

Sistemas distribuidos

Grado en Ingeniería Informática

Tema 3.1: Sincronización en SD

Departamento de Ingeniería Informática
Universidad de Cádiz

Escuela Superior de Ingeniería
Dpto. de Ingeniería Informática



Versión 1.1

Índice

- 1 Introducción
- 2 Tiempo físico
- 3 Sincronización
- 4 Método Cristian
- 5 Algoritmo de Berkeley
- 6 Network Time Protocol
- 7 Tiempo lógico
- 8 Tiempo lógico de Lamport
- 9 Vectores de tiempo
- 10 Tareas

Sección 1 | Introducción

Introducción

Propiedades de los Sistemas Distribuidos

- **La información relevante se distribuye** entre varias máquinas.
- Los procesos toman las decisiones sólo con base en la información disponible en **forma local**.
- Debe evitarse un punto único de fallo.
- **No existe un reloj común** o alguna otra fuente precisa del tiempo global.

Introducción

Problemas a tener en cuenta

- Tiempos y estados globales.
- Exclusión mutua.
- Algoritmos de elección. Problemas de consenso.
- Operaciones atómicas distribuidas: Transacciones.
- Cada nodo posee un reloj estándar.
- La comunicación entre nodos presenta retardos.
- Cada nodo presenta una visión subjetiva del estado global.

Introducción

Posibles soluciones

Propuesta de solución: un único reloj preciso + red dedicada para transmitir la señal sin retardos

- No es práctico (coste), o es inviable (Internet)

Solución: tiempo distribuido

- Cada nodo posee su propio reloj **tiempo físico local**
- Los relojes son imprecisos: necesarios ajustarlos periódicamente a un **tiempo físico de referencia**

Por otra parte, la gestión consistente del estado global requiere al menos ordenar los eventos producidos por los nodos

- **Tiempo lógico**

Introducción

Posibles soluciones

Sincronizar todos

- Sincronizar todos los relojes con reloj de referencia (NTP).

Sincronizar entre los nodos

- Estimar retrasos por la conexión (Algoritmo de Cristian).
- Establecer un promedio (Algoritmo de Berkeley).

No sincronizar, sólo ordenar eventos

- Ordenar los eventos, algoritmo de Algoritmo Lamport.
- Ordenar todos los nodos, relojes vectoriales.

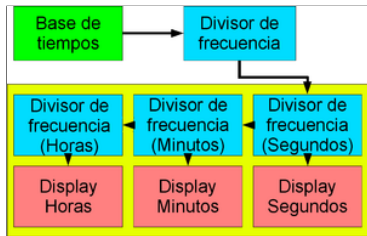
Sección 2 | Tiempo físico

Tiempo físico

Relojes de cuarzo

Los relojes de los ordenadores son de **cuarzo**

- Las moléculas en el cristal de cuarzo vibran millones de veces por segundo (piezoelectricidad), a una velocidad que *nunca* cambia. El ordenador usa las vibraciones en el reloj del sistema para tomar el tiempo de sus operaciones de procedimiento
- La frecuencia de oscilación varía con la temperatura
- deriva(retraso): $\sim 10^{-6}$ (90 ms en un día, 1s cada 11,6 días)



Tiempo físico

Relojes atómicos

Son de gran precisión pero muy caros

- deriva: $\sim 10^{-13}$ (9 ns en un día, 1s cada 300.000 años)
- precio: \$ 50.000 - \$100.000



Otros modelos pueden ser los NIST (National Institute of Standards and Technology) en <http://tf.nist.gov/timefreq/cesium/atomichistory.htm>

Tiempo físico

Definiciones

- Segundo solar o astronómico: $1/86.400$ del periodo de rotación de la Tierra (mean solar second) pese a ser perfectamente válido para las situaciones de la vida cotidiana, la Tierra no gira a velocidad constante (va perdiendo lentamente velocidad), por lo que no sirve como referencia
- Segundo atómico (IAT, 1967): 9.192.631.770 periodos de transición en un átomo de Cesio-133. Los relojes atómicos miden este tiempo deriva de $\sim 3 * 10^{-8}$ con el segundo solar (~ 1 s al año)
- Tiempo universal coordinado (UTC): medido en segundos atómicos, sincronizado con tiempo astronómico (diferencia > 900 ms \rightarrow inserción de 1 s)

Tiempo físico

Definiciones

- **Tiempo físico o de referencia** normalmente UTC (*Universal Time Coordinated*)
http://www.bipm.org/en/scientific/tai/time_server.html
- **Resolución** periodo entre dos actualizaciones del registro del tiempo local
- **Desviación** *offset, skew, θ* Diferencia entre el tiempo local y el tiempo físico de referencia de un instante
- **Deriva** *drift, δ* desviación por unidad de tiempo (lo que adelanta o retrasa el reloj)
- **Precisión** *accuracy* desviación máxima que se puede garantizar en el ajuste de un reloj

Sección 3 | Sincronización

Sincronización

de relojes físicos

- Los computadores de un sistema distribuido poseen relojes que no están sincronizados (derivas)
- Importante asegurar una correcta sincronización
 - En aplicaciones de tiempo real
 - Ordenación natural de eventos distribuidos (fechas de ficheros)
 - Análisis de rendimiento
- Tradicionalmente se han empleado protocolos de sincronización que intercambian mensajes
- Actualmente se puede mejorar mediante GPS

Sincronización

Externa

- Sincronización: procedimiento por el que se ajusta el valor de un reloj a un tiempo físico de referencia con una precisión preestablecida
- Referencias de tiempo UTC se difunden periódicamente por radio
- Precisión de receptores comerciales: Estaciones terrestres, Satélites geoestacionarios, Satélites GPS
- Usos: servicio horario preciso, contabilidad...

Sincronización

Interna

- Para muchas aplicaciones es más importante mantener bien sincronizados entre sí los relojes locales que conseguir una gran precisión en la sincronización externa (permite ordenar eventos (causalidad))
- Propuesta de solución: receptor UTC en cada nodo (no es práctico por coste)
- Solución: algoritmos de sincronización interna
 - **centralizados**: basados en un servidor específico
 - **distribuidos**: estadísticos

Sincronización

Métodos de Sincronización

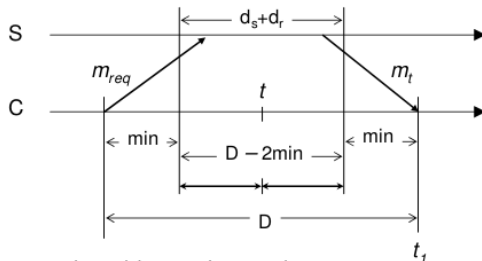
- UTC: Universal Coordinated Time
 - Transmisión de señal desde centros terrestres o satélites.
 - Una o más máquinas del sistema distribuido son receptoras de señal UTC.
- Método de Cristian
 - Es un algoritmo probabilístico
 - Es un algoritmo centralizado en el cual un servidor brinda el tiempo unificado
 - Este servidor puede estar conectado a un UTC para tener la hora correcta
- Algoritmo de Berkeley
 - Es un algoritmo centralizado en el cual un servidor es seleccionado como maestro
 - Este maestro NO puede estar conectado a un UTC para tener la hora correcta
 - El maestro estima el valor de los relojes de los esclavos
- NTP (Network Time Protocol)
 - Permitir sincronizar clientes con UTC sobre Internet
 - Jerarquía de servidores en diferentes estratos
 - Los fallos se solventan por medio de ajustes en la jerarquía

Sección 4 | Método Cristian

Método Cristian

Explicación

Algoritmo de Cristian



$t(m_t)$: tiempo devuelto por el servidor en el mensaje m_t

D : tiempo desde que se envía m_{req} y se recibe m_t

min : tiempo mínimo de transmisión de un mensaje

Suposición: el servidor asigna la referencia $t(m_t)$ en la mitad del intervalo $D - 2 \cdot min$ (instante t en el cliente). Así se logra la mejor precisión

Desviación: $\theta = t - t(m_t) = t_1 - D/2 - t(m_t)$

Precisión = $D/2 - min$

Método Cristian

Cuestiones

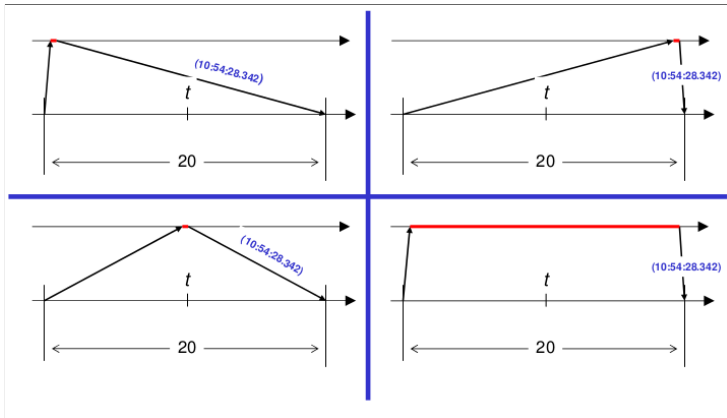
- Ejemplo del algoritmo de *Cristian* (3 peticiones):

<u>Pet.</u>	<u>D (ms)</u>	<u>t_i (hh:mm:ss.ms)</u>	<u>$t(m_i)$ (hh:mm:ss.ms)</u>
(1)	22	10:54:22.236	10:54:23.674
(2)	26	10:54:24.000	10:54:25.450
(3)	20	10:54:26.946	10:54:28.342

- ¿Qué petición debería usar el cliente? Calcula la precisión y desviación obtenidas
Precisión = 10 ms Desviación = 1406 ms
- Conociendo que el tiempo mínimo de transmisión es de 7 ms, ¿cambia en algo la respuesta anterior?
Precisión = 3 ms
- ¿Qué se necesitaría para obtener una precisión de 2 ms?
Siendo min = 7 ms, una petición con $D \leq 18$ ms

Método Cristian

Posibilidades



Método Cristian

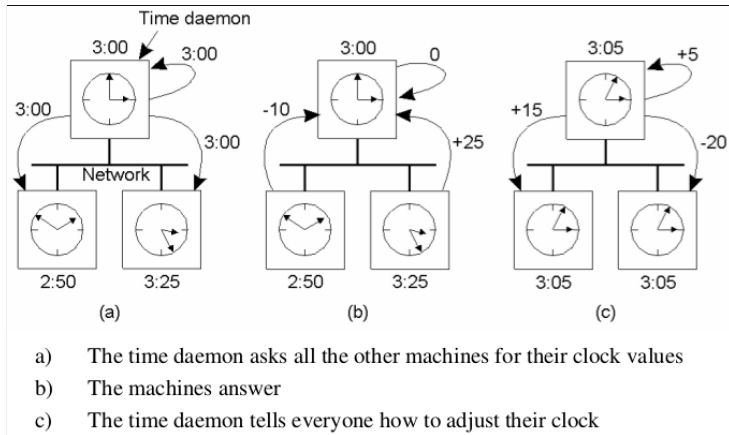
Problemas

- Duración variable del tiempo de transmisión de los mensajes en la red, y del tiempo de respuesta del servidor
- Servidor único: sobrecarga y caída. **Solución** grupo de servidores sincronizado
- Servidor impostor: seguridad **Solución** autenticación del servidor
- Servidor con reloj que falla: tiempo erróneo en el SD **Solución** algoritmo de Berkeley

Sección 5 | Algoritmo de Berkeley

Método Algoritmo de Berkeley

Explicación



Método Algoritmo de Berkeley

Cuestiones

- Ejemplo del algoritmo de *Berkeley*:

<i>Nodo</i>	<i>D (ms)</i>	<i>t (hh:mm:ss.ms)</i>	<i>Desviaciones</i>
<i>N1 (coord.)</i>	0	10:54:23.118	-948 ms
<i>N2</i>	22	10:54:22.236	-1842 ms
<i>N3</i>	26	10:54:24.000	-79 ms
Excluido! <i>N4</i>	190	10:41:46.179	-757983 ms
<i>N5</i>	20	10:54:26.946	+2870 ms

- Calcula el tiempo medio para la sincronización y la desviación a enviar a cada nodo

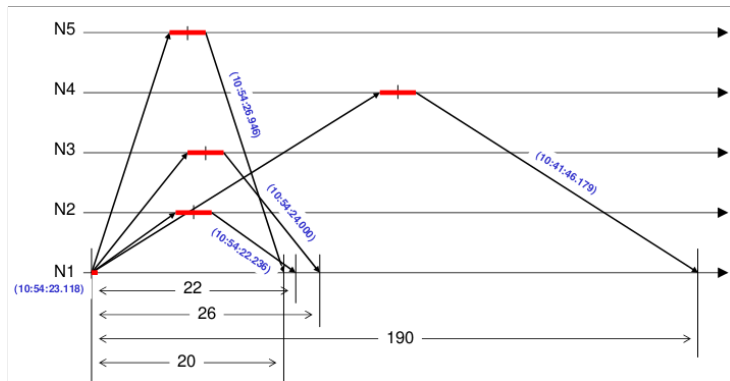
Tiempo medio = 10:54:24.067

- Usando el método de *Cristian*, calcula la precisión en el ajuste de cada nodo al sincronizarse (*min* no se conoce)

D/2

Método Algoritmo de Berkeley

Posibilidades

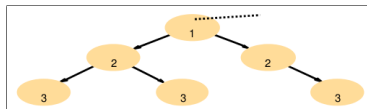


Sección 6 | Network Time Protocol

Network Time Protocol

Explicación

- Estándar en Internet
- Proporciona sincronización redundante con UTC
- Estructurado en capas (strata) de servidores de tiempo
 - servidores primarios (strata 1): referencias UTC fiables
 - servidores de nivel 2 (strata 2): sincronizados con primarios
- Modos de operación:
 - red local con soporte adecuado: modo multicast
 - mayor precisión: modo de llamada a procedimiento
 - mejor sincronización interna: modo simétrico
- La red de servidores se puede reconfigurar



Sección 7 | Tiempo lógico

Tiempo lógico

Introducción

- A veces la precisión obtenida al sincronizar los relojes no nos permite usar el tiempo físico para ordenar los eventos de los diferentes nodos de un sistema distribuido
 - siempre es posible ordenar los eventos de un mismo nodo (si se respeta la monotonidad del reloj)
 - en cambio, relaciones de causalidad entre eventos de nodos diferentes pueden verse distorsionadas
- Muchas aplicaciones requieren únicamente ordenar los eventos (y no tanto conocer el instante exacto en que ocurrieron)
- Reloj lógico: contador de software que se incrementa monótonamente, cuyo valor no necesita estar relacionado con ningún reloj físico.
- Generalmente se asocia a cada proceso un reloj lógico.

Tiempo lógico

Modelo de eventos

- Modelo de sistema: conjunto de procesos que comunican únicamente mediante paso de mensajes
 - enviar (p_i , mensaje)
 - recibir (p_j , mensaje)
 - p_i y p_j son el emisor y el receptor del mensaje
- Simplificación: un proceso por nodo/máquina
- Cada proceso genera una secuencia de eventos. Identificamos tres tipos de eventos:
 - Envío de un mensaje (al ejecutar enviar)
 - Recepción de un mensaje (al ejecutar recibir)
 - Eventos locales/internos (resto de eventos, sin comunicación)

Tiempo lógico

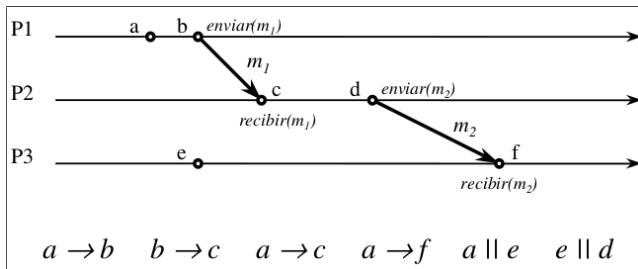
Modelo de eventos

- Para ordenar los eventos de un mismo proceso bastaría con asociar a cada evento x el tiempo local $T(x)$ (si la resolución es suficiente)
- Se dice que existe una **relación de causalidad** entre dos eventos del sistema ($x \rightarrow y$, “ x ha sucedido antes que y ”, “ x happened before y ”) si:
 - 1 x e y son eventos del mismo proceso y $T(x) < T(y)$
 - 2 x e y son los eventos enviar(m) y recibir(m) del mismo mensaje m
 - 3 Existe otro evento z tal que $x \rightarrow z$ y $z \rightarrow y$ (cierre transitivo de la relación)

Tiempo lógico

Concurrentes

Si entre dos eventos no hay relación de casualidad, se dice que son **concurrentes - paralelos**: $x \parallel y$



Sección 8 | Tiempo lógico de Lamport

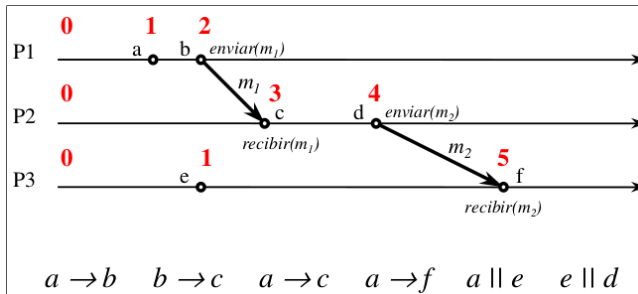
Tiempo lógico de Lamport

Introducción

- Propuesto en 1978, para indicar relaciones de causalidad. Cada proceso P_i tiene su reloj lógico local C_i , para asociar marcas de tiempo a sus eventos (un simple contador asíncrono basta)
- Algoritmo:
 - Inicialmente, $C_i = 0, \forall i$
 - Antes de un evento local o envío de mensajes en p_i : $C_i = C_i + 1$
 - Cuando p_j envía un mensaje m a p_i incluye el valor de su reloj lógico en el mensaje, C_m . Al recibir el mensaje, p_i actualiza su reloj local de la siguiente manera
 - 1 $C_i = \max(C_i, C_m)$
 - 2 $C_i = C_i + 1$
- Problema $C_i(x) < C_j(y)$ no implica $x \rightarrow y$

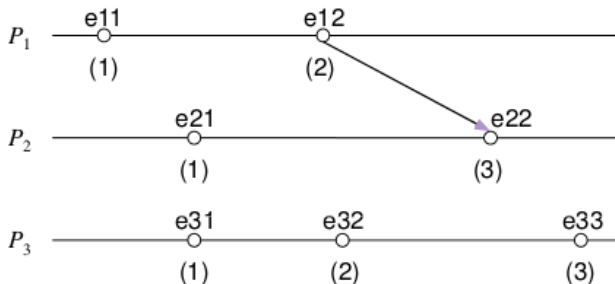
Tiempo lógico de Lamport

Esquema



Tiempo lógico de Lamport

Problemas



$C(e_{11}) < C(e_{22})$, y $e_{11} \rightarrow e_{22}$ es cierto

$C(e_{11}) < C(e_{32})$, pero $e_{11} \rightarrow e_{32}$ es falso

Sección 9 | Vectores de tiempo

Vectores de tiempo

Introducción

Relojes vectoriales

- Son un caso más potente que Lamport.
- Sistemas con comunicaciones entre muchos nodos: Amazon

Concepto intuitivo

- Cada nodo envía no sólo su reloj lógico, si no todos los que tiene constancia.

Vectores de tiempo

Introducción

- Garantizan $V_i(x) < V_j(y) \leftrightarrow x \rightarrow y$
 - V : Vector de N componentes (N =número de procesos)
 - $V_i[i]$: Reloj lógico (local) del proceso P_i
 - $V_i[j]$: Último valor que el proceso P_i conoce del reloj del proceso P_j
- Algoritmo
 - Inicialmente $V_i[j] = 0, \forall i, j$
 - Evento local o envío de mensaje p_i : $V_i[i] = V_i[i] + 1$
 - Cuando p_j envía un mensaje m a p_i , p_j incluye el valor de su vector de tiempos, V_m . Al recibir dicho mensaje, p_i actualiza su vector de la siguiente manera
 - 1 $\forall k: V_i[k] = \max(V_i[k], V_m[k])$
 - 2 $V_i[i] = V_i[i] + 1$

Vectores de tiempo

Introducción

- Definición

- $V1 = V2 \leftrightarrow \forall_i V1[i] = V2[i]$
- $V1 \leq V2 \leftrightarrow \forall_i V1[i] \leq V2[i]$
- $V1 < V2 \leftrightarrow V1 \leq V2 \wedge V1 \neq V2$

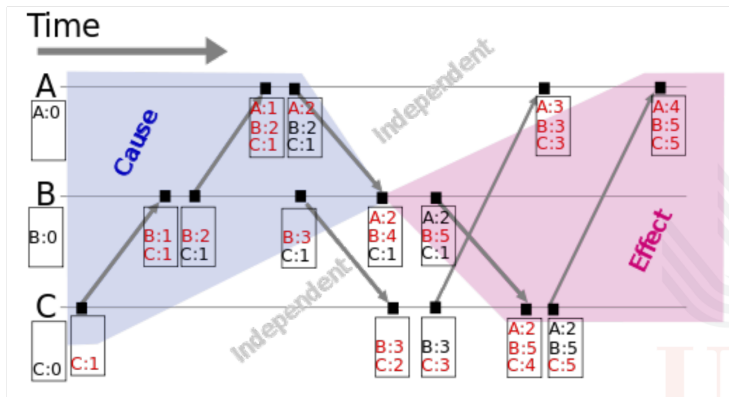
- Existirá relación de causalidad entre dos eventos (x, y) si y sólo si

- $V(x) < V(y)$
- $V(y) < V(x)$

- Si no, los eventos son concurrentes

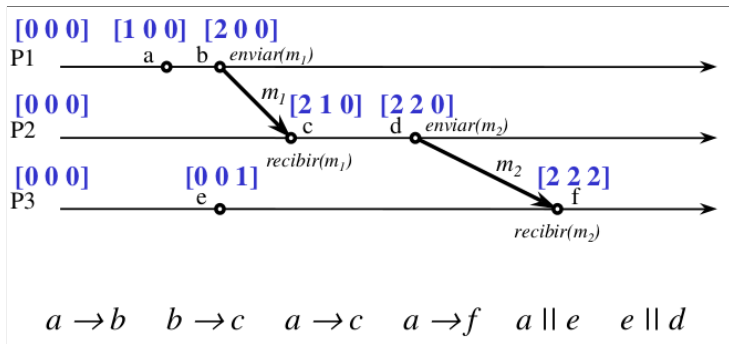
Vectores de tiempo

Ejemplo



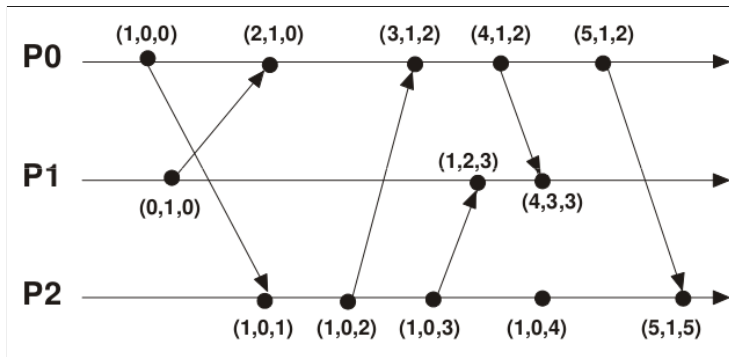
Vectores de tiempo

Esquema



Vectores de tiempo

Esquema



Sección 10 | Tareas