DEPARTAMENTO DE

INGENIERÍA INFORMÁTICA

UNIVERSIDAD DE SANTIAGO DE CHILE

SINCRONIZACIÓN

13169

SISTEMAS DISTRIBUIDOS

Motivación

- Comunicación y procesos cumplen un rol fundamental en los SD.
- La manera en que los procesos cooperan y se sincronizan resulta igual de importante.
- La asignación de nombres soporta, parcialmente, la cooperación de recursos y entidades en general.
- Sincronización distribuida es más compleja que la sincronización en sistemas mono o multiprocesador.

Sincronización de Reloj

- A nivel local, lograr un acuerdo respecto al tiempo es sencillo ya que cada proceso consulta el mismo “reloj”.

- A nivel distribuido no es trivial lograr un acuerdo sobre el tiempo.

- Consideremos el caso de *make* de UNIX.
  - *Make* compila los archivos de salida respecto a la última vez que los archivos de entrada han sido modificados.
  - De esta forma, no pierde tiempo en compilar objetos ejecutables que no han variado.
  - A nivel local es algo trivial, todos tienen el mismo concepto del “tiempo”.
  - A nivel distribuido, sin embargo, pueden existir errores.

Sincronización de Reloj

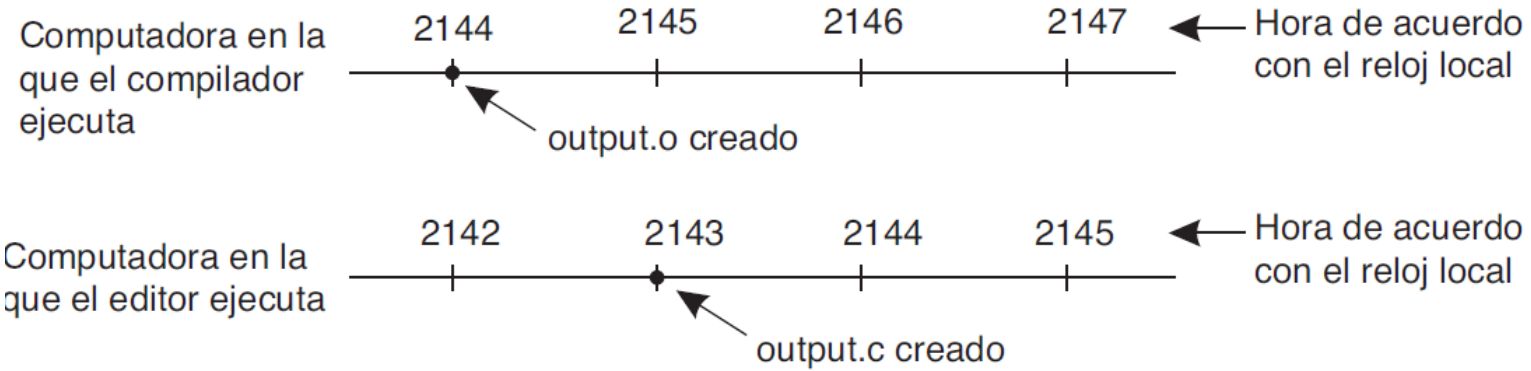

Figura 6-1. Cuando cada máquina tiene su propio reloj, es posible que a un evento que ocurrió después de otro se le asigne una hora anterior.

Relojes Físicos

- Todos los computadores contabilizan el tiempo.
- Típicamente se menciona que tienen “relojes”, pero no son relojes en cuanto al uso usual de la palabra.
- Es más correcto hablar de **Timer** o **Cronómetros**.
- Dicho Timer consta, en general, de un cristal de cuarzo mecanizado con precisión que oscila dependiendo del corte, tipo de cristal utilizado y cantidad de tensión.
- Hay dos registros asociados al cristal, el **contador** y el **registro mantenedor**.

Relojes Físicos

- Cada oscilación del cristal disminuye el contador en uno.
- Cuando el contador llega a 0, se genera una interrupción (**Tic de Reloj**) y el contador se reinicia respecto al registro mantenedor.
- En una sola máquina, no importa si el reloj está desfasado. Básicamente, lo importante es que los tiempos relativos sean consistentes respecto a los procesos internos.
- Cuando se tienen varias CPU, cada una con sus propios relojes, la situación cambia. Si se tienen *n* CPUs los *n* cristales funcionan a velocidades ligeramente diferentes.
- Esta diferencia, se conoce como **distorsión de reloj**.

Relojes Físicos

¿Cómo se sincronizan los distintos relojes en un sistema distribuido?

- En principio, se mide el tiempo respecto a la trayectoria solar.
- En 1940, se establece que la rotación de la tierra no es constante.
- En 1948, se inventa el reloj atómico que permite cuantificar el tiempo de manera más exacta.
- Instauración del **TAI (Tiempo Atómico Internacional)**. Promedio de relojes de cesio 133 de distintos lugares del mundo.

Relojes Físicos

¿Cómo se sincronizan los distintos relojes en un sistema distribuido?

- El problema del TAI es que no va acorde al tiempo solar (debido a que tiempo solar se va volviendo más lento).
- Para resolver este problema se introducen segundos vacíos en el **UTC (Tiempo Universal Coordinado)**.
- Con estos dos estándar, se logra tener un tiempo común a nivel mundial.

Relojes Físicos

¿Cómo se sincronizan los distintos relojes en un sistema distribuido?

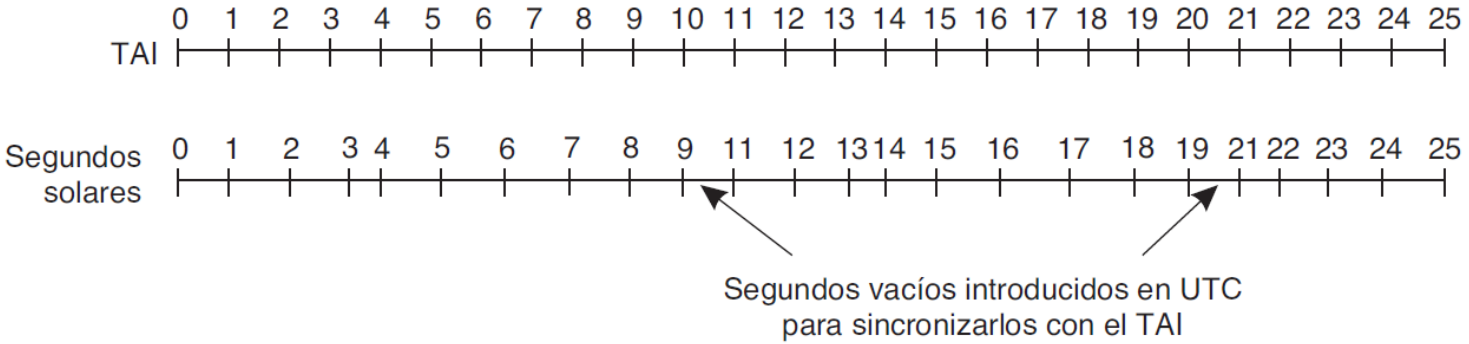

Figura 6-3. Los segundos TAI son de longitud constante, a diferencia de los segundos solares. Los segundos vacíos se introducen cuando es necesario mantenerlos en fase con el Sol.

Sistema de Posicionamiento Global

- El **GPS** es un sistema distribuido basado en satélites.
- Utilizado con fines militares en un inicio, pero ampliamente utilizado para fines civiles hoy en día.
- Se utilizan 29 satélites que tienen hasta 4 relojes atómicos calibrados regularmente.
- Cada satélite transmite su posición de manera continúa y, respecto al tiempo de los mensajes, se logra tener la posición de los dispositivos receptores.

Sistema de Posicionamiento Global

- Para calcular la posición, consideremos un caso bidimensional con 2 satélites.
- Se puede extender a más satélites.
- **¿Problema?** No se puede asumir que todos los relojes están perfectamente sincronizados.
- Existe una corrección por parte del software respecto a los problemas. Como considerar el ingreso de segundos vacíos, la propagación no es constante, no se conoce la posición exacta de los satélites, etc.

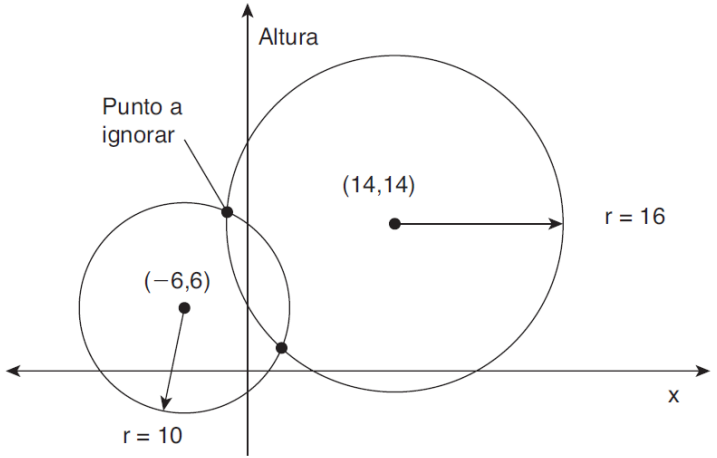

Figura 6-4. Cálculo de una posición en un espacio bidimensional.

Sistema de Posicionamiento Global

Ejemplo – Sealite

- Empresa que utiliza GPS para la sincronización de ritmos de linternas marinas.
- Utiliza estas linternas para marcar un canal, un puerto o un río mediante linternas independientes que destellan todas ellas de forma sincronizada.
- Logra la sincronización mediante satélites y relojes atómicos.

Algoritmos de Sincronización de Relojes

- Todos los algoritmos tienen como base el mismo modelo del sistema.
- Se asume que cada máquina tiene un cronometro que ocasiona una interrupción *H* veces por segundo.
- Cuando el cronometro se apaga, el manipulador de interrupciones agrega 1 al reloj del software. Con esto, se tiene la cantidad de marcas (interrupciones) desde un momento determinado.
- Dado que cada reloj tendrá sus propios tiempos, se definen distorsiones máximas que pueden existir en consideración del tiempo UTC.
- Con esto último, se catalogan los relojes como **reloj rápido**, **reloj perfecto** y **reloj lento**.

Algoritmos de Sincronización de Relojes

Algoritmo de Cristian

- El algoritmo propone que los clientes se conectan con un servidor que tiene el tiempo exacto.
- El problema reside en que existen retrasos de mensaje que ocasionarán que el tiempo reportado no esté actualizado.
- El truco está en estimar los retrasos de la mejor forma. Para esto, se miden los tiempos de propagación de los mensajes y se ajusta el reloj respecto a la estimación generada y el tiempo obtenido desde el servidor.
- El ajuste se debe realizar de forma gradual introduciendo cambios en la cantidad de segundos para producir la siguiente interrupción.

Algoritmos de Sincronización de Relojes

Algoritmo de Cristian

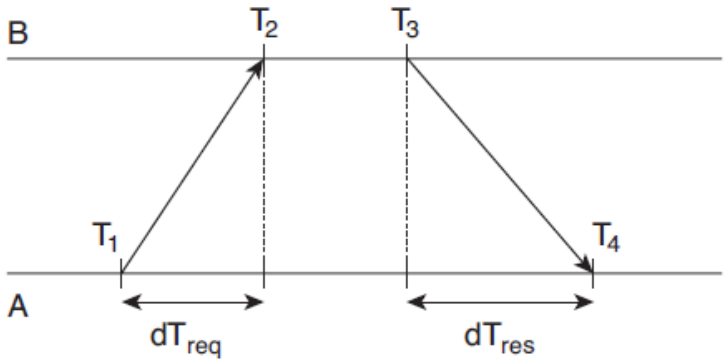

Figura 6-6. Obtención del tiempo actual desde un servidor de tiempo.

Algoritmos de Sincronización de Relojes

Algoritmo de Berkley

- En muchos algoritmos, el servidor del tiempo es pasivo.
- En el algoritmo de Berkley, es todo lo contrario. Se cuenta con un daemon desde el servidor del tiempo que pregunta cada ciertos intervalos el tiempo actual de cada máquina.
- Basado en las respuestas, calcula un tiempo promedio y le indica a las máquinas que adelanten o retrasen sus relojes.
- Notar que, dado que es una concordancia entre las máquinas, es suficiente que las máquinas tengan concordancia en sus tiempos, indiferente del tiempo UTC por ejemplo.

Algoritmos de Sincronización de Relojes

Algoritmo de Berkley

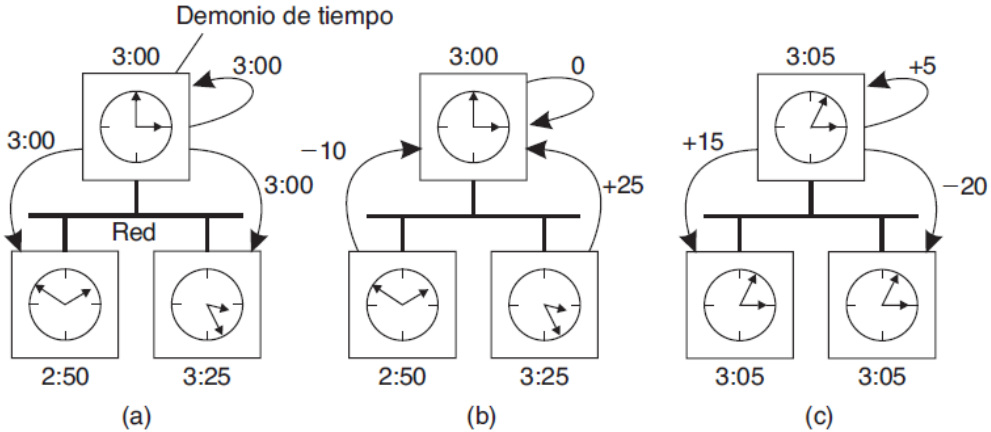

Figura 6-7. (a) El demonio de tiempo pregunta a las otras máquinas los valores de sus relojes. (b) Las máquinas responden. (c) El demonio de tiempo les indica cómo ajustar sus relojes.

Algoritmos de Sincronización de Relojes

Network Time Protocol (NTP)

- Cristian y Berkley están pesando para redes LAN.
- NTP está generado para proveer servicios de tiempo en Internet.
- Utiliza un sistema de jerarquía de **estratos de reloj**, donde los sistemas de estrato 1 están coordinados con un reloj externo. Los sistemas de estrato 2 derivan su tiempo de uno o más de los sistemas de estrato 1 y así consecutivamente.
- Se calculan los tiempos de propagación aplicando el concepto de dispersión respecto al servidor que se solicita la hora. Posterior a eso, se aplica un fase para la modificación de la frecuencia de los relojes en base a la diferencia calculada.

Relojes Lógicos

- Hasta el momento, hemos supuesto que la sincronización de relojes está naturalmente relacionada con el tiempo real.
- Sin embargo, puede ser suficiente que cada sistema coincida con un tiempo actual.
- Aún más, muchas veces lo importante es el orden en que se generan los eventos, indiferente de los tiempos locales de cada sistema.
- Con esta idea, nacen los algoritmos que utilizan **relojes lógicos**.

Relojes Lógicos

Relojes Lógicos de Lamport

- Para sincronizar los relojes lógicos, Lamport definió la relación llamada **ocurrencia anterior**.
- La expresión ***a*** → ***b*** se lee como "*a* ocurre antes que *b*". Esto quiere decir que todos los procesos coinciden en que el evento *a* ocurre primero y, posterior a eso, ocurre el evento *b.*
- Esto se puede observar claramente en dos situaciones.
- Si *a* y *b* son eventos del mismo proceso y *a* ocurre antes que *b*, entonces ***a*** → ***b*** es verdadera.
- Si *a* es el evento en que un proceso envía un mensaje y *b* es el evento de recepción del mensaje, entonces ***a*** → ***b*** también es verdadera.

Relojes Lógicos

Relojes Lógicos de Lamport

- La ocurrencia es transitiva, es decir si $a \rightarrow b$ y $b \rightarrow c$ entonces $a \rightarrow c$.
- Si dos eventos *x* e *y*, ocurren en diferentes procesos que no intercambian mensajes, entonces no se puede determinar qué evento ocurre primero. Se dice que estos eventos son **concurrentes**.
- Se necesita una forma de medir la noción del tiempo, de forma que a cada evento se le pueda asignar un valor de tiempo *C(a)*. Con esta noción, si $a \rightarrow b$ entonces $C(a) < C(b)$.

Relojes Lógicos

Relojes Lógicos de Lamport

- Consideremos 3 procesos, cada uno con su propio reloj.
- Se hacen ajustes de los relojes locales en torno a cómo se generan los eventos.
- El algoritmo va corrigiendo los relojes paso a paso.

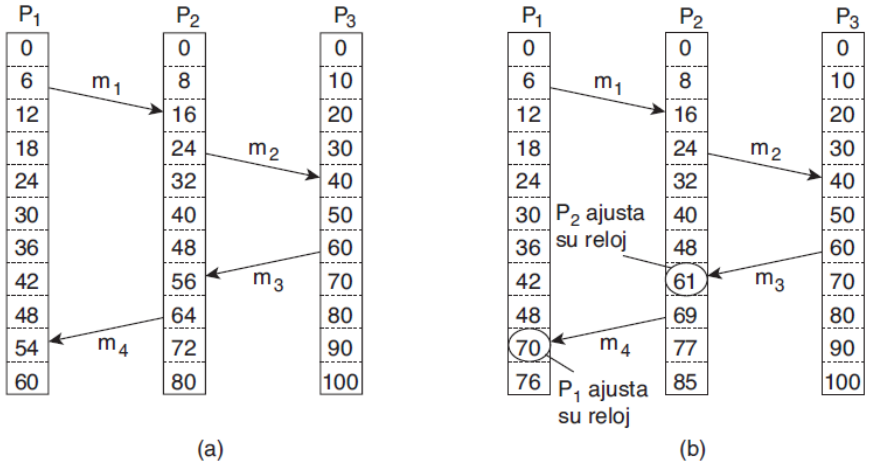

Figura 6-9. (a) Tres procesos, cada uno con su propio reloj. Los relojes avanzan a velocidades diferentes. (b) El algoritmo de Lamport corrige los relojes.

Relojes Lógicos

Relojes Lógicos de Lamport

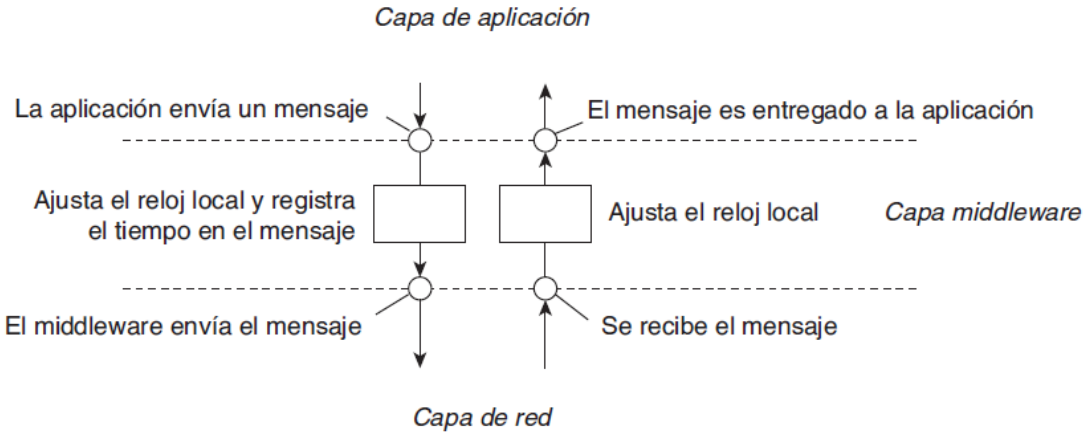

Figura 6-10. Posicionamiento de los relojes lógicos de Lamport en sistemas distribuidos.

Relojes Lógicos

Relojes Lógicos de Lamport - Ejemplo

- Consideremos una base de datos replicada.
- Si es una base de datos de transacciones, se puede generar una situación como la siguiente.

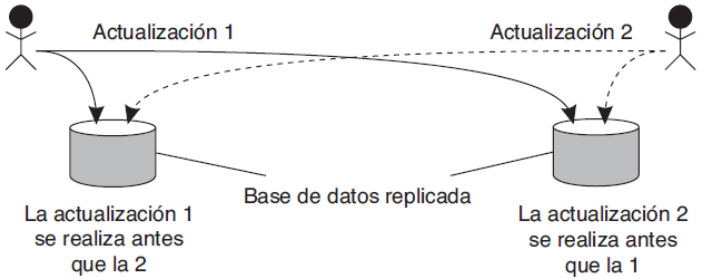

Figura 6-11. Actualización de una base de datos replicada, dejándola en un estado inconsistente.

Relojes Lógicos

Relojes Lógicos de Lamport - Ejemplo

- El problema es que cada actualización se debe realizar en el mismo orden en cada copia.
- No importa el orden, es decir, puede darse *actualización 1 → actualización 2* o *actualización 2 → actualización 1.*
- Se requiere una **transmisión totalmente ordenada**, es decir, una operación de transmisión mediante la cual todos los mensajes se entreguen en el mismo orden a cada destinatario.
- Esto último es posible mediante relojes lógicos de Lamport y ser utilizados en un sistema completamente distribuido como el presentado.

Relojes Lógicos

Relojes Vectoriales

- Los relojes lógicos de Lamport establecen causalidad según el *C(evento)*.
- Sin embargo, nada puede decirse sobre la relación entre dos eventos *a* y *b* sólo comparando *C(a)* y *C(b)*.
- Dicho de otra manera, como sólo se conocen los tiempos locales, nada dice sobre la relación de los eventos *a* y *b.* No se puede establecer que si *C(a) < C(b)* entonces necesariamente *a* ocurrió antes de *b.*

Relojes Lógicos

Relojes Vectoriales

- El problema es que los relojes de Lamport no capturan la **causalidad**.
- Dicha causalidad se puede capturar mediante **relojes vectoriales**.
- Un reloj vectorial, *VC(a)*, asignado a un evento *a*, tiene la propiedad que si *VC(a) < VC(b),* para algún evento *b*, entonces se sabe que el evento *a* precede en causalidad al evento *b.*
- Los relojes vectoriales se construyen de manera que cada proceso P_i mantenga un vector VC_i con las siguientes propiedades:
  - $VC_i[i]$ es el número de eventos que han ocurrido hasta el momento en P_i.
  - Si $VC_i[j] = k$, entonces P_i sabe que han ocurrido k eventos en P_j. Así, este es el conocimiento de P_i del tiempo local en P_j,

Relojes Lógicos

Relojes Vectoriales

- El reloj vectorial, por tanto, permite tener una causalidad de los eventos.
- Si no se puede establecer una relación entre VC(a) y VC(b) entonces quiere decir que son eventos **concurrentes**.
- Con esto, se puede generar una **transmisión causalmente ordenada**, es decir, garantizar que un mensaje se envía sólo si todos los mensajes que causalmente le precede han sido recibidos.

Relojes Lógicos

Relojes Vectoriales - Ejemplo

- Consideremos el mismo caso anterior, pero ahora si hay causalidad entre los eventos.

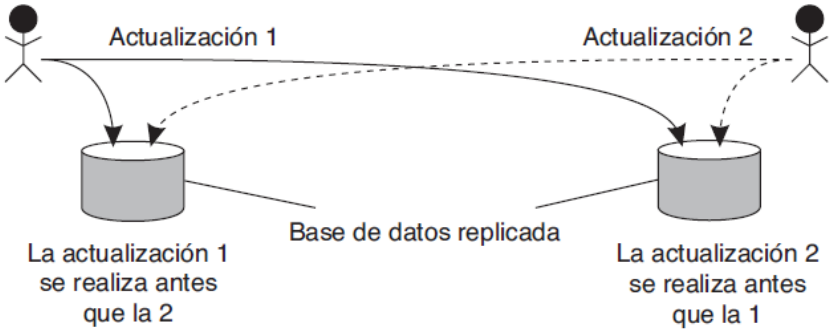

Figura 6-11. Actualización de una base de datos replicada, dejándola en un estado inconsistente.

DEPARTAMENTO DE
INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE SANTIAGO DE CHILE

Departamento de Ingeniería Informática
Universidad de Santiago de Chile

¿CONSULTAS?