Time, clocks and the ordering of events in a distributed system

L. Lamport

Computer Science Laboratory
SRI International

CACM, 21, 7 (1978), pp. 558-565.

1. The Relation *has happened before*

The relation *has happened before* ($\rightarrow$) is defined as the smallest relation satisfying the three following conditions:

a) if a and b are events in the same process and a comes before b, then $a \rightarrow b$,

b) if a is the sending of a message by a process and b its receipt by another process then $a \rightarrow b$,

c) if $a \rightarrow b$ and b $\rightarrow c$ then $a \rightarrow c$.

It is only a pre-order because we cannot always order events that occurred on different processes; these events are said to be *concurrent*.

If a did not happen before b, it cannot causally affect b.

2. Logical Clocks

Logical clocks verify the *clock condition*:

$$\text{if } a \rightarrow b \text{ then } C\langle a\rangle < C\langle b\rangle$$

We can derive from the clock condition two subconditions:

a) if a and b are events in process P_i and a comes before b, then $C_i\langle a\rangle < C_i\langle b\rangle$,

b) if a is the sending of a message by a process P_i and b its receipt by a process P_j then $C_i\langle a\rangle < C_j\langle b\rangle$,

and two implementation rules for systems of logical clocks:

a) each process P_i increments its clock C_i between two consecutive events,

b) if a is the sending of a message m by a process P_i then m includes a timestamp $T_m = C_i\langle a\rangle$; upon receiving a message m, process P_j sets its clock to a value greater than or equal to its present value and greater than T_m.

3. Ordering the Events Totally

Any system of logical clocks defines a total ordering on the set of all system events: we will say that $a \Rightarrow b$ if either $C_i\langle a\rangle < C_j\langle b\rangle$ or $C_i\langle a\rangle = C_j\langle b\rangle$ and $P_i < P_j$.

This ordering depends on the system of clocks and is not unique.

4. Anomalous Behaviors

Logical clocks have anomalous behaviors due to external interactions like phone conversations between users logged on different machines (or the transfer of a floppy between two machines). One way to avoid the problem is to redefine the *has happened before* over the set of all systems events plus the relevant external events $\underline{\xi}$. We can now define the strong clock condition:

For any events a, b in $\underline{\xi}$,

if $a \longrightarrow b$ then $C\langle a\rangle < C\langle b\rangle$

4. Physical Clocks

Reasonably accurate physical clocks are said to verify the two physical clock conditions:

a) there is a constant $\kappa << 1$ such that for all i:

$$|d\, C_i(t)/dt - 1| < \kappa$$

b) there is a constant ε such that for all i, j:

$$|C_i(t) - C_j(t)| < \varepsilon$$

We can derive the two implementation rules for physical clocks:

a) for each i, if P_i does not receive a message at physical time t then $C_i(t)$ is differentiable and $d\, C_i(t)/dt > 0$,

b) if P_i sends a message m at time t then m includes a timestamp $T_m = C_i(t)$; upon receiving the message m at time t', P_j sets $C_j(t')$ to the maximum of $C_j(t'\text{-}0)$ and $T_m + \mu_m$,

where μ_m is the minimum transmission delay between the two processes.