

Tiempo

Magnitud para medir **duración y separación de eventos**.

- Variable monotonica creciente.
- Discreta.
- No necesariamente vinculada con la hora de la vida real.

Usos

- Ordenar y sincronizar.
- Marcar ocurrencia de un suceso (**timestamps**).
- Contabilizar duración entre sucesos (**timespans**).

Relojes Físicos

- Locales vs. Globales.
- Pueden ser: cuarzo, atómicos, por GPS.
- Referencias globales. GMT, UTC, GPS time, TAI.

Drift

- **Descalibración** x cambios de temperatura, presión, humedad.
- **No son confiables** para comparación.
- **Sincronización periódica necesaria.**
 - Desvío respecto de relojes de referencia.
 - Aplicar corrección cambiando frecuencia.
 - **Nunca atrasar** un reloj.
 - Algoritmo de Cristian: $T_{new} = T_{server} + (T_1 - T_0) / 2$

Network Time Protocol (NTP)

Objetivos

- **Sincronización.** Incluso con delays en la red.
- **Alta disponibilidad.** Rutas y servidores redundantes.
- **Escalabilidad.**

Estructura de Servidores

Basada en *stratums* (capas?).

- E0: **Master clocks.**
- E1: Servidores conectados directamente a master clocks.
- E2: Servidores sincronizados con E_1.
- EN: Servidores sincronizados con E_N-1

Modelos de Sincronización

- **Multicast/Broadcast.**
 - LANs de alta velocidad.
 - Eficiente, baja precisión.
- **Cliente-Servidor (RPC).** Grupos de aplicaciones.
- **Simétrico (Peer Mode).**
 - Sincronizados entre sí, backup mutuo.
 - Estratos 1 y 2.

Relojes Lógicos

Conceptos previos

- **Evento:** suceso relativo al proceso P_i que modifica su estado.
- **Estado:** valores de todas las variables del proceso P_i en un momento dado.
- **Ocurre antes.** Relación de causalidad entre eventos o estados tales que:
 - $a \rightarrow b$, si a, b pertenecen al mismo proceso P_i y a ocurre antes de b .
 - $a \rightarrow b$, si a es el envío (en P_i) de un mensaje a P_j , y b es la recepción (en P_j).
 - **Transitividad:** $a \rightarrow c$, si $a \rightarrow b$ y $b \rightarrow c$.

Definición

Función C monotónica creciente que **mapea estados con un número natural**, y garantiza $s \rightarrow t \Rightarrow C(s) < C(t)$ (para todos los estados locales posibles del sistema).

Algoritmo de Lamport

- Conjunto de N procesos, c/u inicia con $reloj = 0$.
- Evento interno $\rightarrow reloj += 1$.
- Evento de envío $P_i \rightarrow P_j$:
 1. (P_i) $reloj += 1$
 2. (P_i) Envía mensaje a P_j incluyendo el valor actualizado.
 3. (P_j) Recibe mensaje y obtiene $reloj$ de P_i .
 4. (P_j) $reloj = \text{Max}(reloj_Pi, reloj_anterior) + 1$.

Problema: No cumplen la recíproca ($C(s) < C(t) \not\Rightarrow s \rightarrow t$).

Vectores de Relojes

Mapeo de todo estado del sistema compuesto por k procesos, con un vector de k números naturales, y garantiza $s \rightarrow t$ sii $s.v < t.v$ (con $A.v$ vector de relojes de A).

Obs. $s.v < t.v$ sii:

- Cada valor del vector $s.v$ es \leq a los de $t.v$.
- Al menos hay una relación de $<$ estricta.

Implementación

- Cada proceso incrementa su reloj.
- Cuando recibimos un mensaje:
 1. Agarra ambos vectores y, posición por posición, se queda con el máximo.
 2. Incrementa en 1 su propio contador.