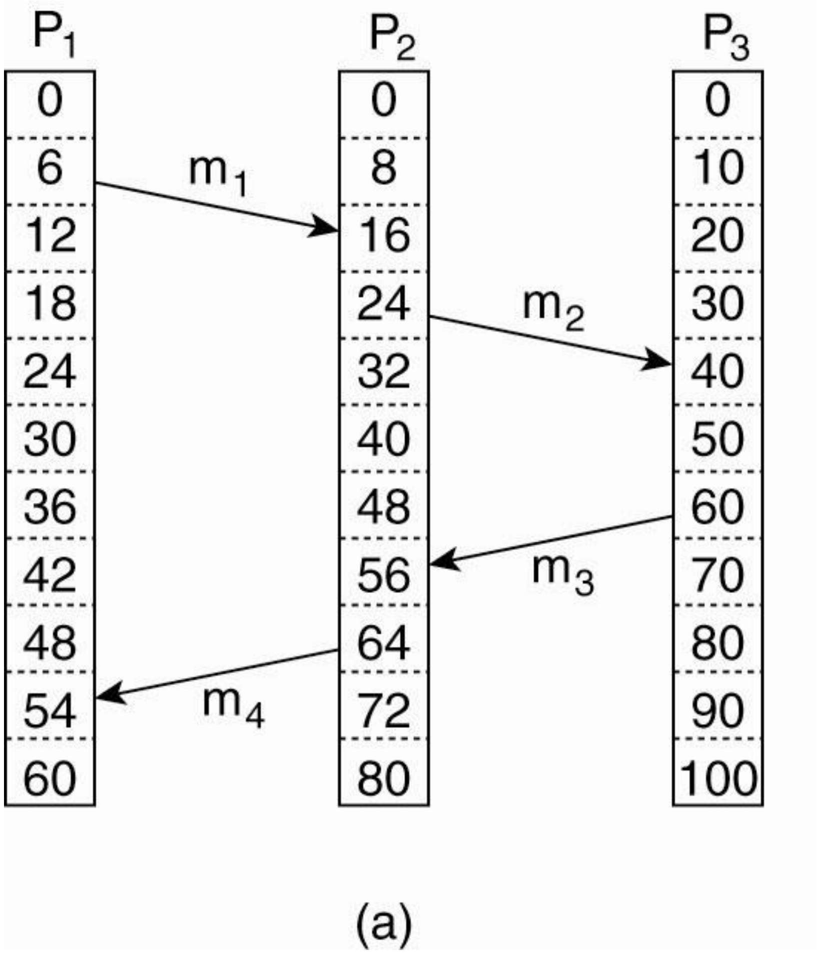

Figure 6-9. (a) Three processes, each with its own clock. The clocks run at different rates.

# Lamport's Logical Clocks (3)

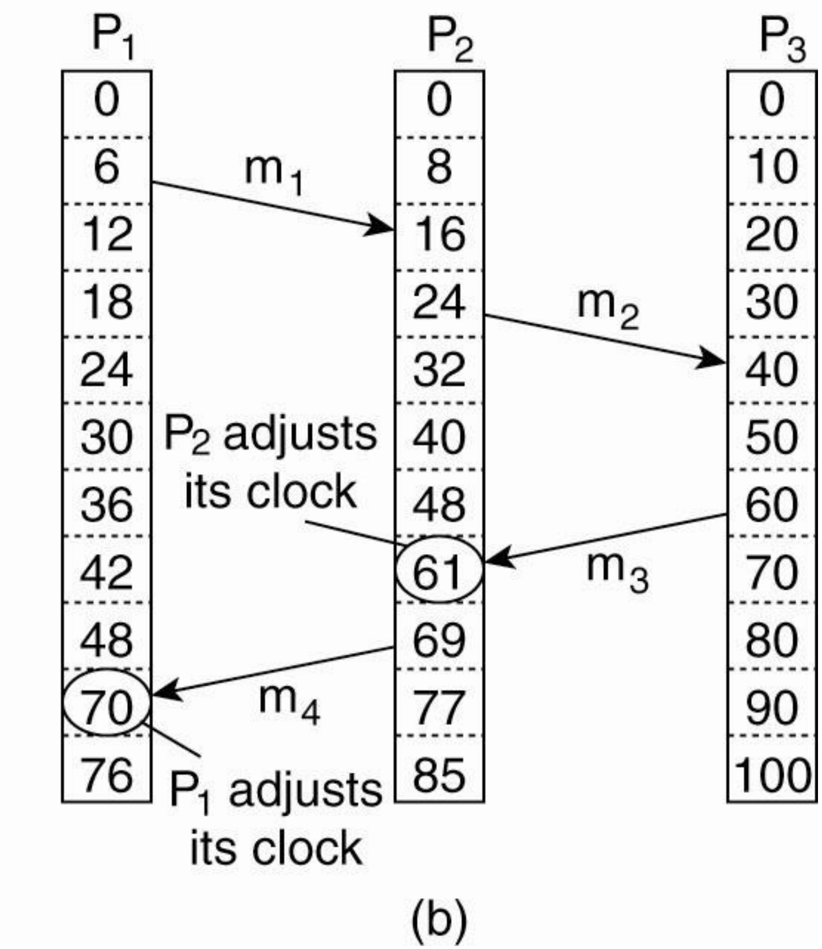

Figure 6-9. (b) Lamport's algorithm corrects the clocks.

# Lamport’s Logical Clocks (4)

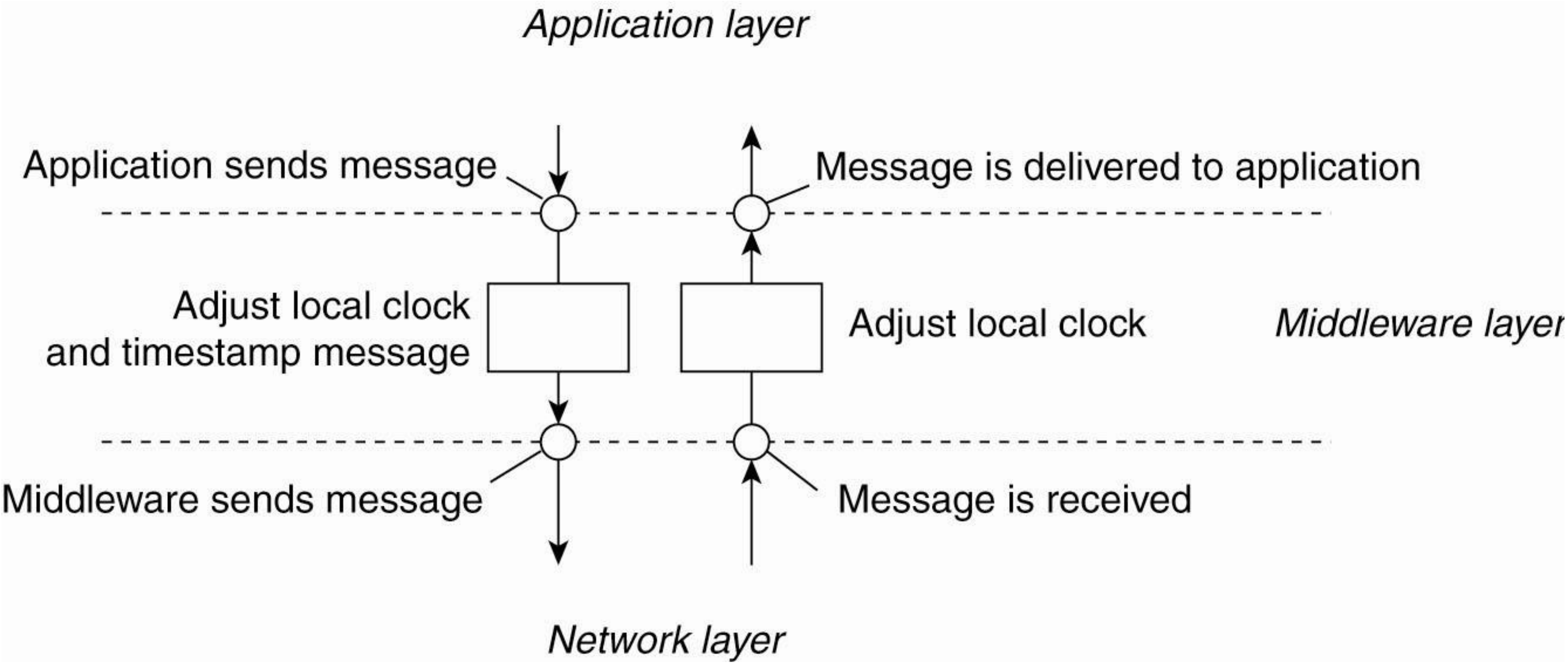

Figure 6-10. The positioning of Lamport’s logical clocks in distributed systems.

# Implementation of logical clocks

C1 : If a and b are two events within the same process Pi and a occurs before b,then Ci(a) <Ci(b)

C2 : If a is the sending of a message by process Pi and b is the receipt of that message by process Pj , then Ci(a) < Cj(b)

C3 : A clock Ci associated with a process Pi must always go forward,never backward.Correctness is made by adding value never subtracting it

# Implementation of logical clocks

IR1 :Each process Pi increments Ci between any two successive events

IR2 :If event a is the sending of a message m by process Pi, the message m contains a timestamp Tm = Ci (a) and upon receiving the message m a process Pj sets Cj greater than or equal to its present value but greater than Tm.