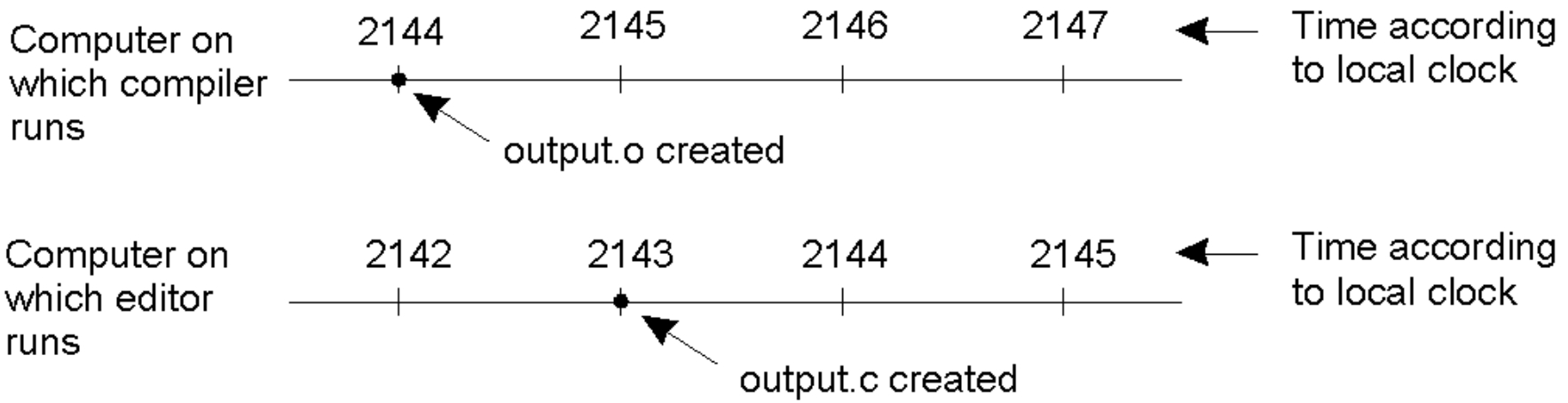
MYE017 Distributed Systems

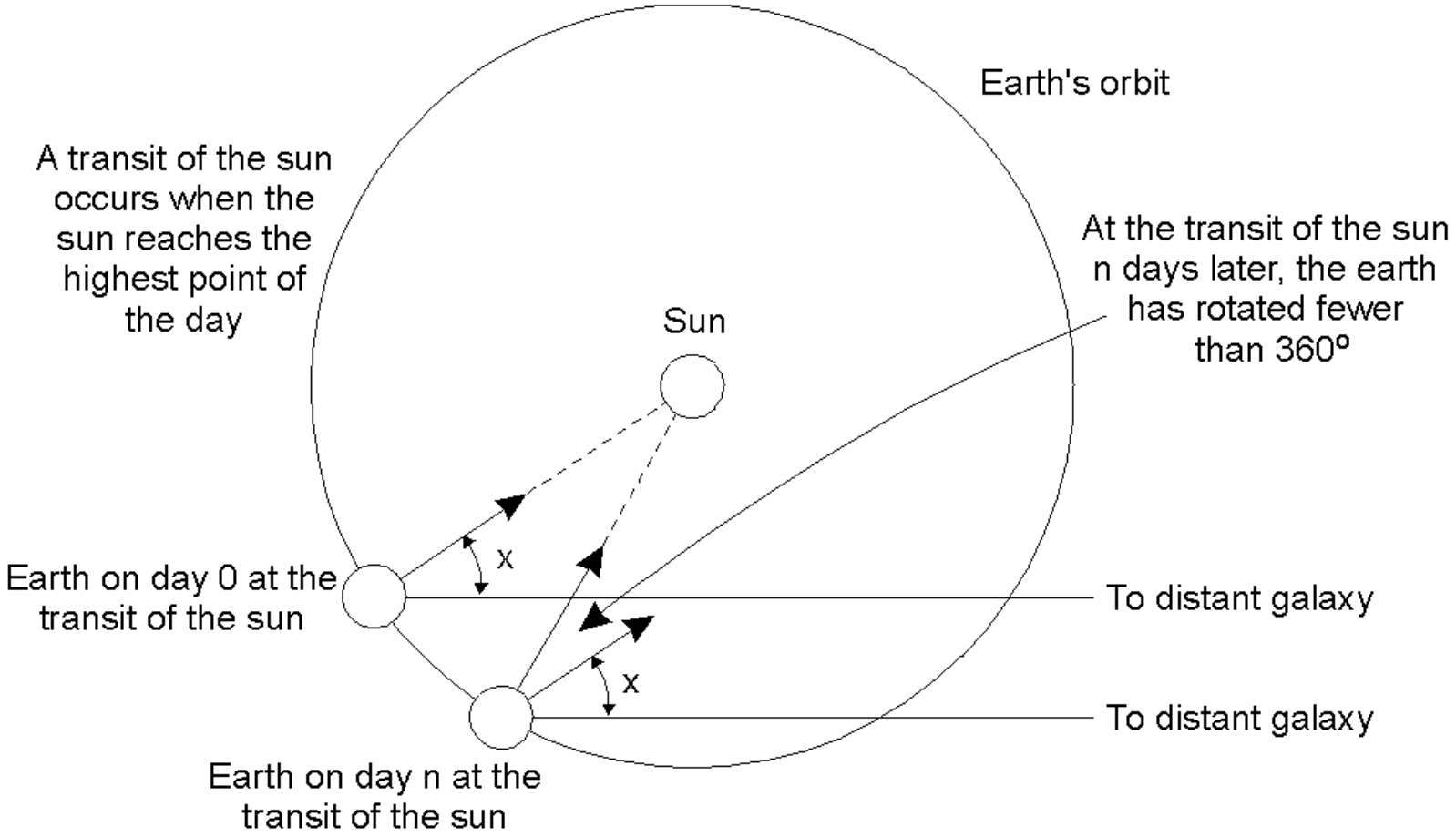
Kostas Magoutis
magoutis@cse.uoi.gr
http://www.cse.uoi.gr/~magoutis

Clock Synchronization



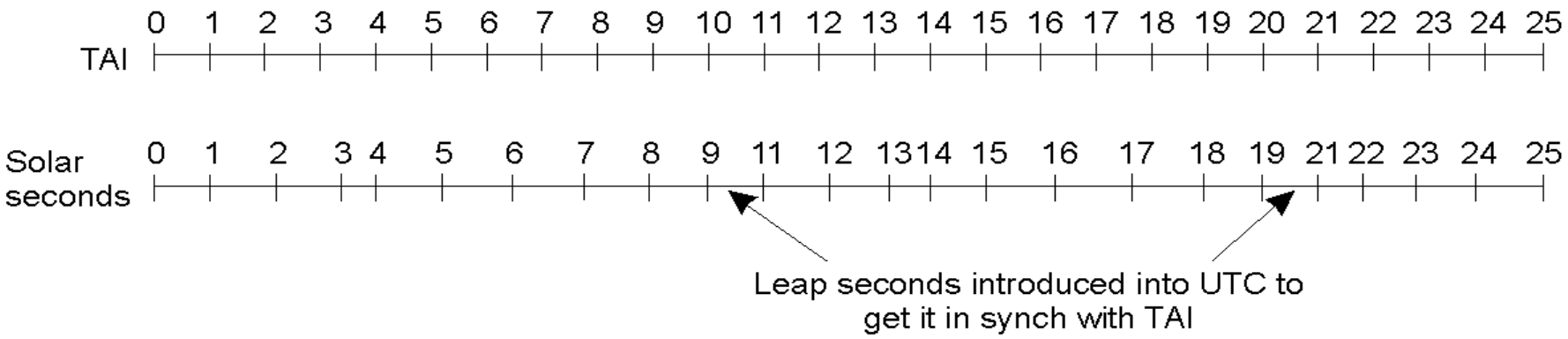
When each machine has its own clock, an event that occurred after another event may nevertheless be assigned an earlier time.

Physical Clocks



Computation of the mean solar day.

Atomic vs. solar clocks

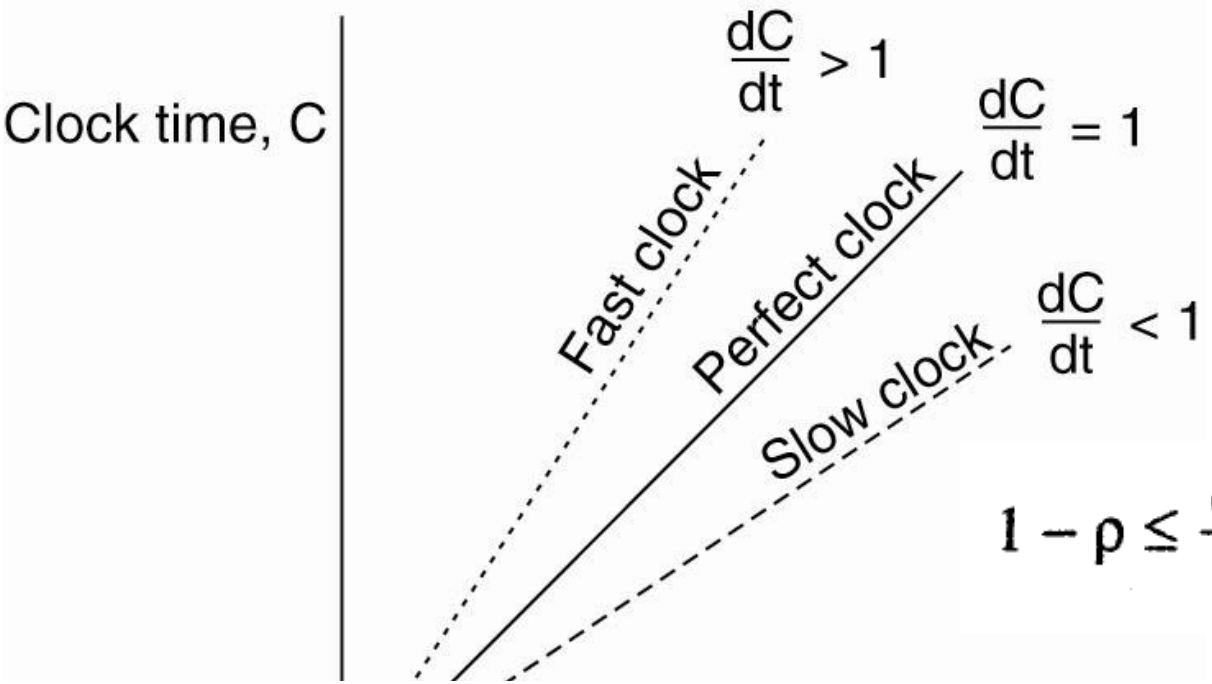


TAI seconds are of constant length, unlike solar seconds. Leap seconds are introduced when necessary to keep in phase with the sun.

Hardware clocks

- Quartz crystal, oscillates at certain freq.
  - 60Hz = 60 interrupts (ticks) per second
  - Every 60 ticks add 1 second to h/w clock
- Clocks in different computers runs at different rates
  - Clock skew
- Relative error $\sim 10^{-5}$
  - For 60Hz, between 215,998 and 216,002 ticks per hour (maximum drift rate)

Clock Synchronization Algorithms

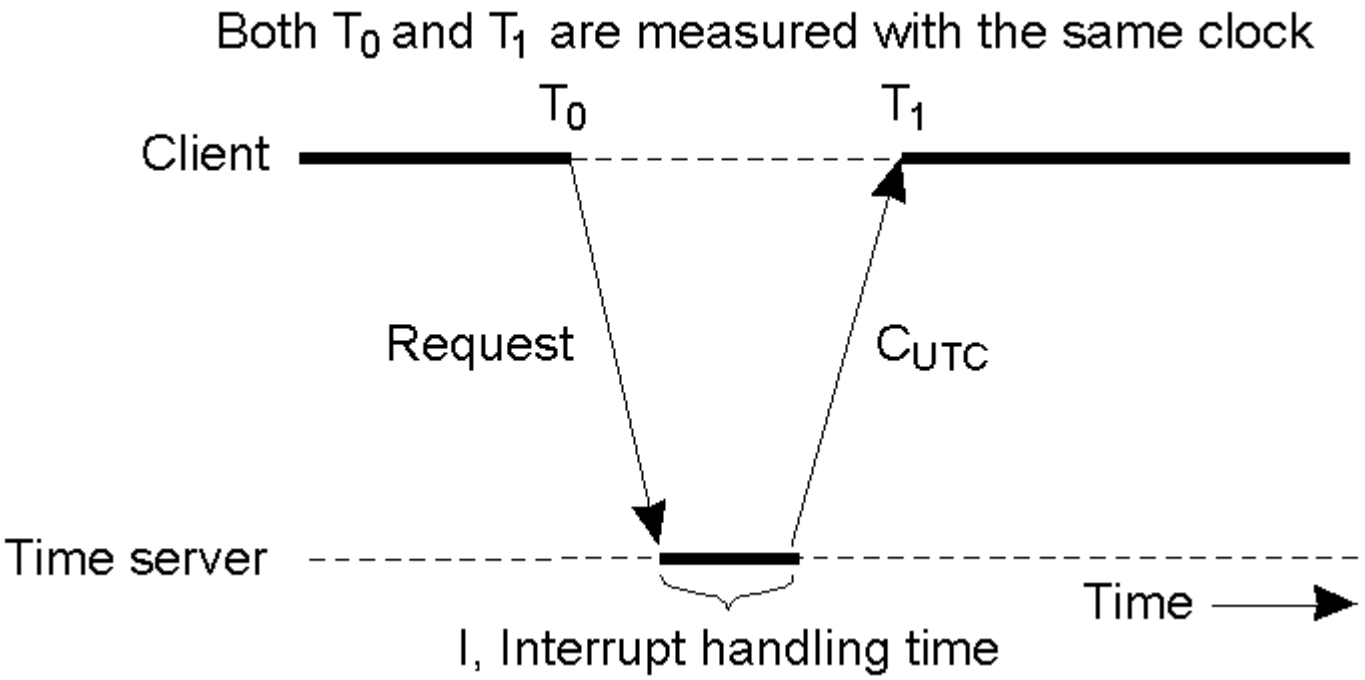


For two clocks never to diverge by more than θ, the clocks must be re-synchronized at least once every θ/2ρ

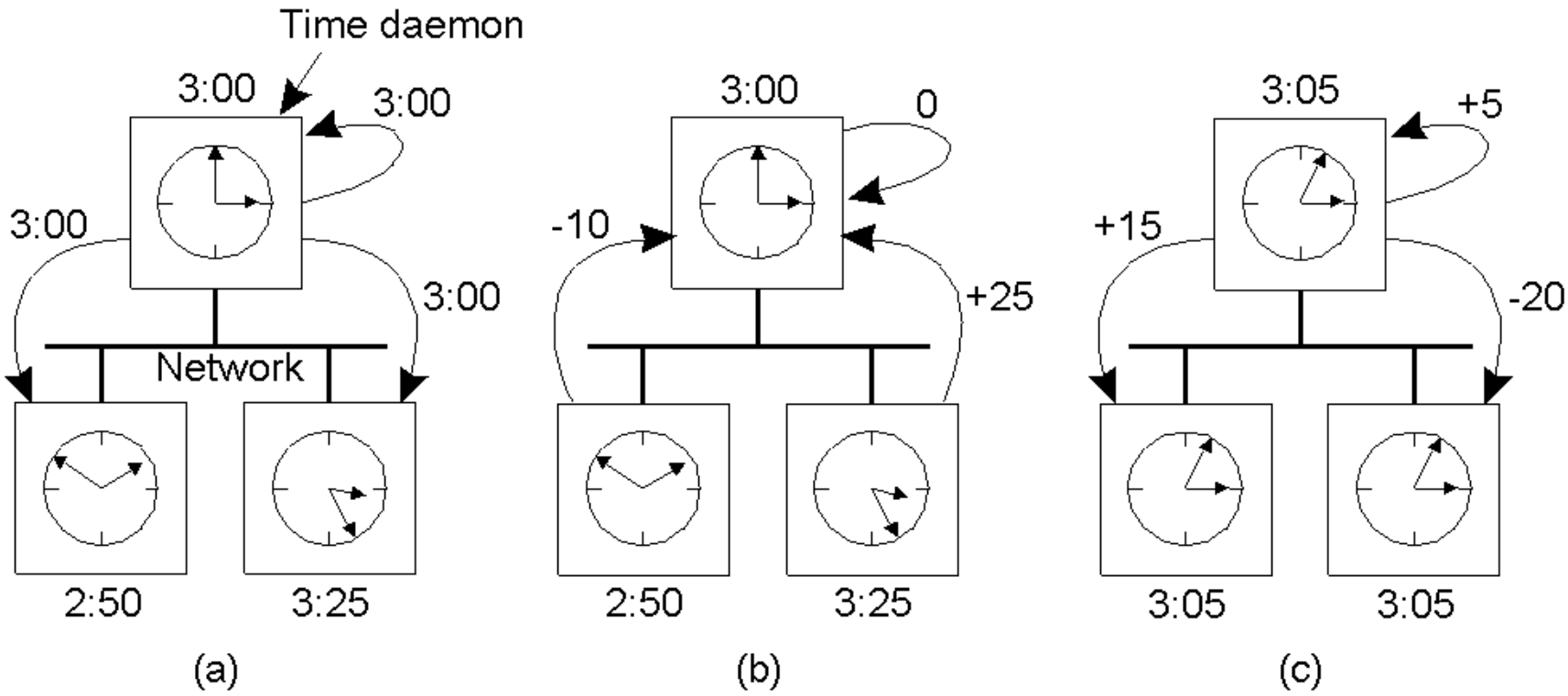
The relation between clock time and UTC when clocks tick at different rates

Cristian's Algorithm

Getting the current time from a time server.

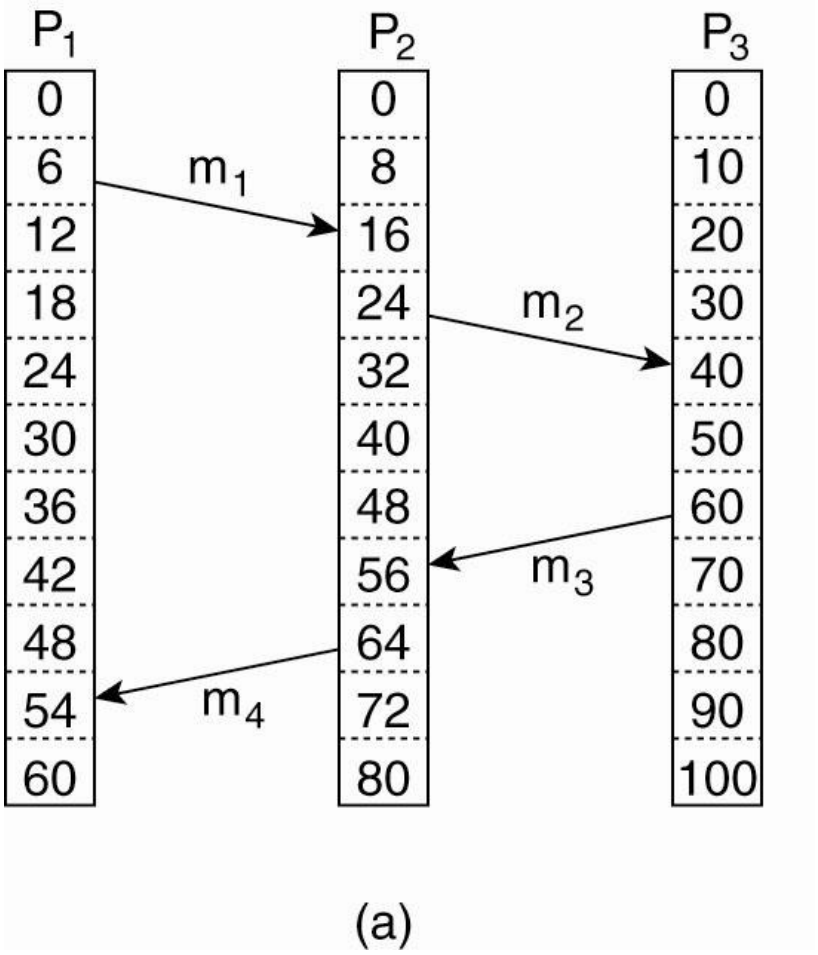


The Berkeley Algorithm



a) The time daemon asks all the other machines for their clock values
b) The machines answer
c) The time daemon tells everyone how to adjust their clock

Lamport's Logical Clocks

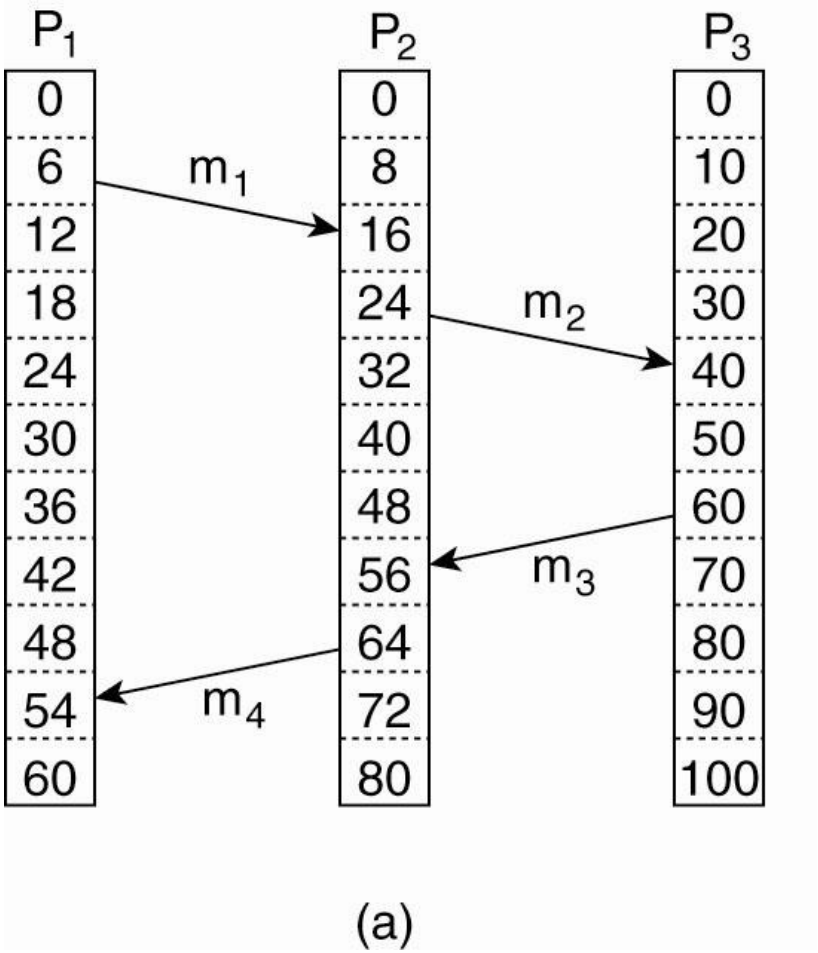


Three processes, each with its own clock
The clocks run at different rates

Lamport's Logical Clocks (1)

- The "happens-before" relation $\rightarrow$ can be observed directly in two situations:
- If *a* and *b* are events in the same process, and *a* occurs before *b*, then $a \rightarrow b$ is true.
- If a is the event of a message being sent by one process, and *b* is the event of the message being received by another process, then $a \rightarrow b$

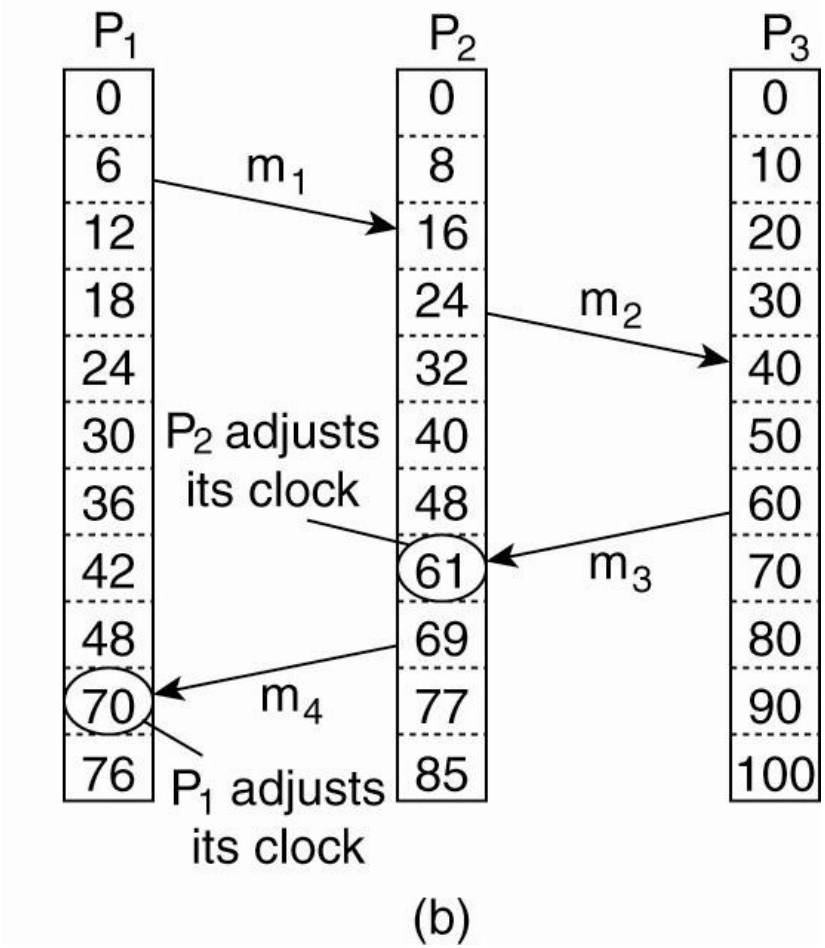
Lamport's Logical Clocks (2)



Three processes, each with its own clock

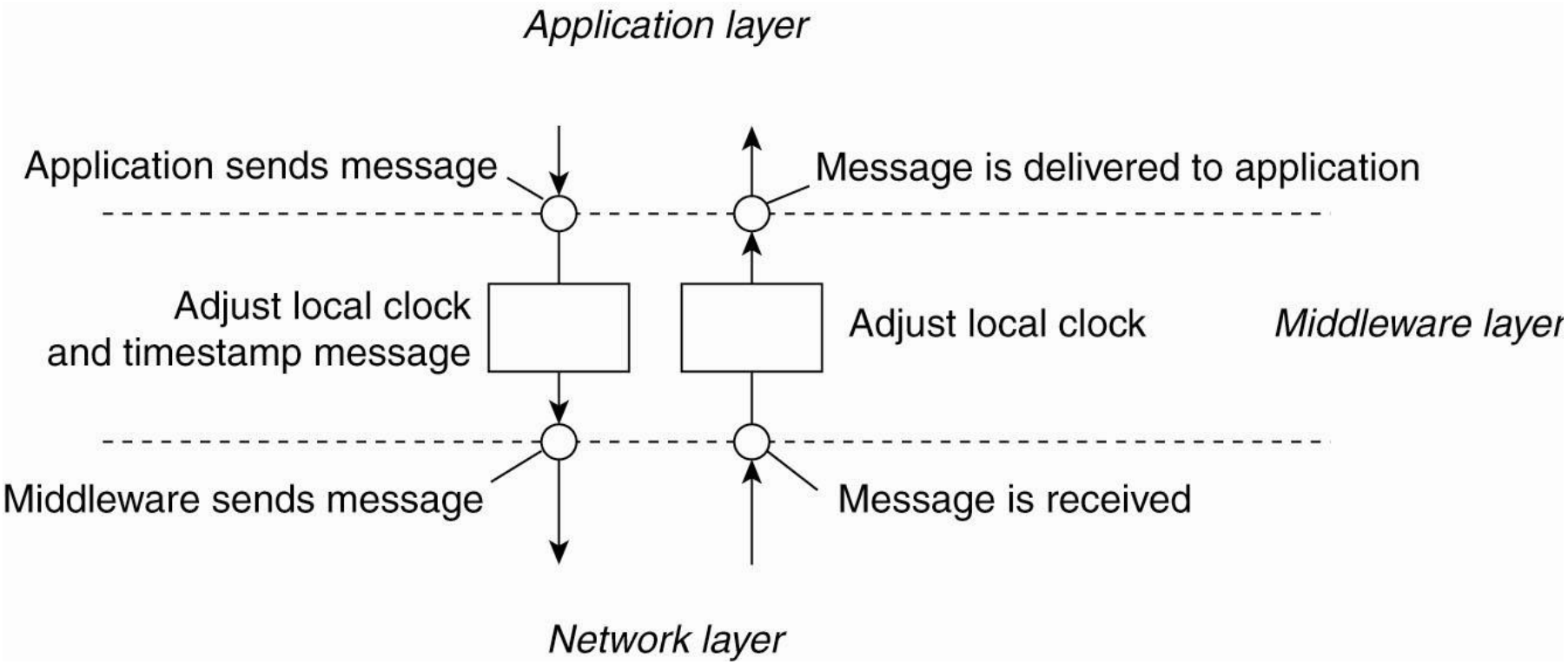
The clocks run at different rates

Lamport's Logical Clocks (3)



Lamport's algorithm corrects the clocks

Lamport's Logical Clocks (4)



The positioning of Lamport's logical clocks in distributed systems.

Lamport's Logical Clocks (5)

- Updating counter C_i for process P_i

1. Before executing an event P_i executes $C_i \leftarrow C_i + 1$.
2. When process P_i sends a message m to P_j, it sets *m*'s timestamp *ts (m)* equal to C_i after having executed the previous step
3. Upon the receipt of a message *m*, process P_j adjusts its own local counter as $C_j \leftarrow \max\{C_j, ts(m)\}$, after which it then executes the first step and delivers the message to the application.