



Unidad 5

Tiempo y estados globales

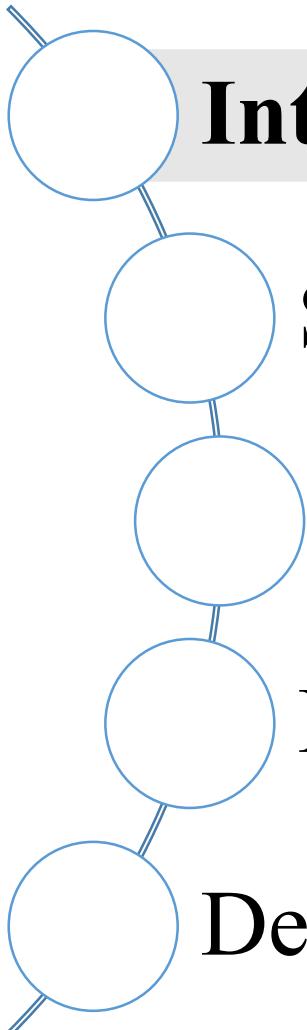
Sistemas Distribuidos

Universidad Nacional de Asunción, Facultad Politécnica

Ingeniería Informática

Ing. Fernando Mancía, Lic. Héctor González

Agenda



Introducción

Sincronización

Tiempos y Relojes Lógicos

Estados Globales

Depuración Distribuida

Time, Clocks, and the Ordering of Events in a Distributed System

Leslie Lamport
Massachusetts Computer Associates, Inc.

Lamport's
paper on
causal ordering
was published
in 1978!

Introducción

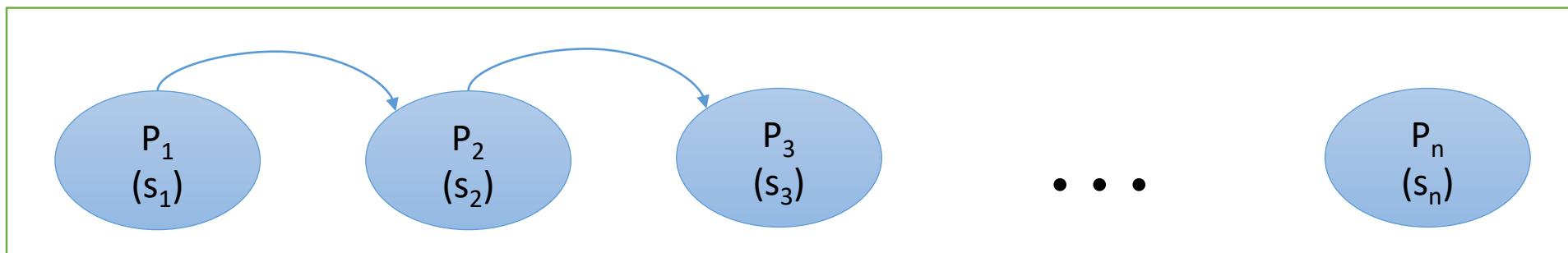
Relojes, eventos y estados de proceso

- Un sistema distribuido consta de una colección Ω de N procesos $i=1, 2, \dots, N$
 - Cada proceso se ejecuta en un único procesador, y los procesadores no comparten memoria.
 - Cada proceso p_i en Ω tiene un estado si que, en general, se transforma a medida que se ejecuta. El estado del proceso (s_i) incluye los valores de todas sus variables.
 - El estado puede incluir también los valores de cualquiera de los objetos en el entorno de su sistema operativo local que lo afecte, como archivos.

Introducción

Relojes, eventos y estados de proceso

- A medida que cada proceso p_i se ejecuta, toma una serie de acciones, cada una de las cuales es un mensaje **Envía** o **Recibe**, o una operación que transforma el estado p_i , que cambia uno o más valores de s_i ,



Relojes, eventos y estados de proceso

- **Evento:** ocurrencia de una única acción que un proceso realiza a medida que se ejecuta.
 - una acción de comunicación
 - una acción de transformación del estado.
- La secuencia de sucesos en un único proceso p_i , puede ser colocada en orden único, que se indica con la relación \rightarrow_i , entre eventos.
 - $e \rightarrow_i e'$ si y sólo si el evento e ocurre antes del e' en p_i .
- **Historia del proceso p_i :** serie de eventos que tienen lugar en el proceso p_i , ordenados de la forma que hemos descrito con la relación \rightarrow_i :
 - $historia(p_i) = h_i = < e_i^o, e_i^1, e_i^2 \dots >$

Relojes, eventos y estados de proceso

- **Relojes:** Los computadores pueden disponer de su propio reloj físico. Estos relojes son dispositivos electrónicos que cuentan las oscilaciones que ocurren en un cristal a una frecuencia definida, y que normalmente dividen esta cuenta y almacenan el resultado en un registro contador.
- El sistema operativo lee el valor del reloj hardware $H_i(t)$ del nodo o proceso p_i , lo escala y añade una compensación para producir un reloj software.
 - $C_i(t) = \alpha H_i(t) + \beta$ que mide aproximadamente el tiempo real, físico t para el proceso p_i .

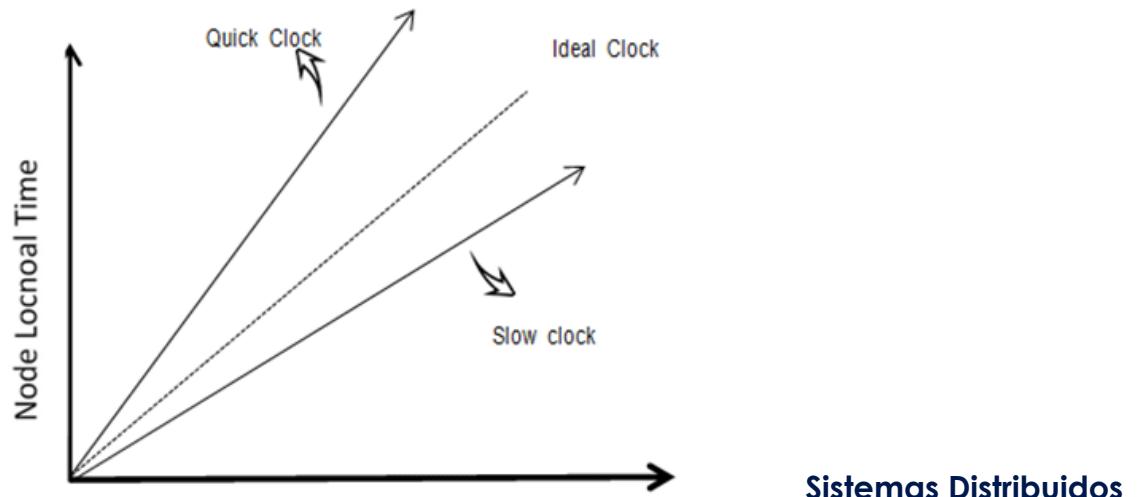
Relojes, eventos y estados de proceso

- Los relojes de los computadores, como los otros, tienden a no estar en perfecto acuerdo.
- **Sesgo:** diferencia instantánea entre las lecturas de dos relojes cualesquiera.
- **Derivas de reloj:** significa que cuentan el tiempo a diferentes ritmos, y por lo tanto divergen. Los relojes basados en cristal utilizados en los computadores están sujetos a la deriva de reloj.



Relojes, eventos y estados de proceso

- **Un ritmo de deriva de reloj** es el cambio en la compensación entre el reloj y un reloj de referencia nominal perfecto por unidad de tiempo, normalmente expresado en partes por millón.
- Para relojes basados en un cristal de cuarzo, suele ser de aproximadamente 10^{-6} segundos/segundo, dando una diferencia de 1 segundo cada 1.000.000 segundos, o 11,6 días. Reloj de cuarzo de alta precisión 10^{-7} o 10^{-8}





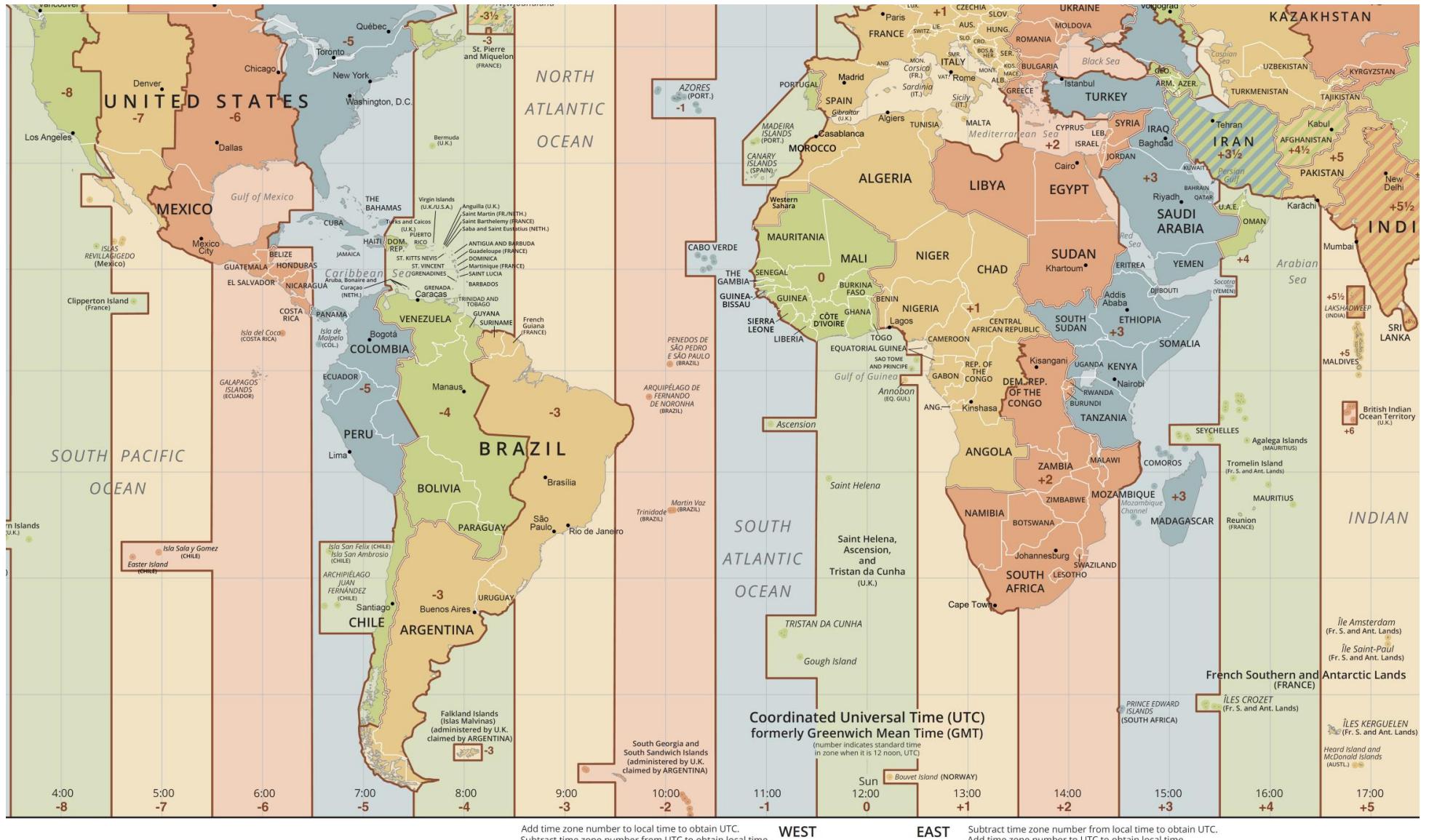
Relojes, eventos y estados de proceso

- Los relojes pueden sincronizarse a fuentes externas de tiempo de gran precisión. Los relojes físicos más precisos utilizan osciladores atómicos cuyo ritmo de deriva es aprox. 10^{-13} .
- La salida de relojes atómicos se utiliza como estándar para el tiempo real transcurrido, conocido como *Tiempo Atómico Internacional*.
- Los segundos, los años y otras unidades de tiempo que nosotros utilizamos están basadas en el tiempo astronómico. Fueron definidos originalmente en términos de la rotación de la tierra sobre su eje y su rotación alrededor del sol.
- Sin embargo, el período de rotación de la tierra sobre su eje va siendo gradualmente más largo. Por lo tanto el tiempo astronómico y el tiempo atómico tienen una tendencia a separarse.

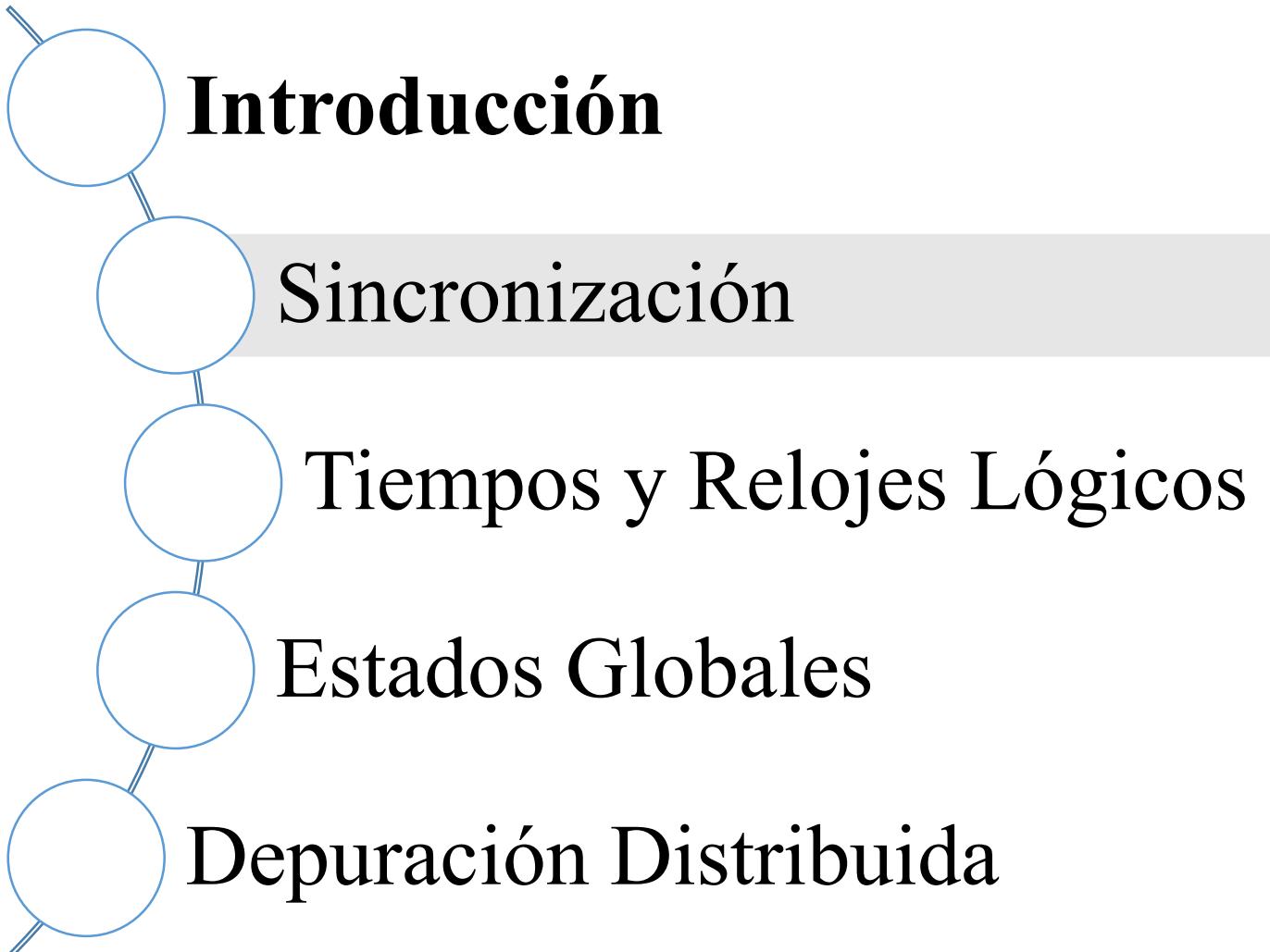
Relojes, eventos y estados de proceso

- ***El Tiempo Universal Coordinado UTC*** es un estándar internacional de cronometraje. Basado en el tiempo atómico, aunque ocasionalmente se inserta un salto de un segundo (o, más raramente, borrado) para mantenerse en sintonía con el tiempo astronómico.
- Las señales UTC se sincronizan y difunden regularmente desde las estaciones de radio terrestres y los satélites, que cubren muchas partes del mundo. Por ejemplo, en USA la estación de radio WWV difunde señales de tiempo en frecuencias de onda corta.
- Señales UTC de la WWV?
 - <https://www.mcodes.org/>

Relojes, eventos y estados de proceso



Agenda



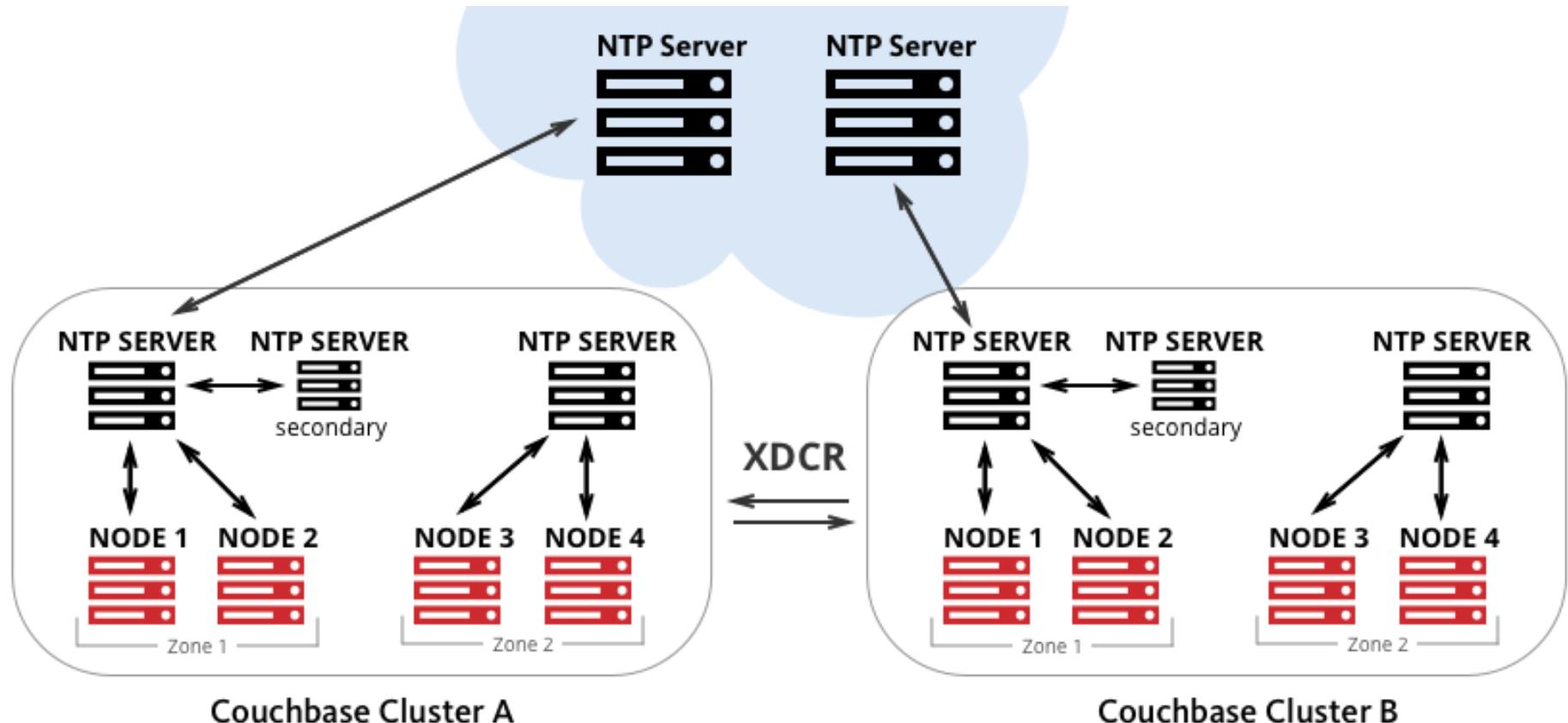
Sincronización de relojes físicos

- Para conocer el tiempo en el que ocurrieron los eventos en un sistema distribuido: es necesario sincronizar los relojes de los procesos.
- ***Sincronización externa***: cuando se utiliza una fuente de tiempo externa autorizada (S_i).
 - $|S_i(t) - C_i(t)| < D$ donde D es un límite conocido; $i=1,2..N$
- ***Sincronización interna***: no existe una fuente de tiempo externa y los nodos (procesos) están sincronizados unos con otros con un grado de precisión conocido.
 - $|C_i(t) - C_j(t)| < D$ donde D es un límite conocido; $i,j=1,2..N$

Sincronización de relojes físicos

- Monotonicidad

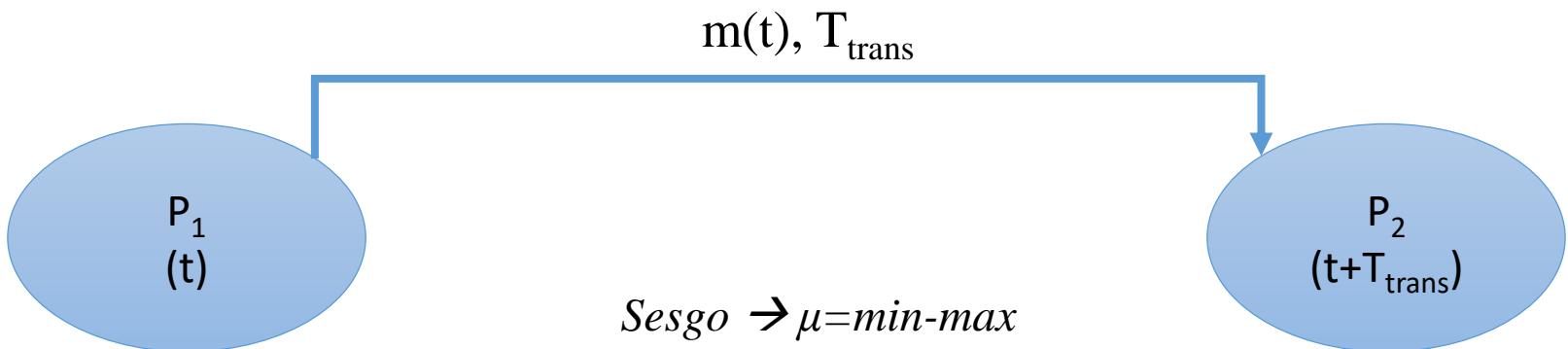
$$t' > t \rightarrow C(t') > C(t)$$



Sincronización de relojes físicos

Sincronización de un sistema síncrono.

- Sistemas síncronos: Conocemos los límites (min-max)
 - Deriva de reloj, retardo de mensajes, tiempo de ejecución de cada paso de un proceso



Sincronización de relojes físicos

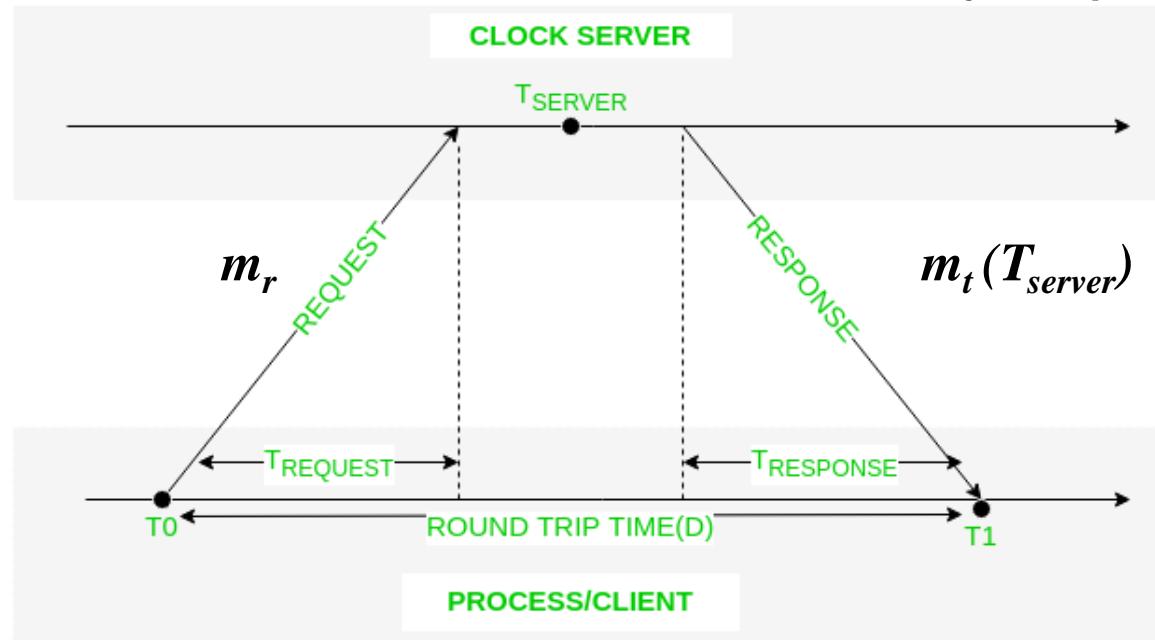
Método de Cristian

- Cristian [1989] sugirió la utilización de un **servidor de tiempo**, conectado a un dispositivo que recibe señales de una fuente de UTC, para sincronizar computadores externamente. Bajo solicitud, el proceso servidor **S** proporciona el tiempo de acuerdo con su reloj.
- Cristian observó que aunque **no hay límite superior** en los retardos de transmisión de mensajes en un sistema asíncrono, los tiempos de ida y vuelta de los mensajes intercambiados entre cada par de procesos son a menudo razonablemente cortos, una pequeña fracción de un segundo.
- El describe el **algoritmo como probabilístico**: el método consigue sincronización sólo si los tiempos de ida y vuelta entre el cliente y el servidor son suficientemente cortos comparados con la precisión requerida.

Sincronización de relojes físicos

Método de Cristian

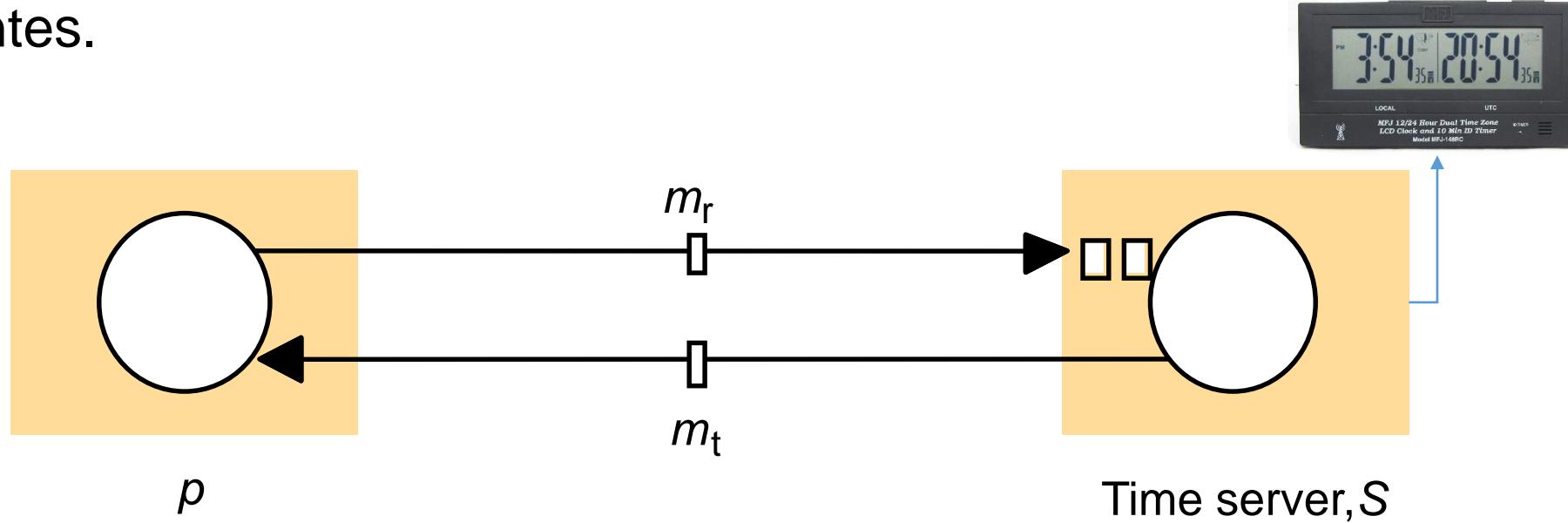
- Un proceso p solicita el tiempo en un mensaje m_r y recibe el valor del tiempo t en un mensaje m_t
- El proceso p registra el tiempo total de ida y vuelta T_{round} tomado para enviar la solicitud m_r y recibir la respuesta m_t . Se puede medir este tiempo con precisión razonable si su ritmo de deriva de reloj es pequeño.



Sincronización de relojes físicos

Método de Cristian

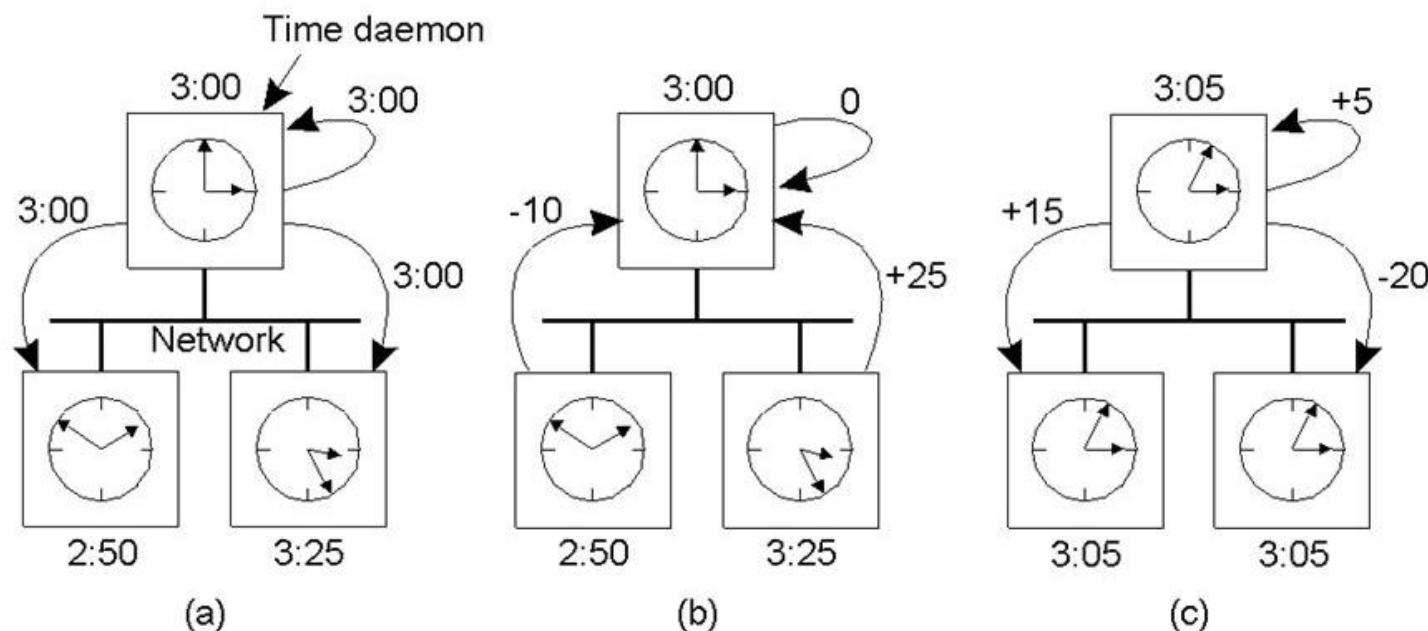
- Una estimación sencilla del tiempo al que p debe fijar su reloj es
 - $t_p = t_{server} + T_{round}/2$,
 - supone que el tiempo transcurrido se desdoble igualmente antes y después de que S coloque t en m_t . Esto es normalmente una suposición razonable de precisión, a menos que los dos mensajes sean transmitidos sobre redes diferentes.



Sincronización de relojes físicos

Algoritmo de Berkeley

- No existe servidor de tiempo, el algoritmo elige un nodo como **master** y los demás serán **slaves**.



- The time daemon asks all the other machines for their clock values
- The machines answer
- The time daemon tells everyone how to adjust their clock

Sincronización de relojes físicos

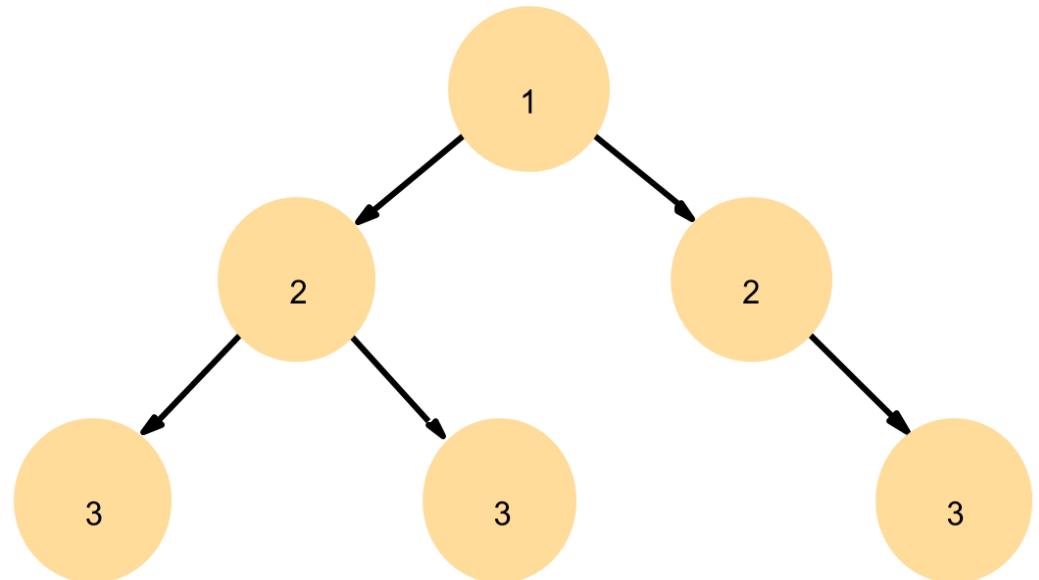
El protocolo de tiempo de red - NTP

- El Protocolo de Tiempo de Red (Network Time Protocol, NTP) define una arquitectura para un servicio de tiempo y un protocolo para distribuir la información sobre Internet.
 - Cristian y Berkeley → intranets
 - NTP → internet
- Los objetivos y las metas principales de diseño de NTP son los siguientes.
 - Proporcionar un servicio que permita a los clientes a lo largo de Internet estar sincronizados de forma precisa a UTC.
 - Proporcionar un servicio fiable que pueda sobrevivir a pérdidas largas de conectividad.
 - Permitir a los clientes re sincronizar con suficiente frecuencia para compensar las tasas de deriva encontradas en la mayoría de los computadores.
 - Proporcionar protección contra la interferencia con el servicio de tiempo, ya sea maliciosa o accidental.

Sincronización de relojes físicos

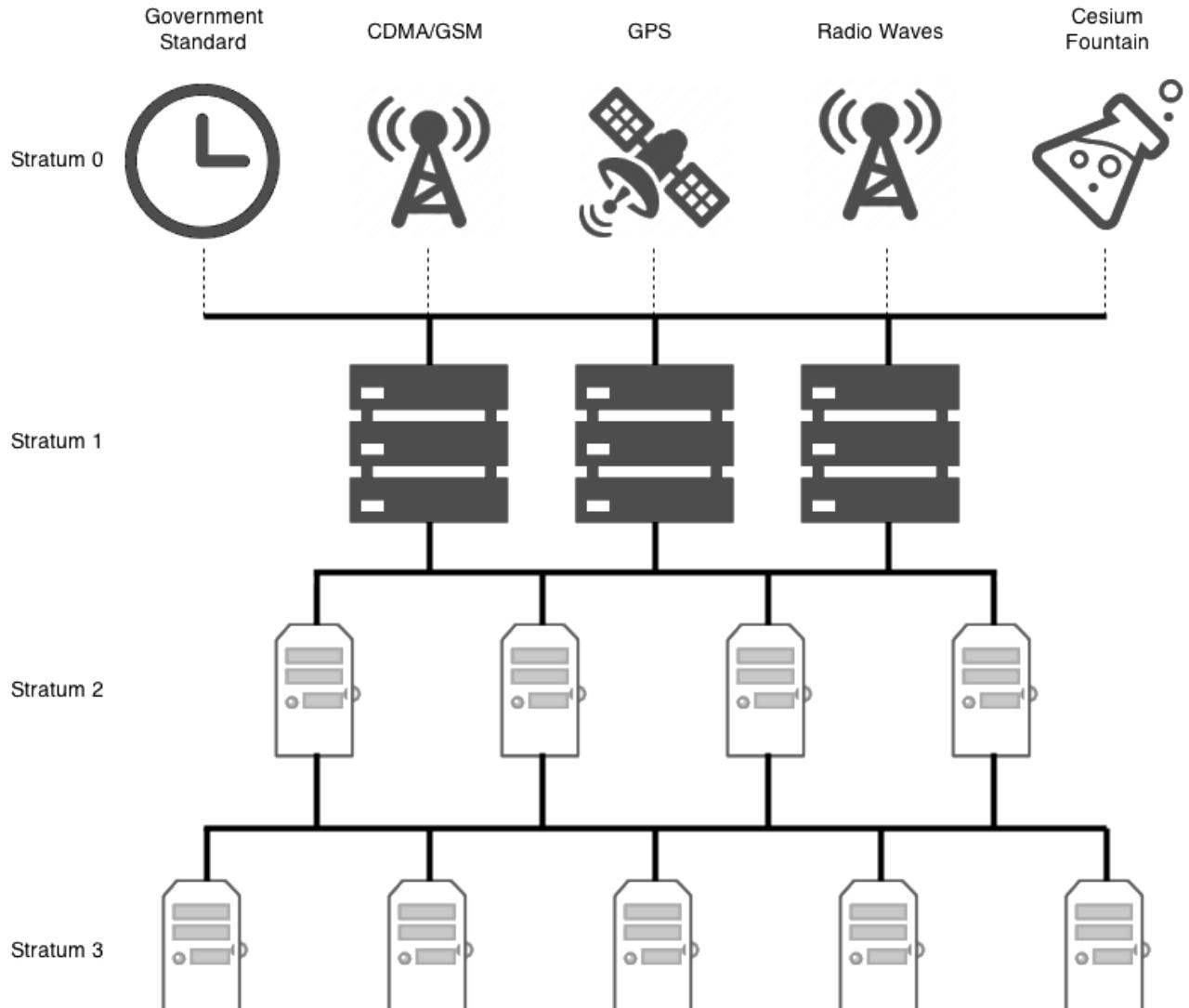
El protocolo de tiempo de red - NTP

- Los servidores están conectados en una jerarquía lógica llamada una subred de sincronización, cuyos niveles se llaman estratos.
 - **Estrato 1:** Servidores primarios, en la raíz.
 - **Estrato 2:** Servidores secundarios que están sincronizados directamente con los servidores primarios.
 - **Estrato 3:** Servidores que están sincronizados con los del estrato 2, y así sucesivamente.
 - Los servidores del nivel más bajo (**hojas**) se ejecutan en las estaciones de trabajo de los usuarios.



Sincronización de relojes físicos

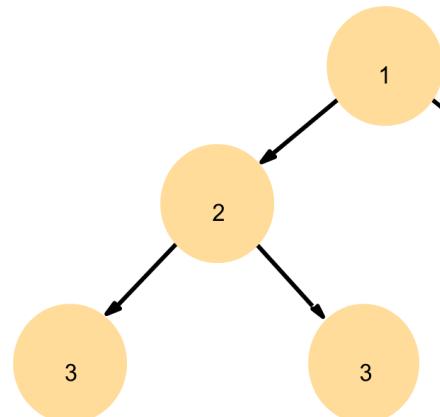
Subred de sincronización- NTP



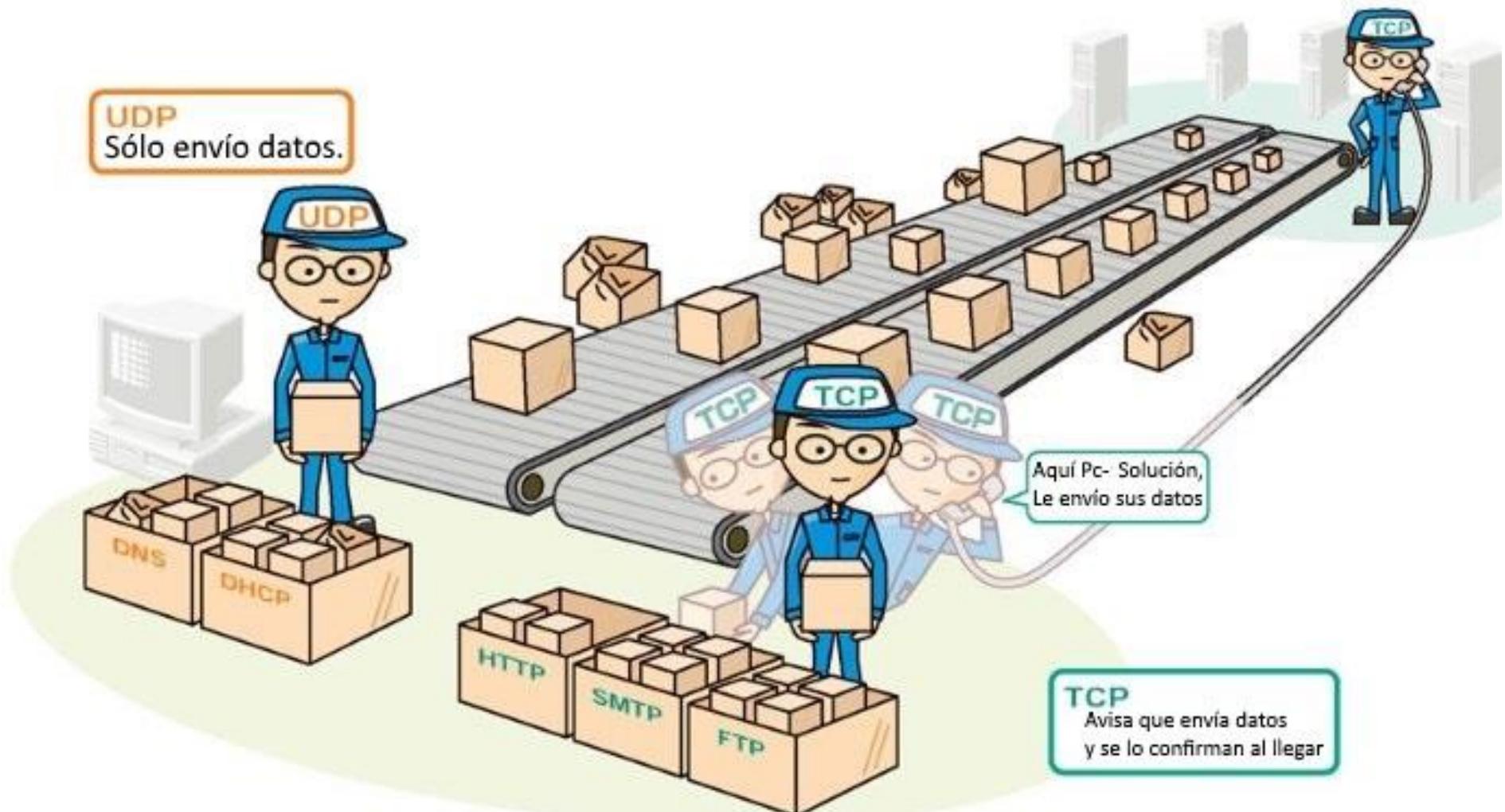
Sincronización de relojes físicos

El protocolo de tiempo de red - NTP

- Los servidores NTP se sincronizan entre sí en uno de estos tres modos: multidifusión, llamada a procedimiento y modo simétrico.
 - **Multidifusión:** está pensado para su uso en una LAN de alta velocidad. Uno o más servidores reparten periódicamente el tiempo a los servidores que se ejecutan en otros computadores conectados en la LAN, que fijan sus relojes suponiendo un pequeño retardo.
 - **Llamada a procedimiento** es similar al funcionamiento del algoritmo de Cristian. Un servidor acepta solicitudes de otros computadores, que el procesa respondiendo con su marca de tiempo (lectura actual del reloj).
 - **Simétrico:** esta aplicado en los estratos mas bajos (niveles mas altos). Se usa entre servidores NTP para sincronizarse entre sí, como mecanismo de respaldo cuando no pueden alcanzar el servidor NTP (externo).
- En todos los modos, los mensajes se entregan de modo no fiable, utilizando el protocolo de transporte estándar Internet UDP

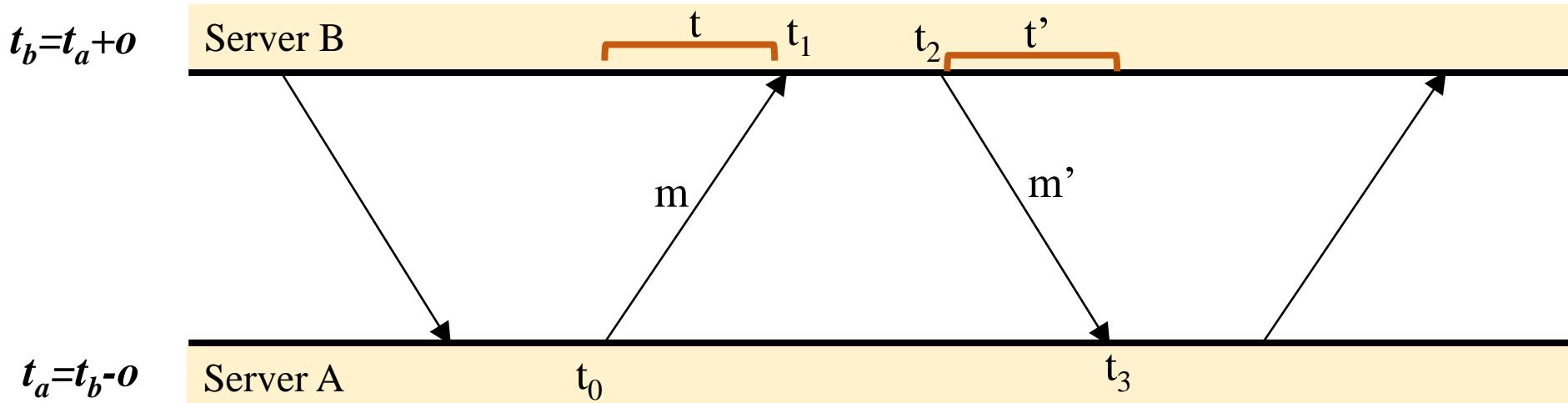


TCP vs UDP



Sincronización de relojes físicos

NTP Simétrico



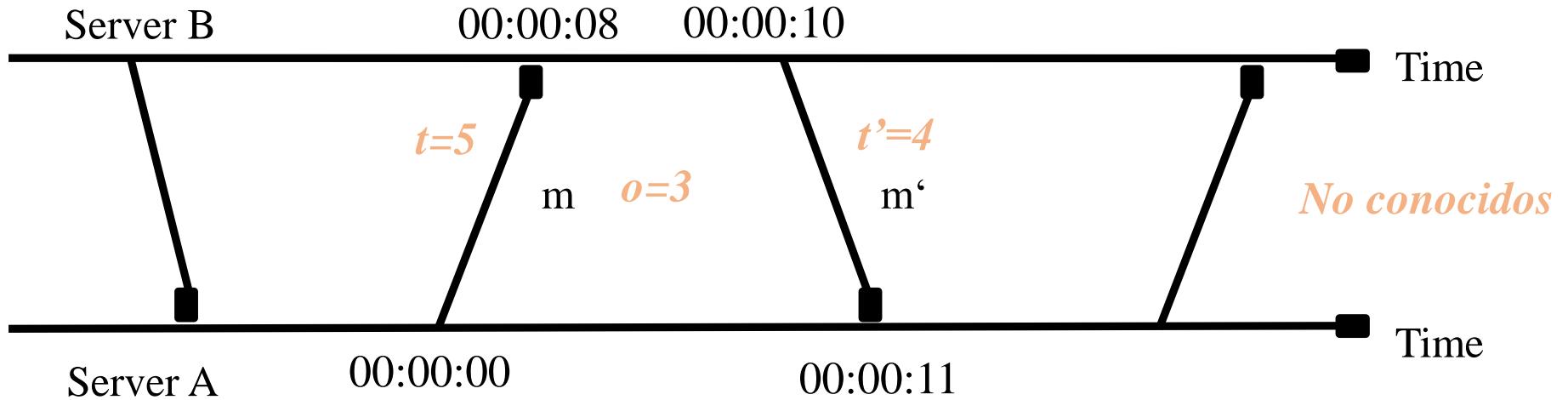
- Los mensajes llegan con un retardo de transmisión d (t y t') y con desplazamiento de B respecto a A (offset o)

$$o_i - \frac{d_i}{2} \leq o \leq o_i + \frac{d_i}{2}$$

- Retardo:** $d_i = t + t' = t_1 - t_0 + t_3 - t_2$
- Offset:** $o_i = (t_1 - t_0 + t_2 - t_3)/2$

Sincronización de relojes físicos

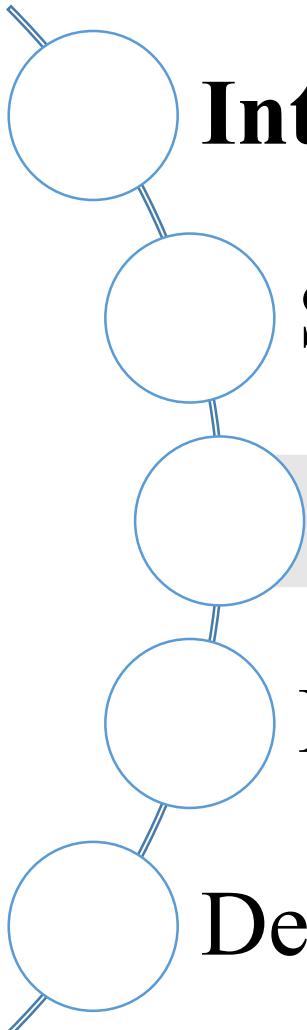
NTP simétrico



- $o_i = \frac{8-1}{2} = 3,5$
- $d_i = \frac{8+1}{2} = 4,5$
- $3,5 - \frac{4,5}{2} \leq o \leq 3,5 + \frac{4,5}{2}$

$$1,25 \leq o \leq 5,75$$

Agenda



Introducción

Sincronización

Tiempos y Relojes Lógicos

Estados Globales

Depuración Distribuida

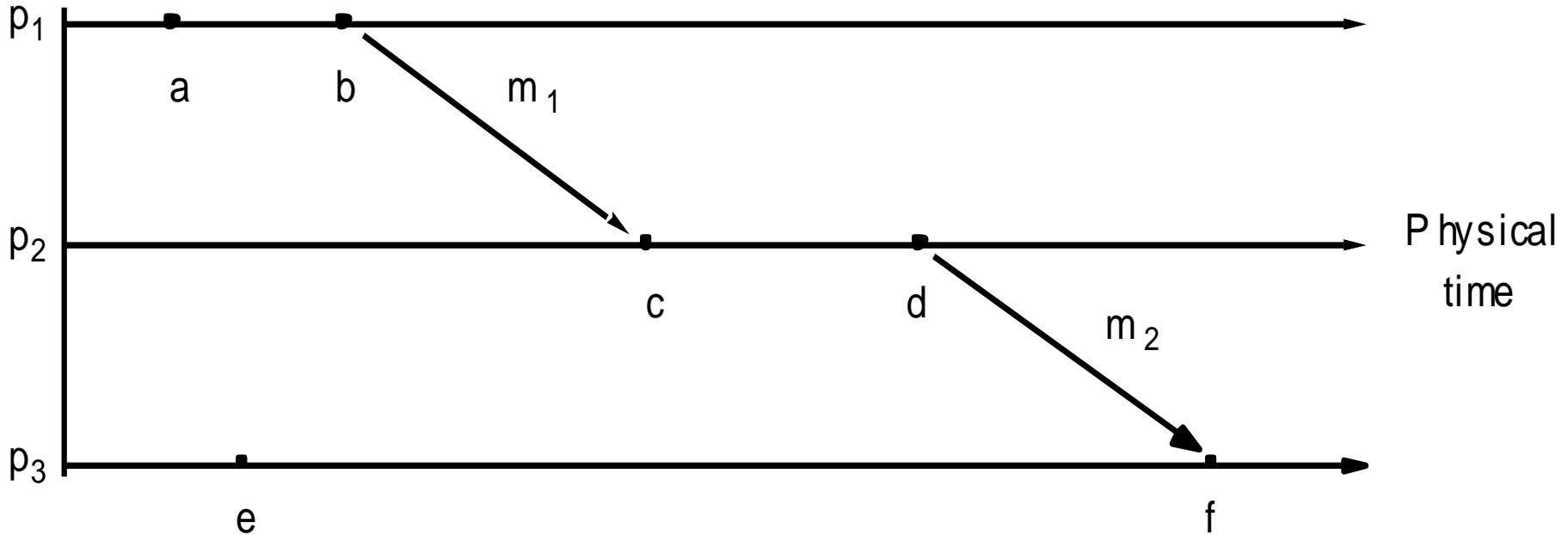
Tiempo lógico y relojes lógicos

- Desde el punto de vista de un único proceso, los sucesos están ordenados de forma única por los tiempos mostrados en el reloj lógico.
- Sin embargo, como apuntó Lamport [1978], puesto que no podemos sincronizar perfectamente los relojes a lo largo de un sistema distribuido, no podemos usar, en general, el tiempo físico para obtener el orden de cualquier par arbitrario de sucesos que ocurran en él.
- En general, podemos utilizar un esquema que es similar a la causalidad física, pero que se aplica en los sistemas distribuidos, para ordenar algunos de los sucesos que ocurren en diferentes procesos.

Tiempo lógico y relojes lógicos

- Esta ordenación está basada en dos puntos sencillos e intuitivamente obvios:
 - Si dos sucesos han ocurrido en el mismo proceso $p_i (i = 1, 2, \dots, N)$, entonces ocurrieron en el orden en el que les observa p_i , éste es el orden \rightarrow_i que hemos definido anteriormente.
 - Cuando se envía un mensaje entre procesos, el suceso de enviar el mensaje ocurrió antes del de recepción del mismo.
- Lamport llamó a la ordenación parcial obtenida al generalizar estas dos relaciones la realización suceder antes. Esto también se conoce a veces como la **relación de orden causal o ordenación causal potencial**.
- Reglas:
 - SA1: $\exists p_i: e \rightarrow_i e' \text{ entonces } e \rightarrow e'$
 - SA2: $\forall m, \text{envia}(m) \rightarrow \text{recibe}(m)$
 - SA3: Si $e, e' y e''$ son sucesos tal que $e \rightarrow e' y e' \rightarrow e''$, entonces $e \rightarrow e''$

Tiempo lógico y relojes lógicos



- SA1 => $p_1: a \rightarrow_1 b = \text{a} \rightarrow \text{b}$
- SA2 => $b = \text{envía}(m_1); c = \text{recibe } (m_1) = \text{b} \rightarrow \text{c}$ **¿Y que pasa con a y e?**
- SA3 => $\text{b} \rightarrow \text{c} \rightarrow \text{d} \rightarrow \text{f} = \text{b} \rightarrow \text{f}$

Tiempo lógico y relojes lógicos

Reloj lógico

- Lamport inventó un mecanismo simple por el que la relación “**sucedió antes**” puede capturarse numéricamente, denominado **reloj lógico**. Un reloj lógico de Lamport es un contador software que se incrementa monótonamente, cuyos valores no necesitan tener ninguna relación particular con ningún reloj físico.
- Cada proceso p_i mantiene su propio reloj lógico, L_i , que él utiliza para aplicar las llamadas marcas de tiempo de Lamport a los sucesos.
- Representamos la marca de tiempo del suceso e en p_i por $L_i(e)$, y representamos por $L(e)$ la marca de tiempo del suceso e cualquiera que sea el proceso en el que ocurrió.

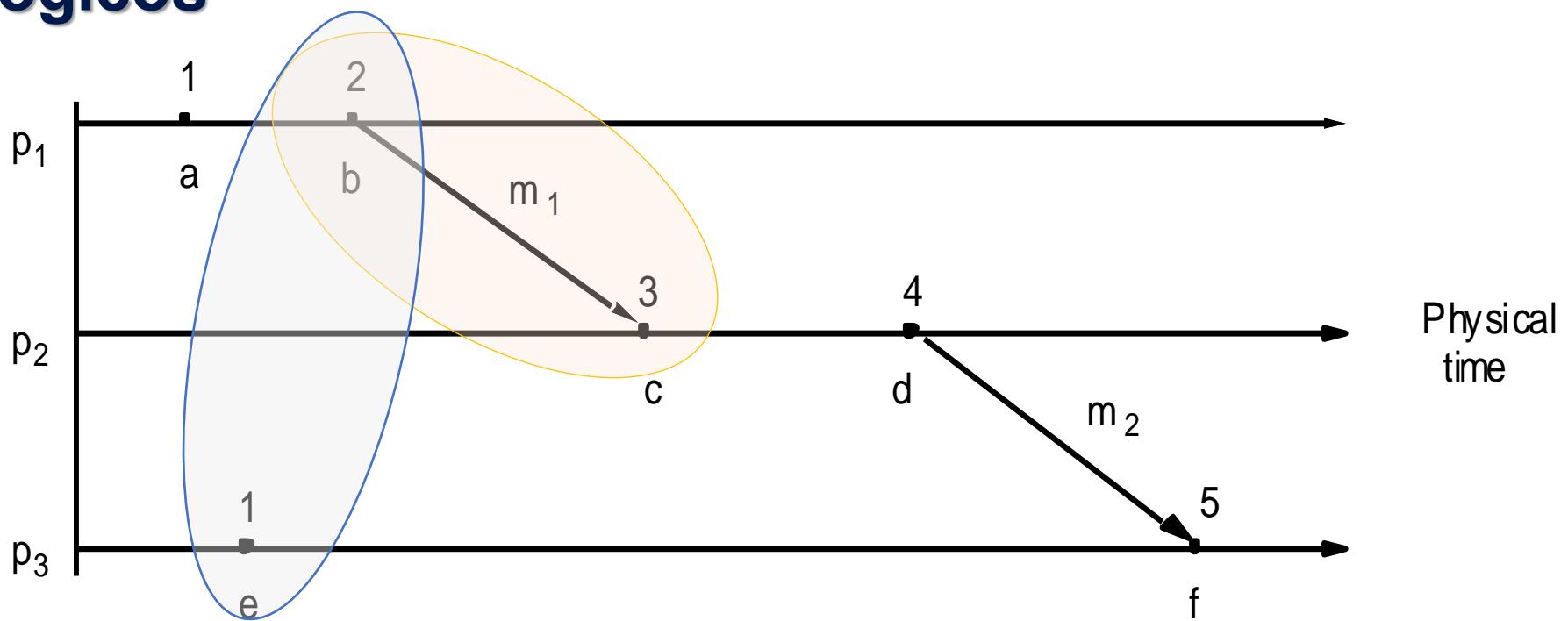
Tiempo lógico y relojes lógicos

Relojes lógicos

- Para capturar la relación “sucedió antes” →, los procesos actualizan sus relojes lógicos y transmiten los valores de sus relojes lógicos en mensajes como sigue:
- RL1: L_i se incrementa antes de emitir cada suceso en el proceso p_i : $L_i = L_i + 1$
- RL2:
 - (a) Cuando un proceso p_i envía un mensaje m , acarrea en m el valor de $t = L_i$
 - (b) Al recibir (m, t) , cada proceso p_i calcula $L_j = \max(L_j, t)$ y entonces aplica RL1 antes de realizar la marca de tiempo del suceso **recibe(m)**.

Tiempo lógico y relojes lógicos

Reloj lógicos



- El incremento puede ser con cualquier otro número
- Por inducción: $b \rightarrow c \Rightarrow L(b) < L(c)$
- Pero si $L(e) < L(e')$ no podemos inferir que $e \rightarrow e'$

Tiempo lógico y relojes lógicos

Relojos lógicos totalmente ordenados

- Algunos pares de sucesos distintos, generados por diferentes procesos, tienen marcas de tiempo de Lamport numéricamente idénticas.
- Podemos crear un orden total sobre los sucesos, esto es, uno para el que todos los pares de sucesos distintos están ordenados, teniendo en cuenta los identificadores de los procesos en los que ocurren los sucesos.
 - Si e es un suceso que ocurre en p_i con marca de tiempo local T_i .
 - Si e' es un suceso que ocurre en p_j con marca de tiempo local T_j .
- Definimos las marcas de tiempo globales para esos sucesos como
- (T_i, i) y (T_j, j) respectivamente. y definimos $(T_i, i) \leq (T_j, j)$ si y solo si:
 - $T_i < T_j$ o $T_i = T_j$ siendo $i < j$
- Relojos totalmente ordenados del ejemplo anterior.

$$H = \langle a_1^1, e_3^1, b_1^2, c_2^3, d_2^4, f_3^5 \rangle$$

Tiempo lógico y relojes lógicos

Reloj vectoriales

- Mattern [1989] y Fidge [1991] desarrollaron relojes vectoriales para vencer la deficiencia de los relojes de Lamport: del hecho que $L(e) < L(e')$ no podemos deducir que $e \rightarrow e'$.
- Un reloj vectorial para un sistema de N procesos es un vector de N enteros.
- Cada proceso mantiene su propio reloj vectorial V_i , que utiliza para colocar marcas de tiempo en los sucesos locales.
- Como las marcas de tiempo de Lamport, cada proceso adhiere el vector de marcas de tiempo en los mensajes que envía al resto, y hay unas reglas sencillas para actualizar los relojes.

Tiempo lógico y relojes lógicos

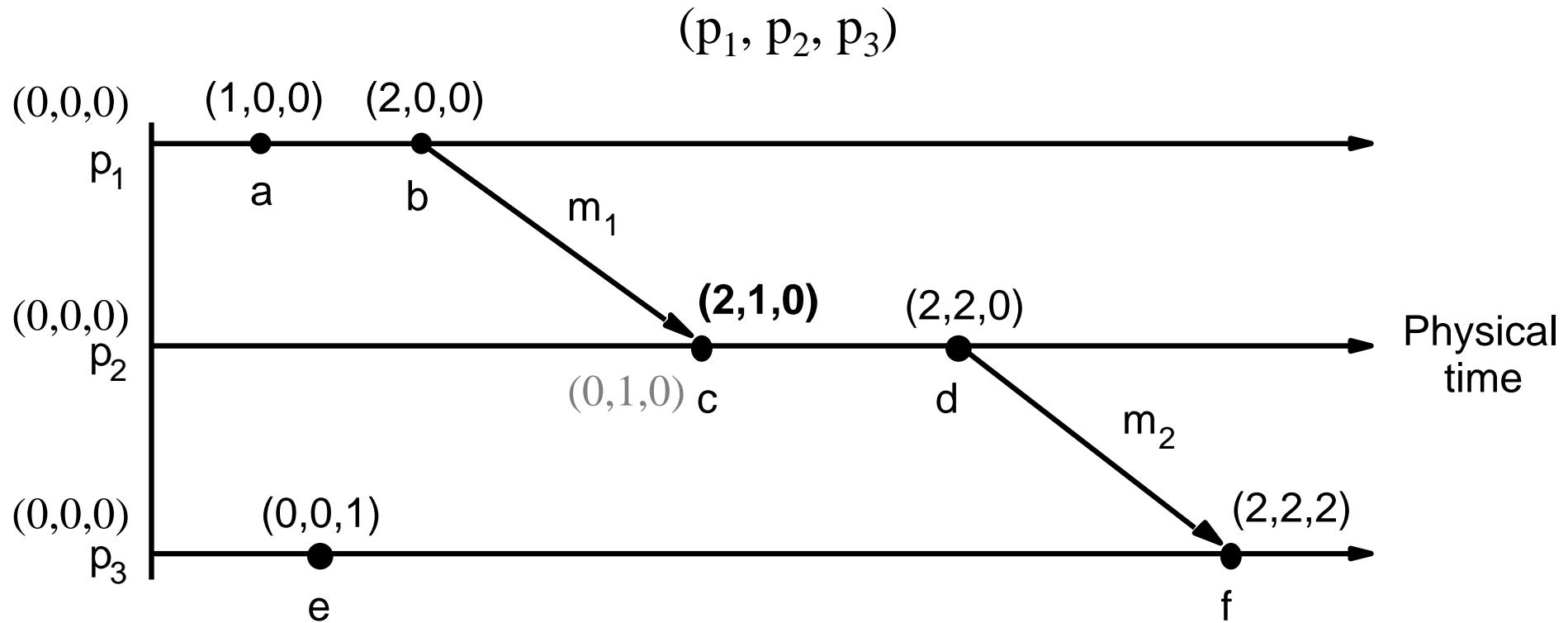
Relojes vectoriales

Reglas de los relojes vectoriales

- RV1: $V_i[j] = 0$ para $i, j = 1, 2, \dots, N$.
- RV2: Justo antes que p_i coloque una marca de tiempo en un suceso, coloca $V_i[i] = V_i[i] + 1$
- RV3: p_i incluye el valor $t = V_i$ en cada mensaje que envía.
- RV4: Cuando p_i recibe una marca de tiempo y en un mensaje, establece $V_i[j] = \max(V_i[j], t[j])$, para $j = 1, 2, \dots, N$. Esta operación de mezcla toma el vector máximo entre dos, componente a componente.
- Sea $V(e)$ el vector de marcas de tiempo aplicadas por el proceso en el que ocurre e . Es sencillo mostrar, por inducción sobre la longitud de cualquier secuencia de sucesos que relacione los sucesos e y e' , que $e \rightarrow e' \Rightarrow V(e) < V(e')$

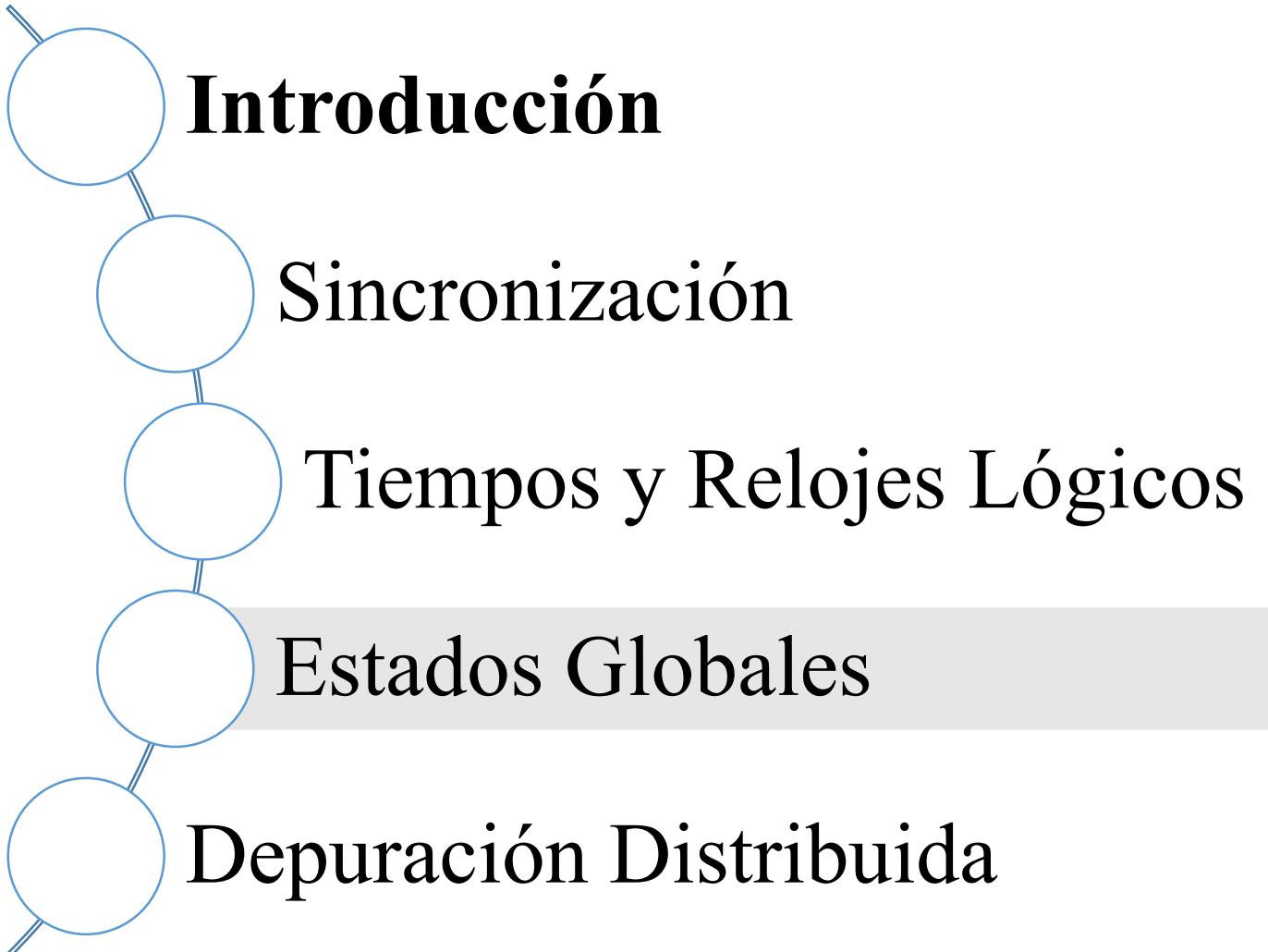
Tiempo lógico y relojes lógicos

Relojas vectoriales



- Por inducción: $a \rightarrow f \Rightarrow V(a) < L(f)$ y también podemos inferir lo inverso $V(a) < L(f) \Rightarrow a \rightarrow f$
- No podemos verificar $V(e) \leq V(c)$ ni $V(c) \leq V(e)$, por tanto $e \parallel c$

Agenda



Estados globales

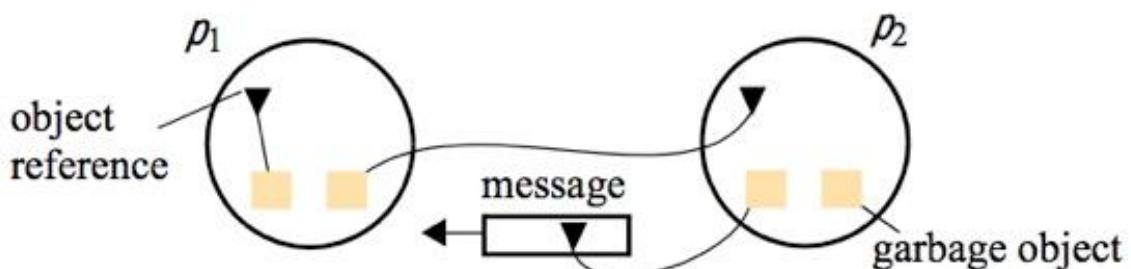
- ***Compactación automática de memoria:***

- Un objeto se considera desecharable si no hay posteriormente ninguna referencia a él desde cualquier parte del sistema distribuido. La memoria ocupada por el objeto puede ser reclamada, una vez que se sabe que éste es desecharable. Para comprobar, debemos verificar que no hay referencias a él en cualquier parte del sistema.

Estados globales

- **Compactación automática de memoria:**

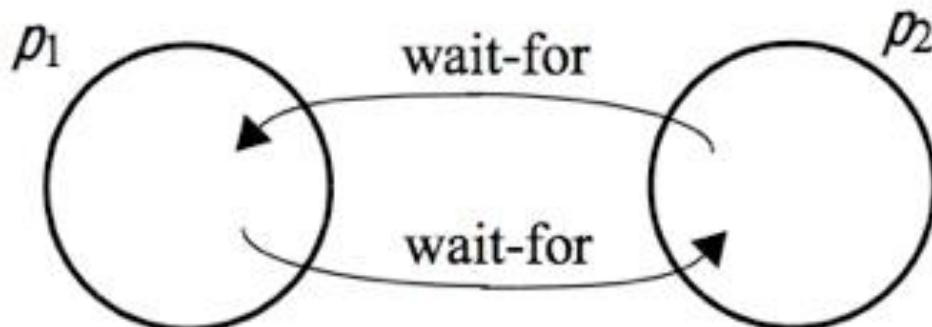
- El proceso p_1 tiene dos objetos, ambos con referencias, uno tiene una referencia al propio p_1 y el otro tiene una referencia a p_2 .
- El proceso p_2 tiene un objeto desecharable, al que no existe ninguna referencia en el sistema. Hay también otro objeto en p_2 al que ni p_1 ni p_2 tienen acceso, pero hay una referencia a él en un mensaje que transita entre ambos. Esto nos muestra que cuando consideramos las propiedades de un sistema, debemos incluir el estado de los canales de comunicación así como el de los procesos.



Estados globales

- **Detección distribuida de bloqueos indefinidos:**

- Un bloqueo indefinido distribuido ocurre cuando cada uno de los procesos de una colección espera por otro proceso para enviarle un mensaje, y donde hay un ciclo en el grafo de esta relación espera por.
- En la Fig. se muestra que cada uno de los procesos p_1 y p_2 espera un mensaje del otro, por lo que el sistema nunca progresará.



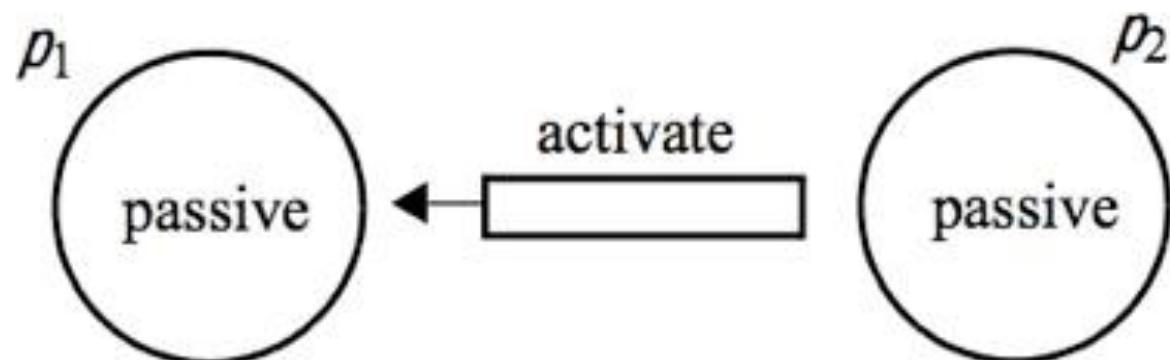
Estados globales

- **Detección de la terminación distribuida:**
 - El problema aquí es detectar que un algoritmo distribuido ha terminado. La detección de la terminación es un problema que parece muy fácil de resolver: al principio parece sólo necesario comprobar si cada proceso se ha detenido.
 - Para ver que esto no es así, consideremos un algoritmo distribuido ejecutado por dos procesos p_1 y p_2 cada uno de los cuales puede necesitar valores del otro.

Estados globales

- **Detección de la terminación distribuida:**

- Instantáneamente, podemos encontrar que un proceso es activo o pasivo, un proceso pasivo no está ligado a ninguna actividad por sí mismo pero está preparado para responder con un valor solicitado por el otro.
- Supongamos que descubrimos que p_1 y p_2 están pasivos. ¿Podemos afirmar que el algoritmo ha terminado?



Estados globales

Cortes consistentes

- En principio, es posible observar la sucesión de estados de un proceso individual, pero cómo establecer un estado global del sistema, el estado de la colección de procesos es más difícil de tratar.
- El problema esencial es la ausencia de **tiempo global**. Si todos los procesos tuvieran los relojes perfectamente sincronizados ellos podrían estar de acuerdo en un tiempo en el que cada proceso debía registrar su estado, el resultado sería un estado global actual del sistema.
 - Si es así, podremos decir, por ejemplo, qué procesos estaban bloqueados indefinidamente, cuales objetos son desecharables, etc.
 - No podemos tener sincronización perfecta de los relojes, por lo que este método no es posible.

Estados globales

Cortes consistentes

- Entonces... ¿podríamos tener un estado global significativo con los estados locales, aún cuando no tengamos sincronización perfecta?



Estados globales

Cortes consistentes

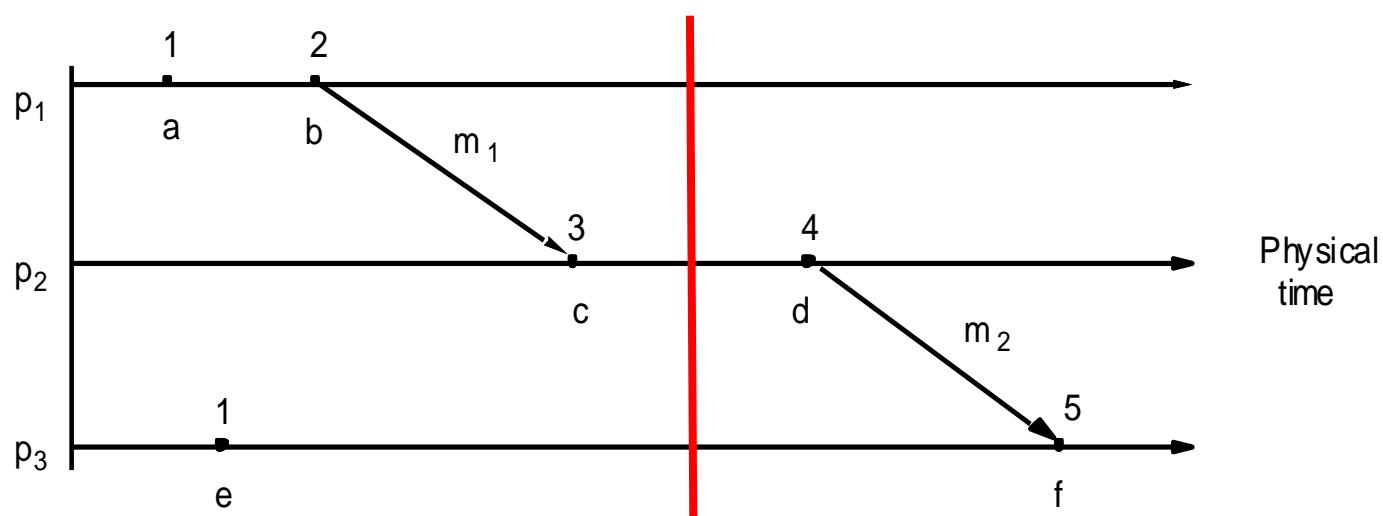
- La respuesta es SI, pero necesitamos algunas definiciones:
- Ya habíamos definido la historia de un proceso
 - $historia(p_i) = h_i = \langle e_i^0, e_i^1, e_i^2 \dots e_i^k \dots \rangle$
- Podemos formar la historia global de Ω como la unión de las historias individuales de los procesos:
 - $H = h_1 \cup h_2 \cup h_3 \dots \cup h_n$
- Matemáticamente, podemos tomar cualquier conjunto de estados de los procesos individuales para formar un estado global $S=(s1, s2 \dots sn)$.
- Pero qué estados son significativos; esto es, ¿cuál de los estados de los procesos podrían haber ocurrido al mismo tiempo? Un estado global corresponde a los prefijos iniciales de las historias individuales de los procesos.

Estados globales

Cortes consistentes

- Un corte de la ejecución del sistema es un subconjunto de su historia global que es la unión de los prefijos de las historias de los procesos:

- $C = h_1^{c_1} \cup h_2^{c_2} \cup h_3^{c_3} \dots \cup h_n^{c_n}$



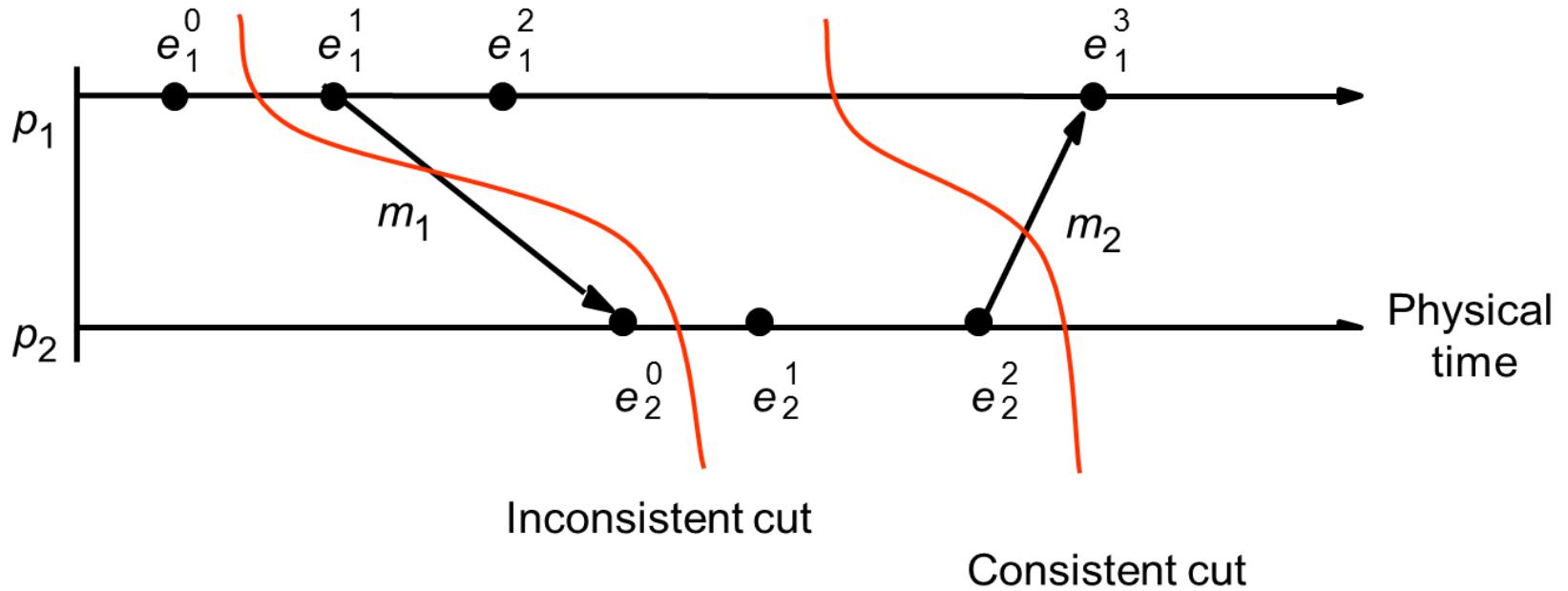
$$\begin{aligned} h_1^{c_1} &= < a, b > \\ h_2^{c_2} &= < c > \\ h_3^{c_3} &= < e > \end{aligned}$$

- El estado s_i en el estado global S correspondiente al corte C es el de p_i inmediatamente después del último suceso procesado por p_i en el corte, $\{e_i^{c_i} : i = 1, 2, 3 \dots n\}$. Este conjunto de se llama la **frontera del corte**.
 - La frontera de corte en la fig. es $\langle b, c, e \rangle$

Estados globales

Cortes consistentes

- Ejemplos: Los procesos p_1 y p_2 de la figura muestra dos cortes.



- Un corte consistente incluye los eventos de la frontera de corte y por cada evento de la frontera incluye todos los eventos que sucedieron antes de ese evento.
- Un estado global consistente es uno que corresponde con un corte consistente. Podemos caracterizar un sistema distribuido como una serie de transiciones entre los estados globales del sistema:
- $S_0 \rightarrow S_1 \rightarrow S_2 \dots$

Estados globales

Algoritmo de instantánea de Chandy y Lamport

- Chandy y Lamport [1985] describen un algoritmo de instantánea para determinar estados globales de sistemas distribuidos.
- El objetivo es **registrar un conjunto de estados de procesos y canales** (una instantánea) para un conjunto de procesos $p_i (i=1,2,\dots,N)$ tal que, incluso a través de una combinación de estados registrados que nunca podrían haber ocurrido al mismo tiempo, el **estado global registrado sea consistente**.
- Veremos que el estado que registra el algoritmo de instantánea tiene propiedades convenientes para evaluar predicados del estado global.
- El algoritmo registra el **estado localmente en los procesos**; no proporciona un método para recoger el estado global en un sitio.

Estados globales

Algoritmo de instantánea de Chandy y Lamport

- El algoritmo supone que:
 - No fallan ni los canales ni los procesos; la comunicación es fiable por lo que cada mensaje enviado es recibido intacto, exactamente una vez.
 - Los canales son unidireccionales y proporcionan la entrega de los mensajes con ordenación FIFO.
 - El grafo de los procesos y canales está fuertemente conectado (hay un recorrido entre dos procesos cualquiera).
 - Cualquier proceso puede iniciar una instantánea global en cualquier instante.
 - Los procesos pueden continuar su ejecución y enviar y recibir mensajes normales mientras tiene lugar la instantánea.

Estados globales

Algoritmo de instantánea de Chandy y Lamport

- El algoritmo se define mediante de dos reglas:
 - ***La regla de recepción del marcador:*** obliga a los procesos a enviar un marcador después de haber registrado su estado, pero antes de que envíen cualquier otro mensaje.
 - ***La regla de envío del marcador:*** obliga a un proceso que no ha registrado su estado a hacerlo. En ese caso, éste es el primer marcador que ha recibido. Él observará qué mensajes llegan posteriormente en los otros canales entrantes. Cuando un proceso que ya ha guardado su estado recibe un marcador (en otro canal), registra el estado de ese canal bajo la forma del conjunto que recibió desde que guardó su estado.
- Cualquier proceso puede comenzar el algoritmo en cualquier instante. Actúa como si hubiera recibido un marcador y sigue la regla de recepción del marcador. Por tanto registra su estado y comienza a registrar los mensajes que llegan sobre todos los canales entrantes.

Estados globales

Algoritmo de instantánea de Chandy y Lamport

Marker receiving rule for process p_i ,

On p_i 's receipt of a *marker* message over channel c :

if (p_i has not yet recorded its state)

it records its process state now;

records the state of c as the empty set;

turns on recording of messages arriving over other incoming channels;

else

p_i records the state of c as the set of messages it has received over c since it saved its state.

end if

Marker sending rule for process p_i ,

After p_i has recorded its state, for each outgoing channel c :

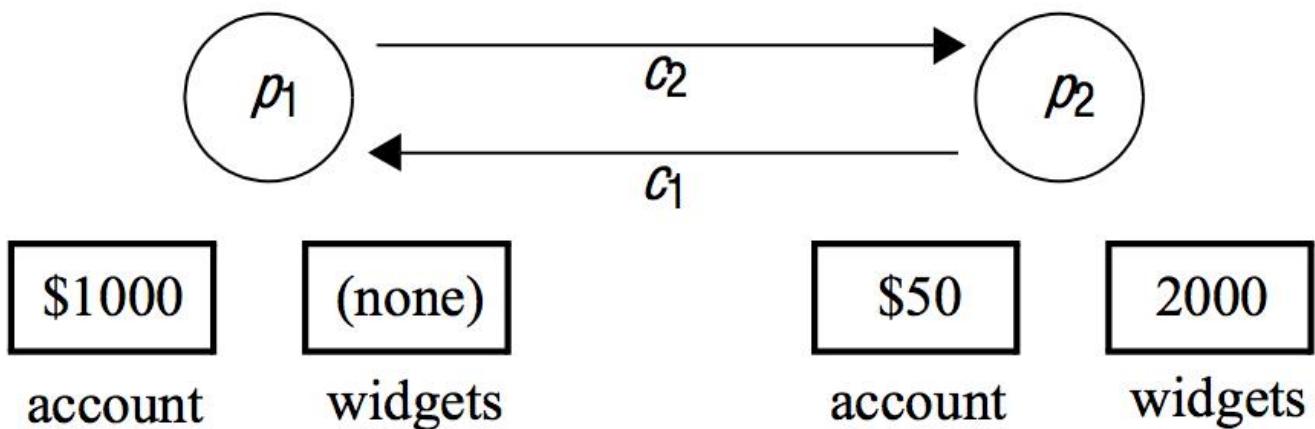
p_i sends one marker message over c

(before it sends any other message over c).

Estados globales

Algoritmo de instantánea de Chandy y Lamport

- Ejemplo: dos procesos p_1 y p_2 conectados por dos canales unidireccionales c_1 y c_2 . Los dos procesos comercian con artículos. El proceso p_1 , envía una orden de compra de artículos sobre c_2 hacia p_2 incluyendo el pago al precio de 10 dólares por artículo.
- Algún tiempo después, el proceso p_2 envía artículos a través del canal c_1 , hacia p_1 . Los procesos están en el estado inicial que se ve en la Figura. El proceso p_2 ya ha recibido una orden por cinco artículos que serán enviadas en breve a p_1 .

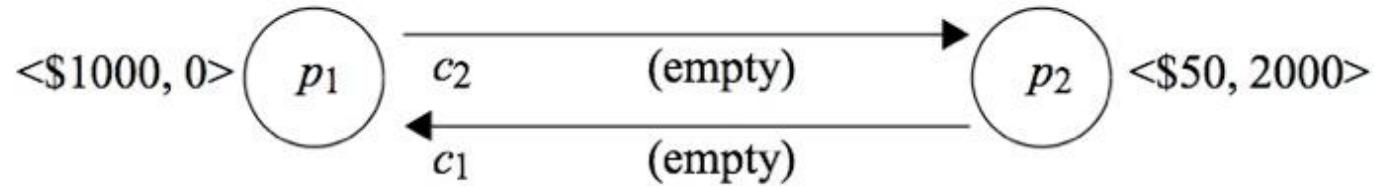


Estados globales

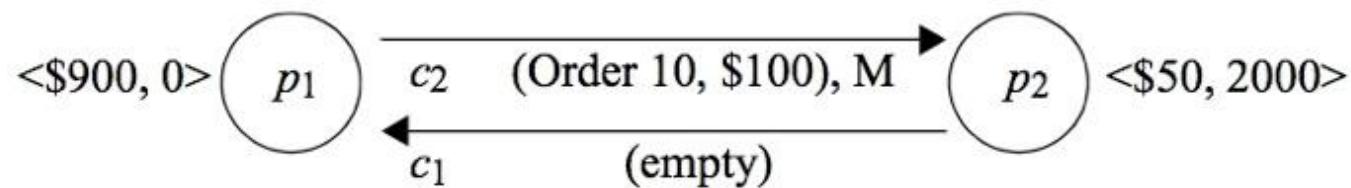
Algoritmo de instantánea de Chandy y Lamport

- La ejecución de los procesos se visualiza en la siguiente Figura.

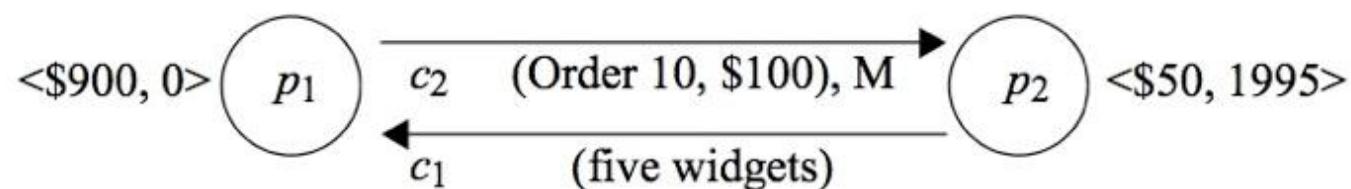
1. Global state S_0



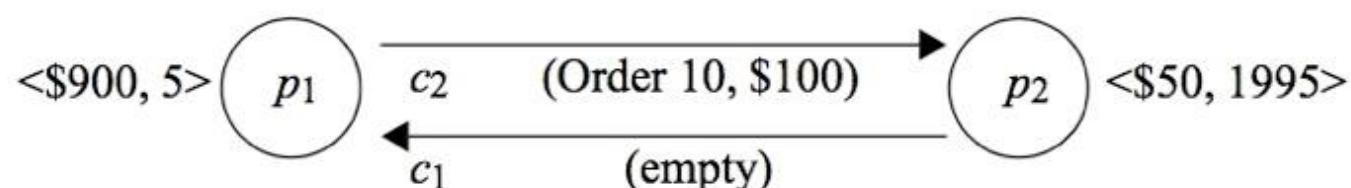
2. Global state S_1



3. Global state S_2



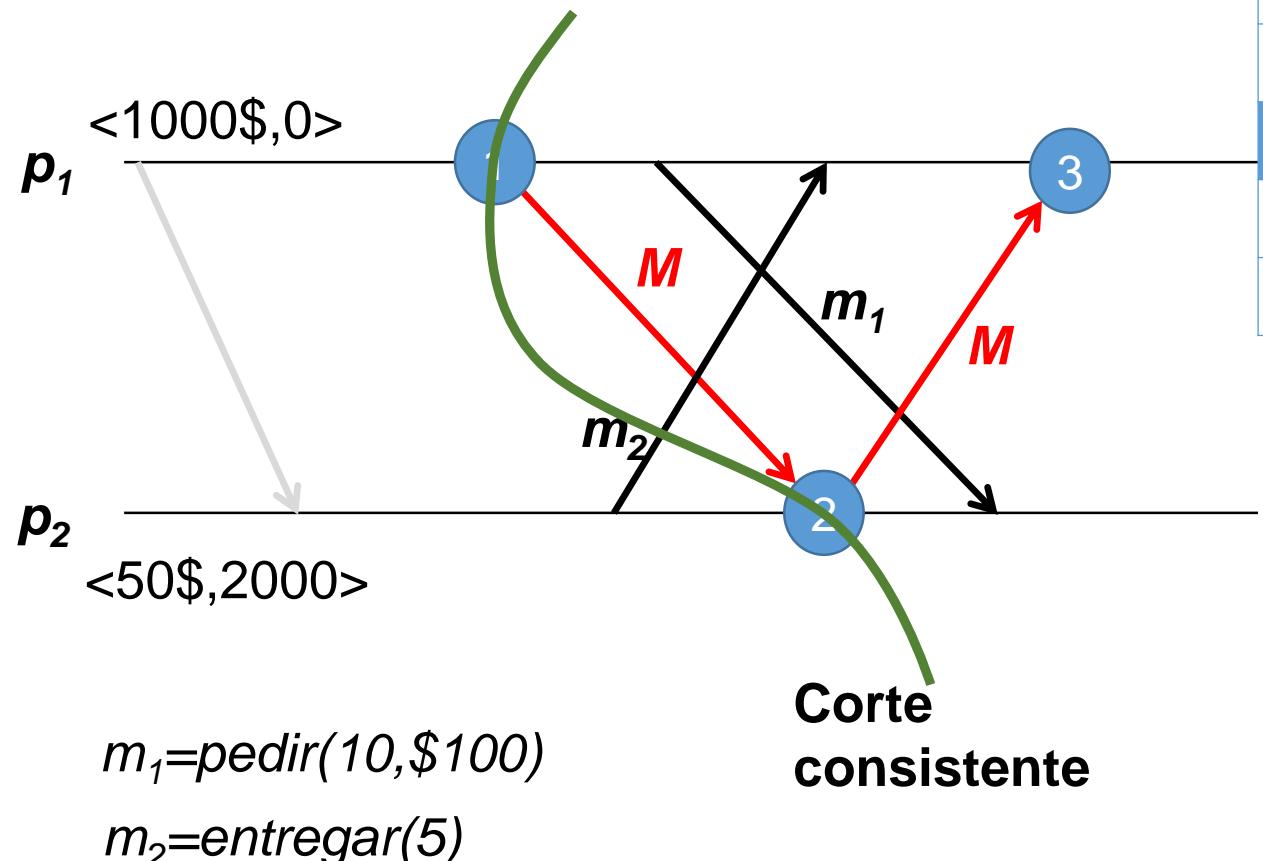
4. Global state S_3



(M = marker message)

Estados globales

Algoritmo de instantánea de Chandy y Lamport



Estado local	Paso	Valor
P_1	1	$<1000\$,0>$
P_2	2	$<50\$,1995>$
Estado Canal	Paso	Valor
$C_2(p_1, p_2)$	1	3
$C_1(p_2, p_1)$	2	$<\text{vacío}>$

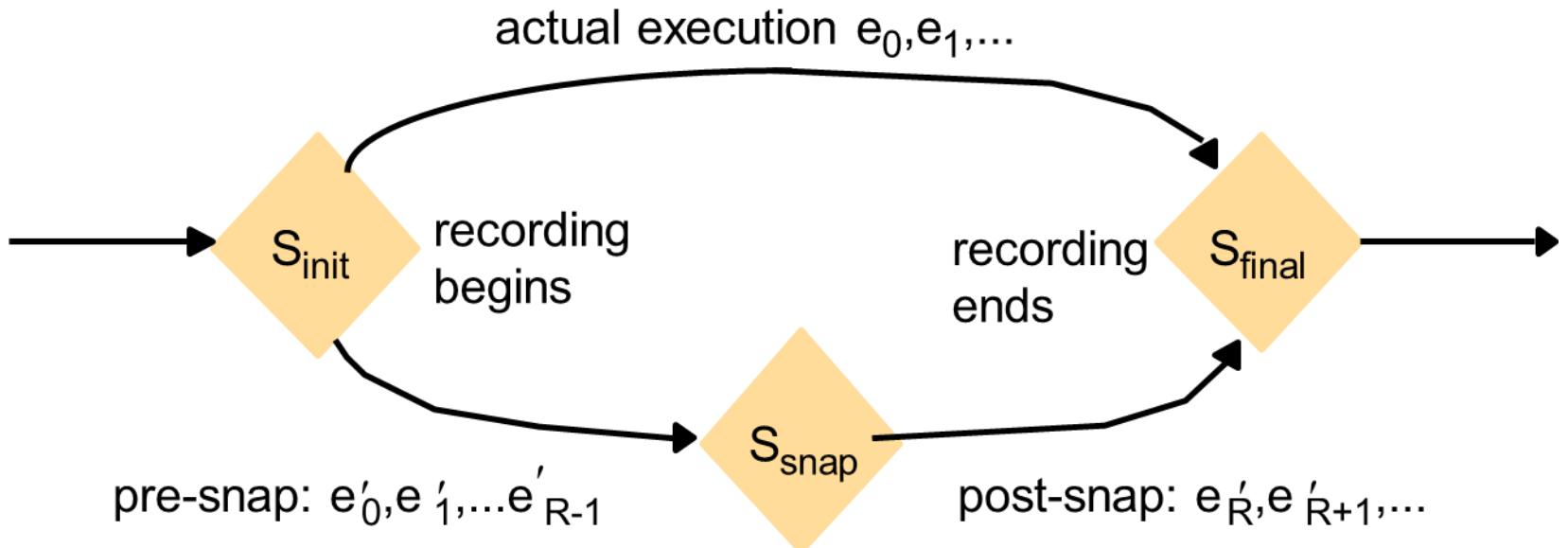
Estado
global
consistente

m_2

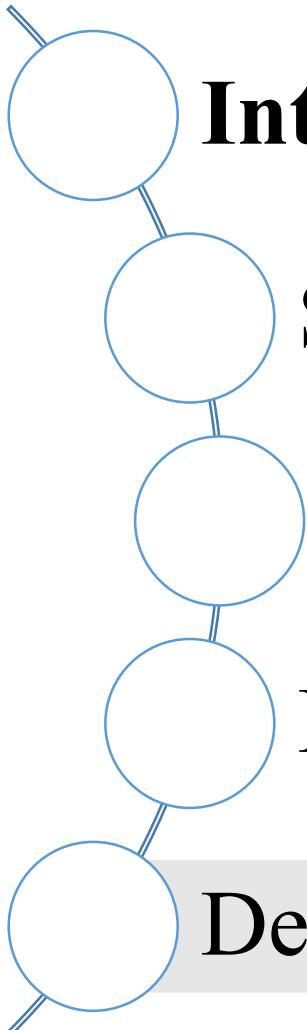
Estados globales

Alcanzabilidad entre estado en el algoritmo de instantánea

- Sea $Sys = e_0, e_1, e_2 \dots, e_{n-1}$ el conjunto de eventos ordenados del sistema
- Sea $Sys' = e'_0, e'_1, e'_2 \dots$



Agenda



Introducción

Sincronización

Tiempos y Relojes Lógicos

Estados Globales

Depuración Distribuida

Depuración distribuida

- Los SD son complejos de depurar.
- Ejemplo: Un desarrollador ha escrito una aplicación en la que cada proceso p_i contiene una variable $x_i (i = 1, 2, \dots, N)$. Las variables cambian a medida que se ejecuta el programa, pero se precisa que estén a menos de cierto valor δ de otras. Pero, hay un error en el programa, y el desarrollador sospecha que bajo ciertas circunstancias $|x_i - x_j| > \delta$ incumpliendo la restricción de consistencia.
- Problema: ¿Dónde ocurre eso? ¿En qué proceso/s? ¿Cuáles son las circunstancias?

Depuración distribuida

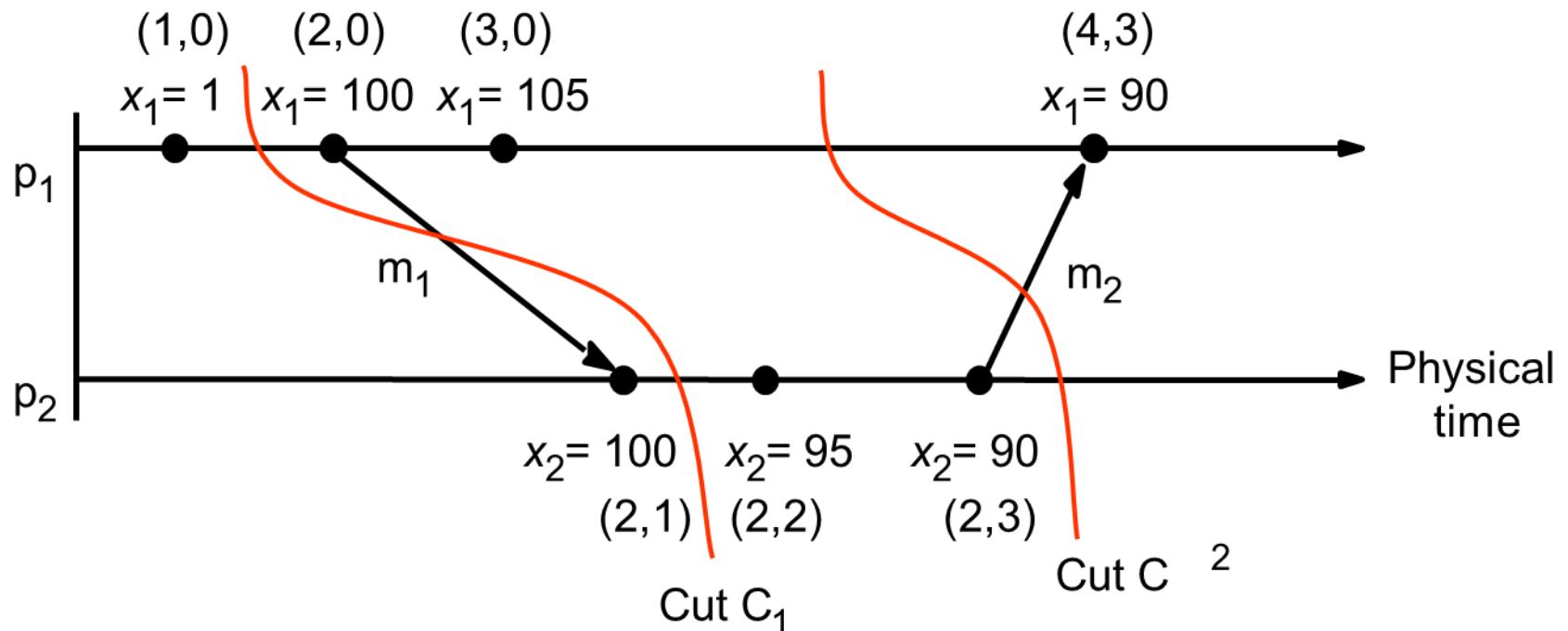
Predicados

- La ejecución de un SD se puede caracterizar (y depurar) por las transiciones entre estados globales consistentes
 - $S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow \dots \rightarrow S_n$
- Un predicado de estado global es una función
 - $\{\text{Conjunto de estados globales}\} \rightarrow \{V, F\}$
 - Determinar una condición del SD equivale a evaluar su predicado
- Características posibles de un predicado
 - **Estabilidad:** el valor del predicado no varía con los nuevos sucesos (por ejemplo, en el caso de interbloqueo o terminación)
 - **Seguridad:** el predicado tiene valor falso para cualquier estado alcanzable desde S_0 (deseable para errores)
 - **Vitalidad:** el predicado tiene valor verdadero para algún estado alcanzable desde S_0 (deseables para situaciones necesarias)

Depuración distribuida

Predicados

- Ejemplo: Imaginemos un sistema de 2 procesos donde queremos controlar el predicado $\Phi: |x_i - x_j| > 50$



En C_1 : $|1 - 100| > 50 \Rightarrow \Phi$ verdadero

En C_2 : $|105 - 90| > 50 \Rightarrow \Phi$ falso

Depuración distribuida

Monitorización

- Depurar un SD requiere registrar su estado global, para poder hacer evaluaciones de predicados en dichos estados.
 - Generalmente, la evaluación trata de determinar si el predicado Φ cumple con la condición “posiblemente” o “sin duda alguna”.
- Monitorización del estado global:
 - Descentralizada: algoritmo de instantánea de Chandy y Lamport
 - Centralizada: algoritmo de Marzullo y Neiger
 - Los procesos envían su estado inicial al proceso monitor
 - Periódicamente, le vuelven a enviar su estado.
 - El monitor registra los mensajes de estado en colas de proceso: una por proceso

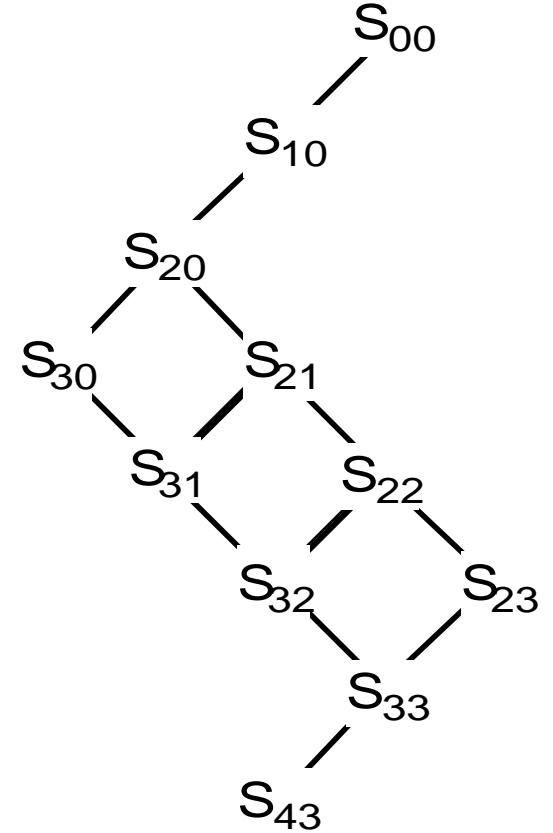
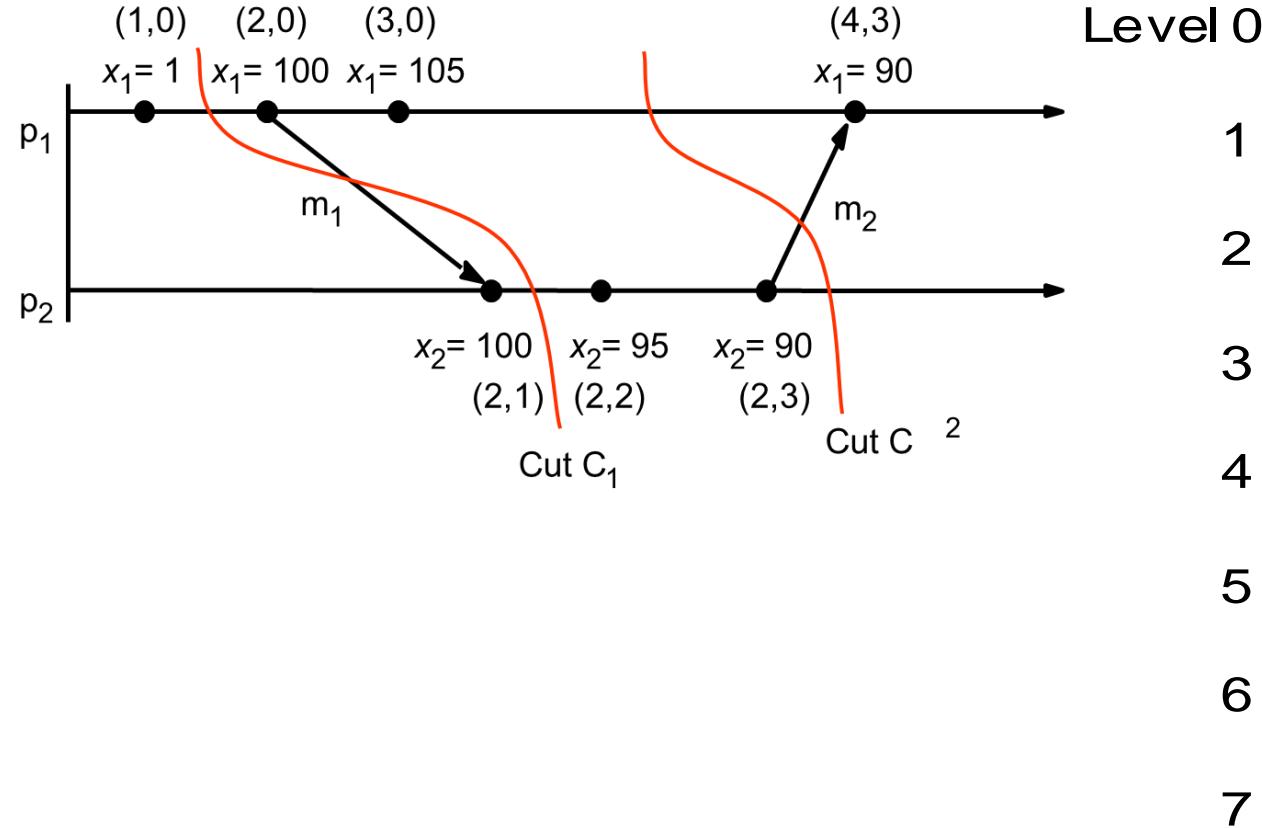
Depuración distribuida

Evaluación de predicados

- Objetivo de la monitorización
 - Determinar si un predicado Φ es “posiblemente” o “sin duda alguna” verdadero en un determinado punto de la ejecución.
 - El proceso monitor sólo registra los estados globales consistentes
 - Los únicos en que podemos evaluar el predicado con certeza
- Linealización: es la ruta entre estados.
- La afirmación ***posiblemente* Φ** significa que hay un estado consistente S a través del cual pasa una linealización de H tal que $\Phi(S)$ es Verdadero.
- La afirmación ***sin duda alguna* Φ** significa que para todas las linealizaciones L de H , hay un estado consistente S a través del cual pasa L tal que $\Phi(S)$ es Verdadero.

Depuración distribuida

Red de estados globales

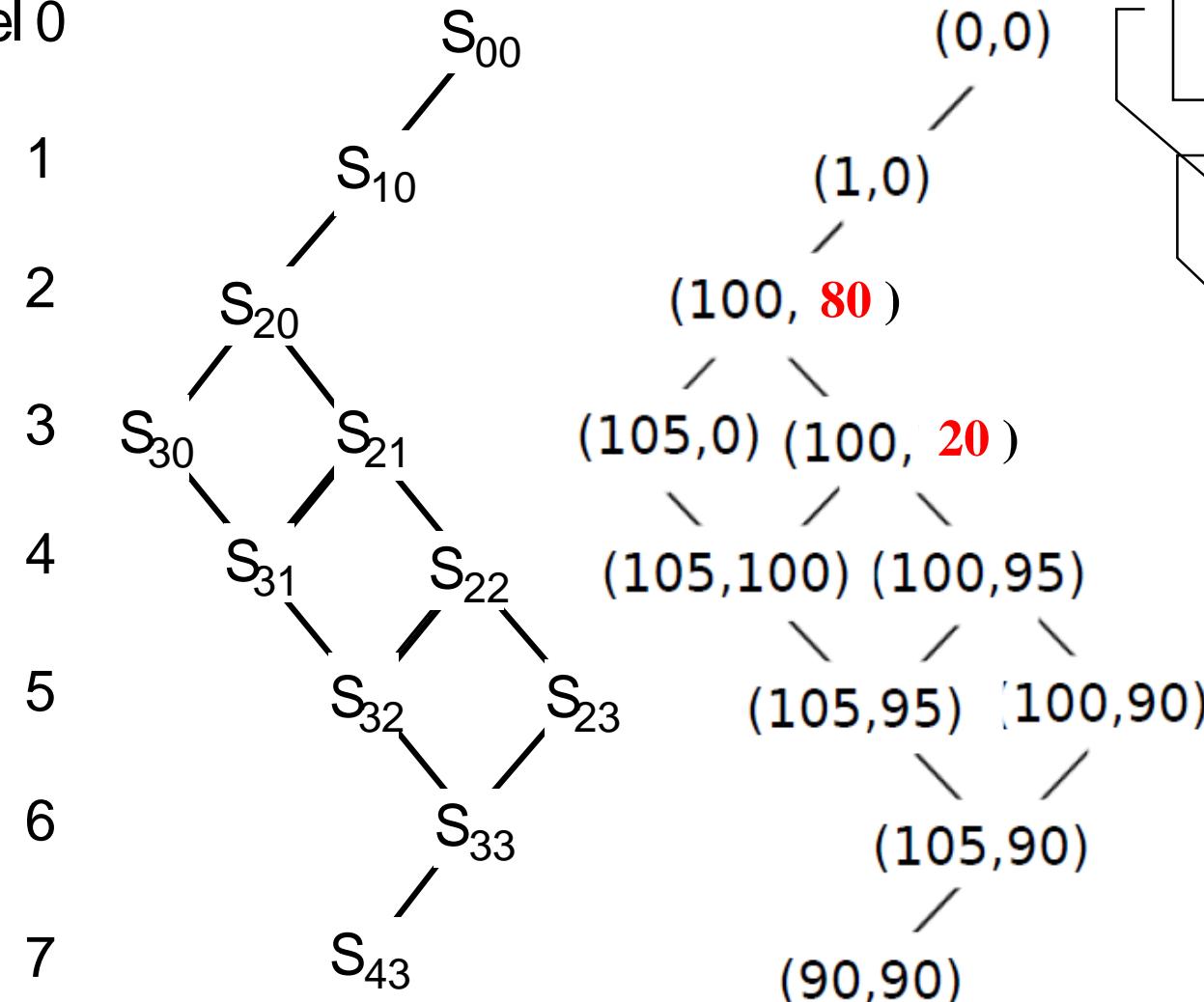


S_{ij} = estado global después de i eventos en el proceso 1 y j eventos en el proceso 2.

Depuración distribuida

Evaluación instantánea de predicados

Level 0



$$\Phi: |x_i - x_j| > 50$$

Sin duda alguna Φ

Posiblemente Φ

V V

F F

F

F F

F F

F F

F F

F F

Depuración distribuida

Evaluación instantánea de predicados

- Algoritmos para evaluar posiblemente \Box y sin duda alguna \Box

1. *Evaluating possibly ϕ for global history H of N processes*

$L := 0;$

$States := \{ (s_1^0, s_2^0, \dots, s_N^0) \};$

while ($\phi(S) = False$ for all $S \in States$)

$L := L + 1;$

$Reachable := \{ S': S' \text{ reachable in } H \text{ from some } S \in States \wedge \text{level}(S') = L \};$

$States := Reachable$

end while

output "possibly ϕ ";

2. *Evaluating definitely ϕ for global history H of N processes*

$L := 0;$

if ($\phi(s_1^0, s_2^0, \dots, s_N^0)$) *then* $States := \{ \}$ *else* $States := \{ (s_1^0, s_2^0, \dots, s_N^0) \};$

while ($States \neq \{ \}$)

$L := L + 1;$

$Reachable := \{ S': S' \text{ reachable in } H \text{ from some } S \in States \wedge \text{level}(S') = L \};$

$States := \{ S \in Reachable: \phi(S) = False \}$

end while

output "definitely ϕ ";

Depuración distribuida

Resumen

- Consiste en
 - Determinar el predicado que queremos evaluar.
 - Especificar un método para construir una red o historia de estados globales consistentes
 - Teniendo en cuenta el predicado para optimizar tráfico
 - Quizás teniendo en cuenta los canales de comunicación
 - Evaluar si nuestro predicado se cumple en algún momento
 - Si es posible, se cumplirá para alguna de las linealizaciones
 - Si es sin duda, se cumplirá para todas las linealizaciones



Resumen

Tiempo y Estados Globales

- Cualquier medición de tiempo **requiere de referencias**.
- Dichas referencias nunca son totalmente constantes, con lo que las mediciones de tiempo reales **siempre tienen una deriva**.
- Para el **uso diario**, obtenemos la máxima precisión de las mediciones **TAI** y **UT1**, pero ni siquiera estas son precisa.
- Mediante **relojes físicos** podemos hacer sincronización externa a una referencia local (Cristian), global (NTP) o hacer sincronización interna (Berkeley). En cualquier caso, **no son suficientemente precisos** para algoritmos delicados. Mediante relojes físicos podemos hacer sincronización externa a una referencia local (Cristian), global (NTP) o hacer sincronización interna (Berkeley). En cualquier caso, no son suficientemente precisos para algoritmos delicados



Resumen

Tiempo y Estados Globales

- **Lamport** definió las **dos condiciones temporales lógicas** que podemos conocer: el evento de recepción ocurre antes que el evento de envío y los eventos locales ocurren en orden. Estas condiciones permiten establecer **relaciones antes-que** y trabajar mediante **relojes lógicos**.
- El **estado global** de un SD se refiere al estado de todos sus procesos y mensajes en transito. Un algoritmo para capturarlo es el de **instantánea de Chandy/Lamport**.
- La **depuración distribuida** determina las posibles transiciones entre estados globales, para identificar la posibilidad o certeza de que un determinado evento ocurra.

Fin

- Muchas gracias