# Sincronización de relojes

Paula Sanzana y Samuel Goldfarb

# Sincronización de relojes

- Consiste en ajustar los relojes de distintos sistemas para que tengan el mismo tiempo global.
- Relojes desincronizados son un problema en algunos sistemas distribuidos donde es necesario que todos los nodos lleven el tiempo con precisión.
- Aún cuando estén inicialmente ajustados, distintos relojes se irán desincronizando con el tiempo. Por lo tanto, deben sincronizarse periódicamente.
- A veces no es práctico mantener los relojes sincronizados, para estos casos se usa un reloj lógico.

# Soluciones Actuales

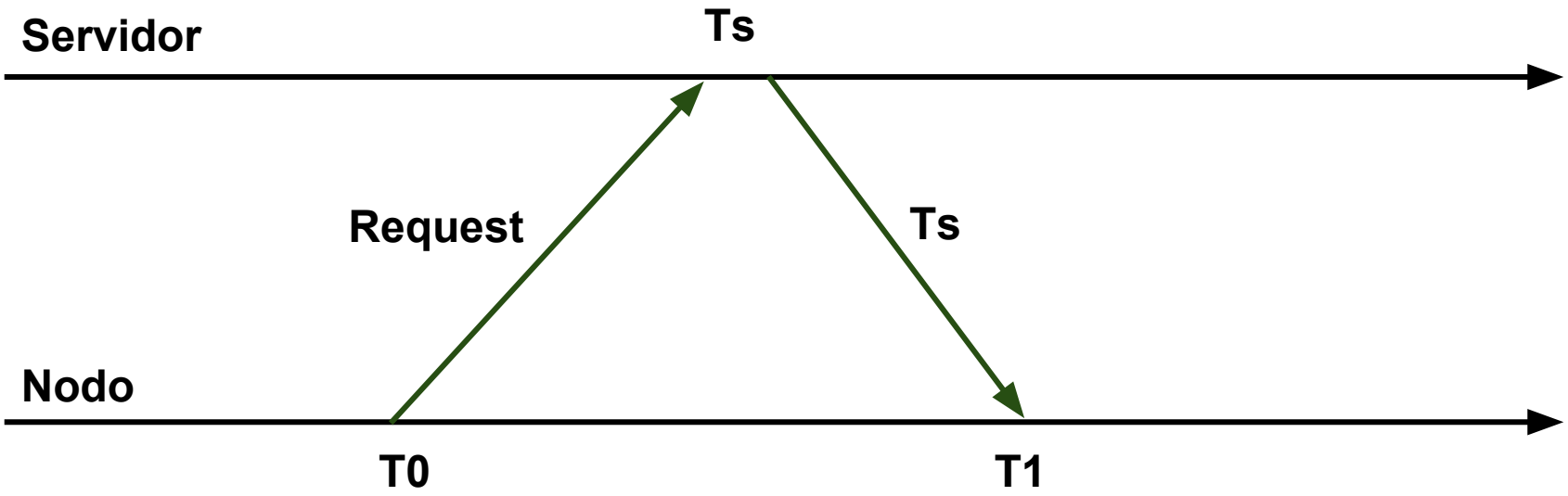
# Cristian’s Algorithm

Cuando existe un nodo en el sistema con la hora correcta por definición, se puede usar *Cristian’s Algorithm*:

- Cuando un nodo quiera sincronizar su reloj, envía una *request* al nodo maestro.
- El nodo maestro envía de vuelta su hora actual **T**.
- El nodo inicial ajusta su reloj a **(T + RTT/2)**, donde **RTT** es el tiempo que demoró en llegar la respuesta desde que se envió la *request*.

Este algoritmo fue introducido por Flaviu Cristian en 1989.

# Cristian’s Algorithm




RTT = T1 - T0

T = Ts + RTT/2

# Berkeley Algorithm

Cuando no se puede asumir un nodo con la hora correcta, se puede sacar un promedio entre los nodos usando el *Berkeley Algorithm*:

- Se elige un nodo maestro si es que no lo hay.
- El nodo maestro le pide la hora a todos los otros nodos y anota las horas recibidas con la misma fórmula que *Cristian's Algorithm*: T + RTT/2.
- El nodo maestro saca un promedio de los relojes, y luego envía a cada nodo la **diferencia** positiva o negativa que existe entre su reloj y la hora promedio.
- Cada nodo ajusta su reloj.

Este algoritmo fue desarrollado en la Universidad de California, Berkeley en 1989

# Algunas consideraciones

- Los algoritmos anteriores tienen una precisión de **RTT/2** segundos, lo que puede ser inaceptable en algunos casos.
- Mientras más constante sea el tiempo de viaje (**menor latencia**), mayor será la precisión del algoritmo.
- Estos algoritmos son ideales para redes LAN, porque se caracterizan por tener poca latencia.
- El Algoritmo de Berkeley puede ignorar las horas de los nodos que se encuentran demasiado alejadas de las demás. Esto permite que un solo nodo defectuoso no desvíe drásticamente la hora promedio. Para esto se debe configurar una tolerancia de antemano.

# Algunas consideraciones

- Generalmente se evita retroceder los relojes cuando se recibe una diferencia negativa, porque muchos procesos requieren que el tiempo del sistema sea monótono. Parar el reloj durante el tiempo requerido tampoco es una opción.
- Lo que se hace entonces es que se aumenta o disminuye un poco la velocidad del reloj, aplicando la corrección en un período más largo de tiempo.

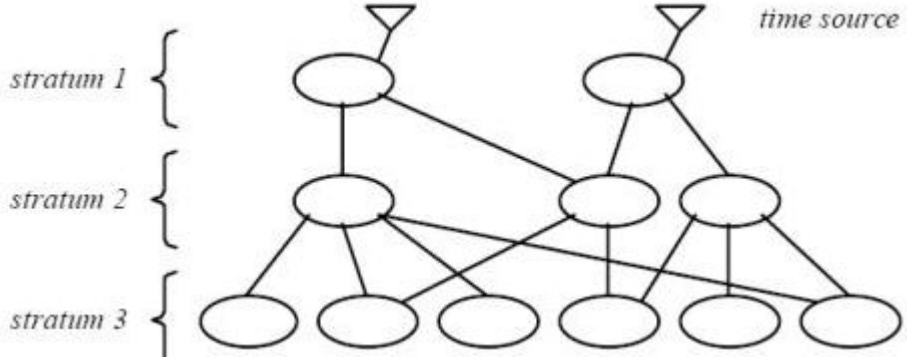
# Network Time Protocol

Es un estándar de internet que pretende:

- Permitir que los clientes de internet estén sincronizados con el UTC (universal coordinated time) independiente del delay en mensajes.
- Proveer un servicio confiable ante pérdidas de conexión.
- Permitir  clientes sincronizarse frecuentemente y compensar los efectos del *clock drift*.
- Proveer protección ante interferencias.

# NTP: Estructura

- Los servidores de NTP están arreglados en estratos.
- El primer estrato está conectado directamente con una fuente de tiempo precisa.
- El segundo estrato contiene máquinas sincronizadas a partir del primer estrato. Y así para el resto de estratos.
- Todos estos servidores forman una *subred de sincronización*.




Ejemplo de subred de sincronización NTP.
(Fuente)

# NTP: Modos de sincronización de las máquinas

- Modo simétrico activo
- Modo simétrico pasivo
- Modo de procedimiento de llamada
- Modo multicast

# Simple Network Time Protocol

- Versión simplificada de NTP.
- Pensado para ambientes donde la implementación de NTP no es necesaria o es injustificada.
- Pensado para ser usado en los estratos más altos de la subnet de sincronización más que para sincronizar servidores de tiempo.
- Puede operar en modos:
  - Unicast
  - Multicast
  - Anycast

# Lamport Timestamps

A veces no es práctico llevar relojes sincronizados entre todos los nodos de un sistema distribuido. Los Tiempos de Lamport son una forma de llevar el *orden* de los eventos en un sistema distribuido, usando un *reloj lógico*.

Algoritmo:

- Cada nodo lleva un contador que parte en **0**.
- El contador aumenta con cada evento interno y con cada mensaje enviado.
- Al enviar un mensaje, cada nodo debe enviar su contador actual.
- Al recibir un mensaje, cada nodo aumenta su contador al contador recibido si el contador recibido es mayor al actual.

# Lamport Timestamps

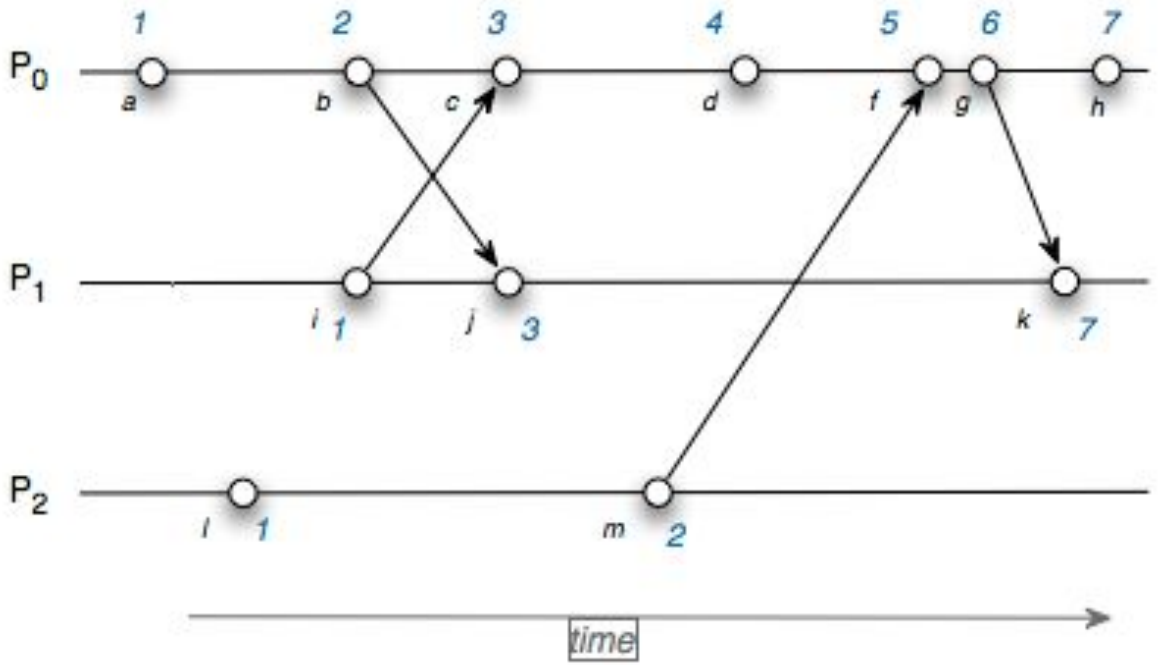



Imagen: https://www.cs.rutgers.edu/~pxk/rutgers/notes/clocks/index.html

# Lamport Timestamps

El algoritmo nos garantiza lo siguiente:

- *Si el mensaje **A** influye en el mensaje **B**, entonces necesariamente **TA < TB.***

De esta manera, si un nodo recibe dos mensajes al mismo tiempo, **atiende primero al que tenga el menor contador**.

Por ejemplo, si un nodo de almacenamiento recibe una solicitud de escritura del nodo **A** y una solicitud de lectura del nodo **B** al mismo tiempo, y **TA < TB,** entonces no es posible que el mensaje de **A** haya sido afectado por el mensaje de **B**, entonces se atiende a **A** primero.

Este algoritmo fue introducido por Leslie Lamport en 1978.

# Vector Clocks

- Lamport Timestamps no permite saber si dos pares de eventos están relacionados, *vector clocks* es propuesto como una solución a este problema.
- Se trata de un vector de N enteros para N procesos en un sistema.
- Cada proceso mantiene su propio vector para marcar el tiempo de sus eventos localmente.
- Como en Lamport Timestamps, el vector de enteros es enviado con cada mensaje.

# Sincronización de relojes

Paula Sanzana y Samuel Goldfarb

# Más información

Wikipedia ofrece una maravillosa explicación de estas soluciones y sus ventajas:

- https://en.wikipedia.org/wiki/Clock_synchronization
- https://en.wikipedia.org/wiki/Cristian%27s_algorithm
- https://en.wikipedia.org/wiki/Berkeley_algorithm
- https://en.wikipedia.org/wiki/Network_Time_Protocol
- https://en.wikipedia.org/wiki/Lamport_timestamps
- https://en.wikipedia.org/wiki/Vector_clock