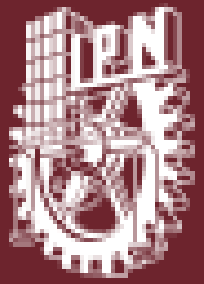


Instituto Politécnico Nacional
Escuela Superior de Cómputo

Sincronización y coordinación

M. en C. Hermes Francisco Montes Casiano
hermes.escom@gmail.com

Índice

1. Introducción
2. Relojes, eventos y estados de proceso
3. Sincronización de relojes físicos
4. Relojes lógicos
5. Exclusión mutua distribuida

Introducción

- Es tiempo es una característica importante e interesante en los sistemas distribuidos.
  - El tiempo es una cantidad que a menudo se quiere medir de forma precisa.
  - Se han desarrollado algoritmos que dependen de la sincronización de reloj para varios problemas en distribución.
- No existe un reloj físico especial en el universo al que podamos invocar cuando se quiere medir intervalos de tiempo.

Sincronización

Para saber en qué hora del día ocurrió un evento particular en una computadora es necesario sincronizar su reloj, con una fuente de tiempo externa.

Introducción

- La noción de tiempo físico es problemática en un sistema distribuido.

¿El tiempo físico es un problema?

El problema se basa en una limitación en la capacidad para marcar sucesos en diferentes nodos de una manera suficientemente precisa para conocer el orden en el que ocurrieron cualquier par de sucesos o si ellos ocurrieron simultáneamente.

- No hay un tiempo absoluto, global al que se pueda invocar.
- En ocasiones se necesita observar sistemas distribuidos y establecer si ciertos estados ocurrieron al mismo tiempo.

Índice

1 Introducción

2 Relojes, eventos y estados de proceso

3 Sincronización de relojes físicos

4 Relojes lógicos

5 Exclusión mutua distribuida

Relojes, eventos y estados de proceso

Introducción

- Considere que un sistema distribuido consta de una colección $\wp$ de N procesos p_i, $i = 1, 2, \ldots, N$.
- Cada proceso se ejecuta en un único procesador y los procesadores no comparten memoria.
- Cada proceso p_i, en $\wp$ tiene un estado s_i que se transforma a medida que se ejecuta.

Estado de un proceso

El estado de un proceso incluye los valores de todas sus variables, puede incluir también los valores de cualquiera de los objetos en el entorno de su sistema operativo local que lo afecte.

- Se supone que los procesos no se pueden comunicar entre ellos sólo por medio del paso de mensajes.

Relojes, eventos y estados de proceso

Introducción

- A medida que cada proceso p_i se ejecuta tomca una serie de acciones:
  - Enviar un mensaje
  - Recibir un mensaje
  - Operación para transformar el estado p_i, que cambia uno o más valores de s_i.
- Un evento es la ocurrencia de una única acción que un proceso realiza a medida que se ejecuta.
- La secuencia de sucesos en un único proceso p_i puede colocarse en un orden único total con base en la relación de ocurrencia anterior ($\rightarrow$).

Relación de ocurrencia anterior

$e \rightarrow e'$ si y sólo si el evento e ocurre antes del e' en p_i.

Relojes, eventos y estados de proceso

Introducción

- Se define la *historia* del proceso p_i como la serie de eventos que tienen lugar en él, ordenados con base en la relación $\rightarrow$.

$$historia(p_i) = h_i =< e_i^1, e_i^2, e_i^3, \cdots > \quad (1)$$

Relojes, eventos y estados de proceso

Relojes

- Las computadoras pueden disponer de su propio reloj físico.

Reloj

Los relojes son dispositivos electrónicos que cuentan las oscilaciones que ocurren en un cristal a una frecuencia definida, dividen esa cuenta y almacenan el resultado en un registro contador.

- El sistema operativo lee el valor del reloj hardware $H_i(t)$ del nodo, lo escala y añade una compensación para producir un reloj software

$$C_i(t) = \alpha H_i(t) + \beta \tag{2}$$

que mide apróximadamente el tiempo físico t. para el proceso p_i.

Relojes, eventos y estados de proceso

Relojes

- En otras palabras, cuando el tiempo real en un marco de referencia absoluto es t, $C_i(t)$ es la lectura del reloj software.
- En general, el reloj no es suficientemente preciso y por lo tanto $C_i(t)$ difiere de t.
- Si $C_i(t)$ se comporta suficientemente bien se puede utilizar para marcar el tiempo de cualquier evento en p_i.
- La tasa a la que ocurren los sucesos dependen de factores tales como la longitud del ciclo de instrucción del procesador.

Marcas de tiempo

Es necesario indicar qué eventos sucesivos tendrán marcas de tiempo diferentes solo si la *resolución de reloj*, el periodo entre actualizaciones del valor del reloj, es más pequeño que el intervalo de tiempo entre eventos sucesivos.

Relojes, eventos y estados de proceso

Relojes

- Los relojes tienden a no estar de acuerdo.
- La diferencia instantánea entre las lecturas de dos relojes cualquierase llama *sesgo*.
- Los relojes basadosen cristal están sujetos a *derivas de reloj*, lo cual significa que ellos cuentan el tiempo a diferentes ritmos.
- La diferencia acumulada en el periodo de oscilación entre dos relojes puede conducir a una diferencia observable.
- Un ritmo de *deriva de reloj* es el cambio en la compensación entre el reloj y un reloj de referencia nominal perfecto.

Ritmos de deriva

- Para un reloj de cuarzo ordinario es de 10^{-6}.
- Para un reloj de cuarzo de alta precisión es de 10^{-7} o 10^{-8}.

Relojes, eventos y estados de proceso

Tiempo Universal Coordinado

- Los relojes de una computadora pueden sincronizarse con fuentes externas de tiempo de gran precisión.
- Los relojes físico más precisos utilizan asciladores atómicos, cuyo ritmo de deriva es de apróximadamente 10^{13}.
- La salida de estos relojes atómicos se utiliza como estándar para el tiempo real transcurrido, conocido como *Tiempo Atómico Internacional*.
- El segundo estándar se ha definido como $9,192,631,770$ periodos de transición entre dos niveles hiperfinos del estado fundamental del Cesio-133.
- Debido a que el tiempo de rotación de la tierra se va haciendo cada vez más largo el tiempo atómico y el astronómico tienen tendencia a separarse.

Relojes, eventos y estados de proceso

Tiempo Universal Coordinado

- El tiempo Universal Coordinado (UTC) es un estándar internacional de cronometraje.
- Está basado en el tiempo atómico, aunque ocasionalmente se inserta o elimina un segundo con la finalidad de mantenerse en sintonía con tiempo astronómico.
- Las señales UTC se sincronizan y difunden regularmente desde las estaciones de radio terrestres y los satélites.
- Las señales recibidas desde las estaciones terrestres tienen una precisión del orden de 0,1-10 milisegundos.
- Las computadoras con receptores adjuntos pueden sincronizar sus relojes con estas señales de tiempo.

Índice

1 Introducción

2 Relojes, eventos y estados de proceso

3 Sincronización de relojes físicos

4 Relojes lógicos

5 Exclusión mutua distribuida

Introducción

- Para conocer en qué hora del día ocurren los sucesos en los procesos de un sistema distribuido $\wp$ es necesario sincronizar sus relojes.
  1. **Sincronización externa:** cuando los relojes de un sistema distribuido se sincronizan con una fuente de tiempo externa autorizada.
     Para una sincronización dada $D > 0$ y para una fuente S de tiempo UTC $|S_i(t) - C_i(t)| < D$, para $i = 1, 2, \ldots, N$
  2. **Sincronización interna:** cuando los relojes C_i están sincronizados con otro con un grado de precisión conocido.
     Para una sincronización dada $D > 0, |C_i(t) - C_j(t)| < D$, para $i = 1, 2, \ldots, N$
- Los relojes que están sincronizados internamente no están necesariamente sincronizados externamente.

Introducción

Sincronización

Se deduce de las definiciones que si un sistema distribuido $\wp$ está sincronizado externamente con un límite D entonces el mismo sistema está sincronozado internamente con un límite $2D$.

- Es común definir que un reloj hardware H es correcto si su ritmo de deriva cae dentro de un límite conocido $\rho > 0$.
- Eso significa que el error midiendo el intervalo entre tiempos reales t y t' $(t' > t)$ está limitado por:

$$(1 - \rho)(t' - t) \leq H(t') - H(t) \leq (1 + \rho)(t' - t) \qquad (3)$$

la cual prohibe saltos en el valor de relojes hardware.

Introducción

Monoticidad

Es la condición que un reloj C sólo avanza siempre que:

$$t' > t \Rightarrow C(t') > C(t) \tag{4}$$

- Un reloj que no se atiene a ninguna de las condiciones de corrección aplicadas se dice que es un *reloj defectuoso*.
- Un *fallo de ruptura* de reloj se dice que ocurre cuando el reloj deja de pulsar por pleto; cualquier otro fallo de reloj es un fallo arbitrario.

Precisión vs Coreectitud

Los relojes no tienen que ser precisos para ser correctos, ya que el objetivo puede ser la sincronización interna más que la externa.

Sincronización en un sistema síncrono

- En un sistema síncrono se cononocen los límites para:
  1. El ritmo de deriva de los relojes,
  2. El máximo retardo de transmisión de mensajes y
  3. El tiempo para ejecutar cada paso de un proceso.

Escenario de sincronización

- Un proceso envía el tiempo t de su reloj local a otro en un mensaje m.
- El proceso receptor podría tener su reloj en el tiempo $t + T_{trans}$.
- T_{trans} es el tiempo para transmitir m entre ellos.
- Los dos relojes podrían coincidir.

Sincronización en un sistema síncrono

Escenario de sincronización

- Un proceso envía el tiempo t de su reloj local a otro en un mensaje m.
- El proceso receptor podría tener su reloj en el tiempo $t + T_{trans}$.
- T_{trans} es el tiempo para transmitir m entre ellos.
- Los dos relojes podrían coincidir.
- Sin embargo, T_{trans} está sujeto a variaciones y es desconocido.
  - Otros procesos están compitiendo por recursos.
  - Otros mensajes compiten con m por la red.

Sincronización en un sistema síncrono

- Siempre hay un tiempo mínimo de transmisión *min* si no se ejecutara ningún otro proceso y no existiera más tráfico en la red.
- *min* puede ser medido o estimado de forma conservadora.
- Por definición también hay un límite superior *max* del tiempo tomado para transmitir cualquier mensaje.

Sistema asíncronos, el modelo de Internet

Para un sistema asíncrono se puede inferir $T_{trans} = min + x$, donde $x \geq 0$. El valor de x ni es conocido en un caso particular.

Sincronización en un sistema síncrono

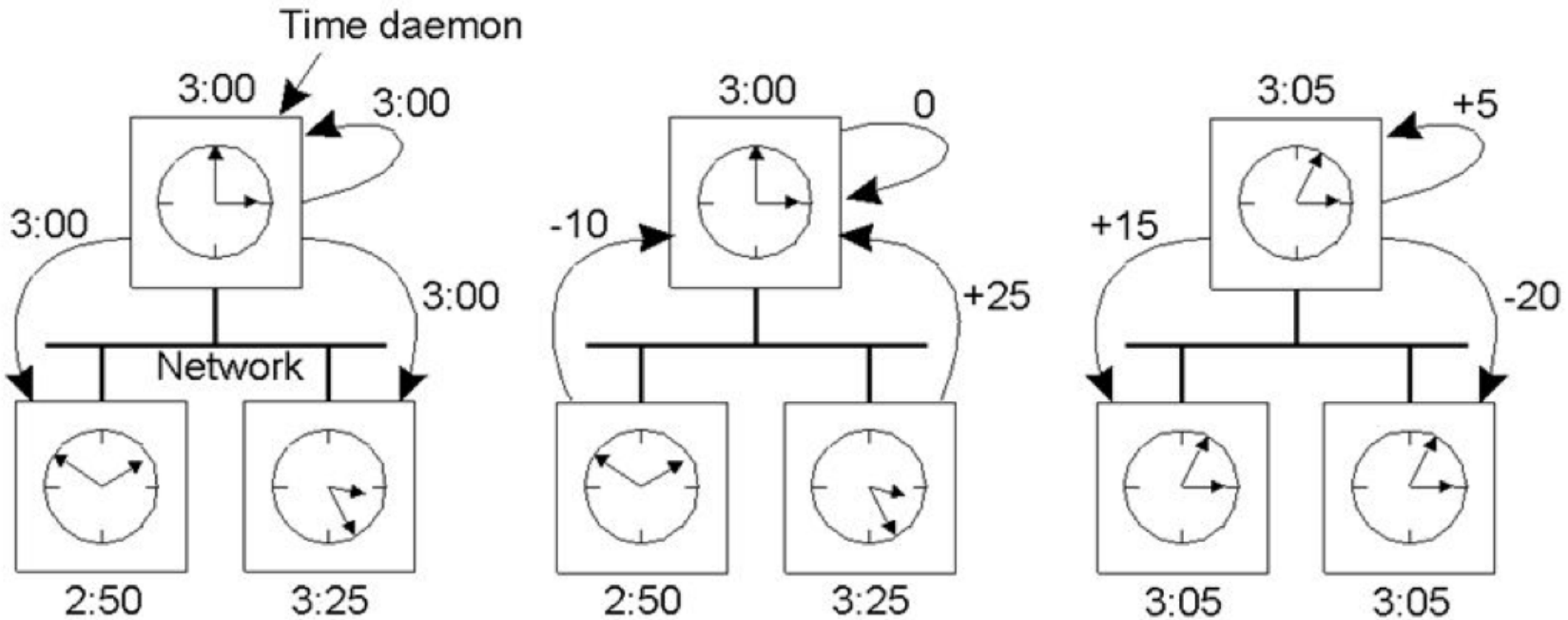
- Sea la incertidumbre en el tiempo de transmisión del mensaje u, donde $u = (max - min)$.
  1. Si el receptor coloca su reloj a $t + min$ el sesgo del reloj puede ser como mucho u.
  2. Si el receptor coloca su reloj a $t + max$ el sesgo del reloj puede ser como mucho u.
  3. Si el receptor coloca su reloj a $t + \frac{(max+min)}{2}$, entonces su sesgo es como mucho $\frac{u}{2}$.
- En general, para un sistema síncrono, el limite óptimo que puede consegirse en el sesgo de reloj cuando se sincronizan N relojes es $u(1 - \frac{1}{N})$.

Algoritmo de Berkley

- Gusella y Zatty describieron un algortimo para sincronización interna que ellos desarrollaron para computadoras usando UNIX Berkeley.
- En este algoritmo se elige un nodo **coordinador** para actuar como maestro.
- El resto de los nodos se denominan **esclavos**.
- El algoritmo consiste en lo siguiente:
  1. El coordinador consulta periódicamente a los esclavos cuyos relojes están para sincronizarse.
  2. Los esclavos le devuelven sus valores de reloj.
  3. El maestro estima los tiempos locales de reloj de los esclavos observando los tiempo de ida y vuelta y promedia sus valores (incluyendo su propio reloj).
  4. El maestro devuelve a los esclavos la cantidad que necesitan para sincrinizar sus relojes.

Algoritmo de Berkley

Ejemplo



Algoritmo de Berkley

- El balance de las probabilidades es que el este primedio contraresta las tendencias de los relojes individuales a funcionar rápido o lento.
- La precisión del protocolo depende de un tiempo de ida y vuelta máximo nominal entre maestro y esclavos.
- El maestro elimina cualquier lectura adicional asociada con tiempos más grandes que el máximo.
- El algoritmo elimina las lecturas de relojes defectuosos.
- El maestro toma un promedio tolerante a fallos, es decir elige un subconjunto de relojes que no difieran entre ellos más de una determinada cantidad y el promedio se toma sólo con esos valores.

El protocolo de tiempo de red (NTP)

- NTP define una arquitectura para un servicio de tiempo y un protocolo para distribuir la información del tiempo sobre Internet.
- Los objetivos de NTP son:
  1. Proporcionar un servicio que permita a los clientes estar sincronizados de forma precisa a UTC.
  2. Proporcionar un servicio fiable a pesar de las pérdidas de conectividad.
  3. Permitir resincronizaciones con frecuencia para compensar las tasas de derivas.
  4. Proporcionar protección contra la interferencia.
- Los *servidores primarios* están conectados directamente a una fuente de tiempo como un radioreloj recibiendo UTC.
- Los *servidores secundarios* están sincronizados, en el fondo, con *servidores primarios*.

El protocolo de tiempo de red (NTP)

- Los servidores NTP se sincronizan entre sí en uno de los tres modos:

  Multidifusión: está pensado para su uso en LAN de alta velocidad.

  Llamada a procedimiento: es similar al método de *Cristian*. este modo es adecuado cuando se requieren precisiones mas altas que las obtenidas con el método de multidifusión.

  Modo simétrico: está pensado para su uso en servidores que proporcionan información de tiempo en LANs y por los estratos más bajos de la subred de sincronización.

- En todos los modos los mensajes se entregan utilizando el protocolo UDP.

El protocolo de tiempo de red (NTP)

- En el modo de llamada a procedimiento remoto y en el simétrico, los procesos intercambian pares de mensajes.
- Cada mensaje lleva marcas de tiempo de los sucesos del mensaje reciente:
  1. El tiempo local cuando el mensaje anterior NTP entre el par fue enviado (T_{i-3}).
  2. El tiempo local cuando el mensaje anterior NTP entre el par fue recibido (T_{i-2}).
  3. El tiempo local cuando el mensaje actual fue transmitido (T_{i-1}).
  4. El tiempo local cuando el mensaje actual fue recibido (T_i).

El protocolo de tiempo de red (NTP)

Compensación

Por cada par de mensajes enviados entre dos servidores NTP calcula una compensación (o), la cual es una estimación de la deriva actual entre dos relojes.

Retardo

El retardo es el tiempo total de transmisión para los dos mensajes.

Si el desplazamiento del reloj en B con A es o, y si los tiemposde transmisión actuales para m y m' son t y t' relativamente, entonces:

$$T_{i-2} = T_{i-3} + t + o \tag{5}$$

$$T_i = T_{i-1} + t' + o \tag{6}$$

El protocolo de tiempo de red (NTP)

Si el desplazamiento del reloj en B con A es o, y si los tiemposde transmisión actuales para m y m' son t y t' relativamente, entonces:

$$T_{i-2} = T_{i-3} + t + o \tag{5}$$

$$T_i = T_{i-1} + t' + o \tag{6}$$

Lo cual conduce a:

$$d_i = t + t' = T_{i-2} - T_{i-3} + T_i - T_{i-1} \tag{7}$$

$$o = o_i + \frac{(t - t')}{2} \tag{8}$$

$$o_i = \frac{(T_{i-2} - T_{i-3} + T_{i-1} - T_i)}{2} \tag{9}$$

El protocolo de tiempo de red (NTP)

- Usando el hecho de que $t, t' \geq 0$ se puede ver que $o_i - \frac{d_i}{2} \leq o \leq o_i + \frac{d_i}{2}$
- o_i es una estimación de la deriva.
- d_i es una medida de la precisión de esta estimación.
- Los servidores NTP aplican un algoritmo de filtrado de datos a pares sucesivos de $< o_i, d_i >$, llamada *filtro de dispersión*.
- Los ocho pares $< o_i, d_i >$ más recientes son retenidos, se elige como estimación de o el valor de o_j que corresponde con el mínimo valor de d_i

Índice

1 Introducción

2 Relojes, eventos y estados de proceso

3 Sincronización de relojes físicos

4 Relojes lógicos

5 Exclusión mutua distribuida

Introducción

Lamport

Señaló que dado que no se pueden sincronizar perfectamente los relojes un sistema distribuido, no se puede usar el tiempo físico para obtener el orden de cualquier par arbitrario de sucesos.

- En el ámbito de un proceso, los sucesos están ordenados de forma única por los tiempos mostrados en el reloj físico.
- Se puede utilizar un esquema que es similar a la causalidad física, pero que se aplique en los sistemas distribuidos.
- La ordenación se basa en los siguientes puntos:
  1. Si dos sucesos han ocurrido en el mismo proceso $p_i (i = 1, 2, ;N)$.
  2. Cuando se envía un mensaje entre procesos, el suceso de enviar el mensaje ocurre antes del de recepción del mismo.

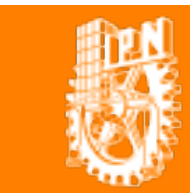
Introducción

- Se define la relación de *ocurrencia anterior*, indicada por $\rightarrow$, como sigue:

  SA1: si $\exists$ un proceso p_i: e $rightarrow_i'$, entonces $e \rightarrow_i e'$.

  SA1: para cualquier mensaje m, $envia(m) \rightarrow recibe(m)$.

  SA1: si e, e' y e'' son sucesos tal que $e \rightarrow e'$ y $e' \rightarrow e''$, entonces $e \rightarrow e''$.



Introducción

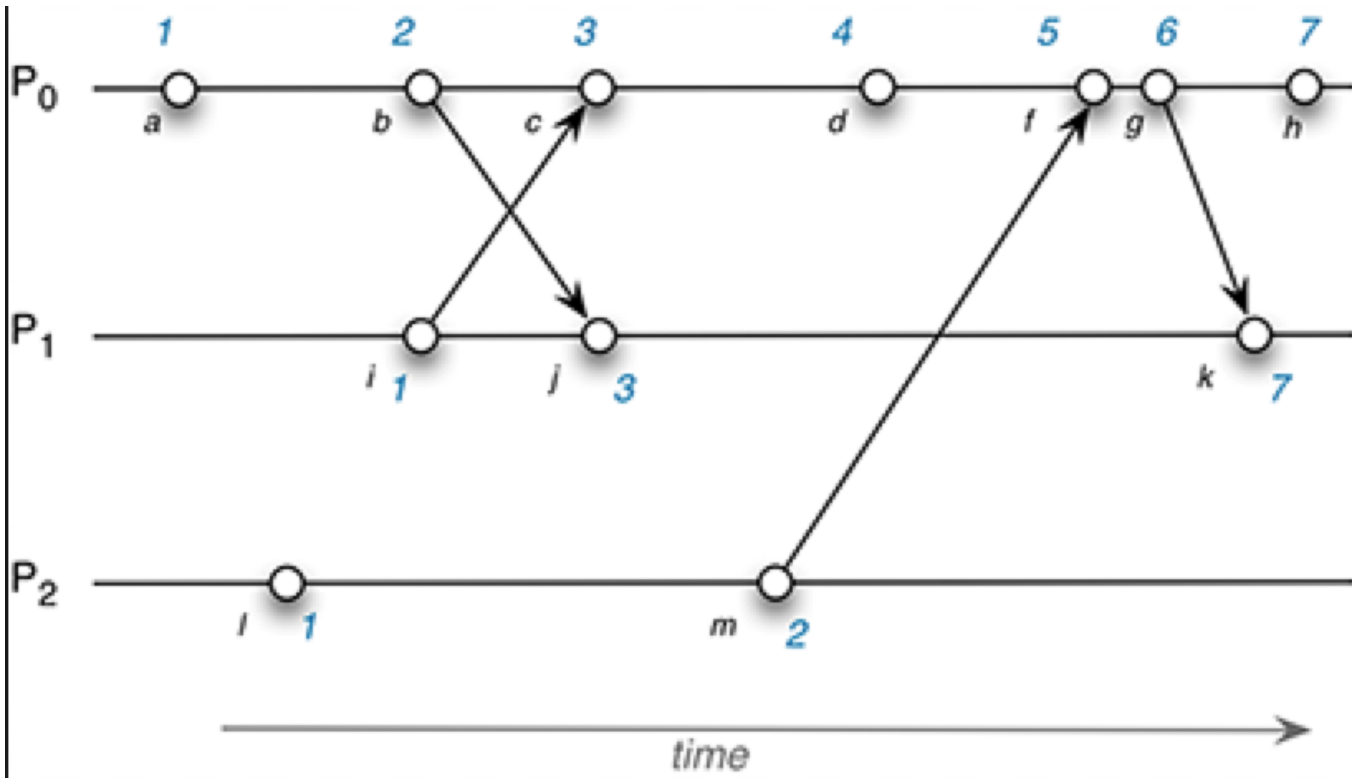


Figura : Relojes logicos de Lamport.

Introducción

- De la figura se puede observar que:
  1. $a \rightarrow b$
  2. $b \rightarrow j$
  3. $a \rightarrow j$
- También se puede observar que:
  1. $a \nrightarrow l$
  2. $l \nrightarrow a$
- Dado que a y l ocurren en diferentes procesos y no hay una cadena de mensajes que intervengan entre ellos, se dice que a y e son concurrentes, $a \parallel e$.
- Un punto importante a señalar es que aun produciéndose la relación de *ocurrencia anterior* entres dos sucesos, el primero podría o no haber causado realmente el segundo.

Relojes lógicos

Lamport

Inventó un mecanismo simple por el que la relación de *ocurrencia anterior* puede capturarse númericamente, denominado *reloj lógico*.

Reloj lógico

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.
- Las *marcas de tiempo de Lamport* en un proceso se hacen con base en su *reloj lógico*.
- Una marca de tiempo para un suceso e se representa como $L(e)$.

Relojes lógicos

- Para capturar la relación $\rightarrow$, los procesos actualizan sus relojes lógicos y transmiten sus valores en mensajes:

  RL1: L_i se incrementa antes de emitir cada suceso en el proceso p_i. $L_i = L_i + 1$.

  RL2:
  1. Cuando un proceso p_i envía un mensaje m, acarrea en m el valor de $t = L_i$.
  2. 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 $recibe(m)$.

- Generalmente los relojes logicos se incrementan en 1 y se inicializan en 0.

Relojes lógicos

- Por inducción en la longitud de cualquier secuencia de sucesos relacionando dos sucesos e y e', que $e \rightarrow e' \Rightarrow L(e) < L(e')$.
- No se puede inferir que si $L(e) < L(e') \Rightarrow e \rightarrow e'$.
- De la figura anterior también se puede observar que $L(b) > L(l)$, pero $b \parallel l$.

Relojes 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 se puede deducir que e *rightarrowe*$'$.

Reloj vectorial

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 las marcas de tiempo de los sucesos locales.

- Cada proceso adhiere el vector de marcas de tiempo en los mensajes que envía al resto.

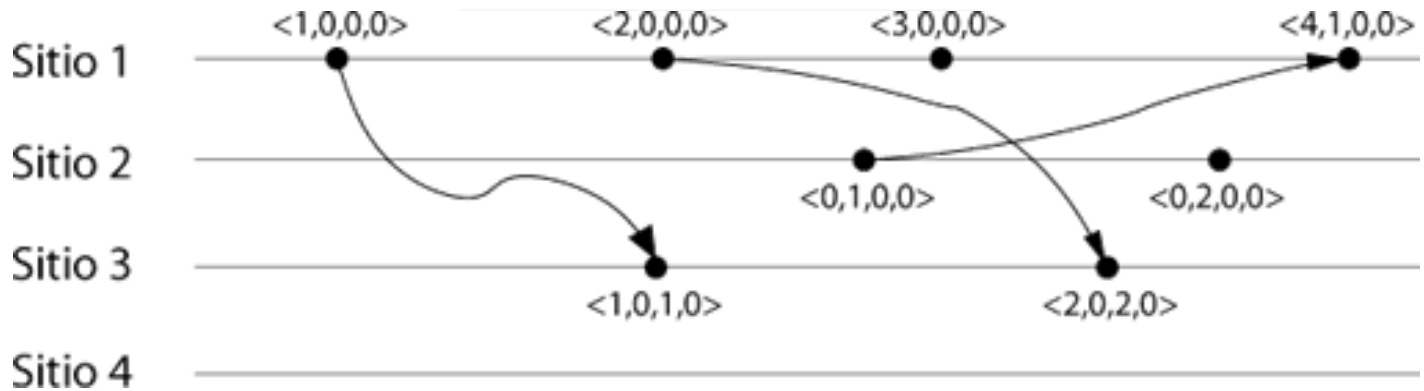
Relojes vectoriales

- Las reglas para actualizar un reloj vectorial son las siguientes:
  - RV1: inicialmente, $V_i[j] = 0$, para $i = 1, 2, \ldots, N$.
  - 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 $j = 1, 2, \ldots, N$.
- Para un vector V_i, $V_i[i]$ es el número de sucesos a los que p_i ha puesto una marca de tiempo.
- $V_i[j](j \neq i)$ es el número de sucesos que han ocurrido en p_j que han sido potencialmente afectados por p_i.

Relojes vectoriales

- Se pueden comparar vectores de marcas de tiempo como sigue:

1. $V = V'$ si y sólo si $V[j] = V'[j]$ para $j = 1, 2, \ldots, N$
2. $V \leq V'$ si y sólo si $V[j] \leq V'[j]$ para $j = 1, 2, \ldots, N$
3. $V < V'$ si y sólo si $V \leq V' \wedge V \neq V'$



Índice

1 Introducción

2 Relojes, eventos y estados de proceso

3 Sincronización de relojes físicos

4 Relojes lógicos

5 Exclusión mutua distribuida

Introduccón

- Los procesos distribuidos necesitan frecuentemente coordinar sus actividades.

Exclusión mutua

Si una colección de procesos comparte un recurso o una colección de recursos, entonces se requiere con frecuencia la exclusión mutua para prevenir enterferencias y asegurar la consistencia de los recursos.

- En un sistema distribuido no se pueden utilizar los mecanismos proporcionados por un núcleo, se requiere una solución basada exclusivamente en el paso de mensajes.

Introducción

- Considere un sistema de N procesos p_i, $i = 1, 2, ;N$ que no comparten variables.
- Los procesos acceden a recursos compartidos comunes.
- Suponga un sistema asíncrono, que los procesos no fallan y que la entrega de mensajes es fiable.

- El protocolo a nivel de aplicación para ejecutar una sección crítica es:
  1. entrar(): entrada a la sección crítica,bloquéese si es necesario.
  2. acceso_a_recursos(): acceso a un recurso compartido en la sección crítica.
  3. salir(): salida de la sección crítica, pueden entrar otros procesos.

Introducción

- El protocolo a nivel de aplicación para ejecutar una sección crítica es:
  1. entrar(): entrada a la sección crítica,bloquéese si es necesario.
  2. acceso_a_recursos(): acceso a un recurso compartido en la sección crítica.
  3. salir(): salida de la sección crítica, pueden entrar otros procesos.

- Los requisitos esenciales para la exclusión mutua son los siguientes:
  1. EM1(seguridad): a los sumo un proceso puede estar ejecutandose cada vez en la sección crítica.
  2. EM2(pervivencia): las peticiones para entrar y salir de la sección crítica al final son concedidas.
  3. EM3(ordenación): si una peticion para entrar a la sección crítica *ocurrió antes* que otra, la entrada se debe garantizar en ese orden.

Introducción

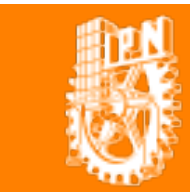
- El rendimiento de los algoritmos para la exclusión mutua se evalua con base en los siguientes criterios:
  1. El *ancho de banda* consumido es proporcional al número de mensajes enviados.
  2. El *retraso del cliente* en el que incurre un proceso en cada operación *entrar* y *salir*.
  3. El *retraso en la sincronización* entre un proceso que sale de la sección crítica y el siguiente proceso que entra en ella.

Suposiciones sobre fallos y detectores de fallos

- Se supone que cada par de procesos están conectados por canales fiables.
- Los procesos utilizan protocolos de comunicación fiables.
- Ningún fallo en un proceso implica una amenaza para la capacidad de otros procesos de comunicarse.

Algoritmo centralizado

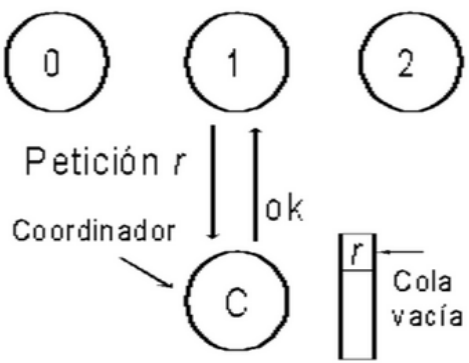
1. Para entrar en una sección crítica, un proceso envía un mensaje de petición al servidor y espera respuesta de su parte. Conceptualmente, la respuesta constituye un testigo que significa permiso para entrar en la sección crítica.
2. Si ningún proceso tiene el testigo en el instante de la petición, el servidor responde inmediatamente, otorganle el permiso.
3. Si en ese momento lo tiene otro proceso, entonces el servidor no responde sino que lo pone en una cola de petición.
4. Cuando un proceso sale de la sección crítica envía un mensaje al servidor, devolviéndole el testigo.

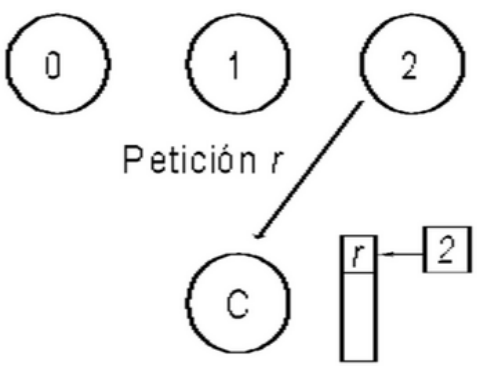


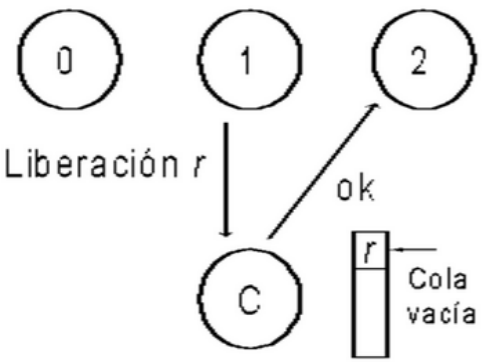
Algoritmo centralizado

La forma más simple

Utilizar un servidor que dé los permisos para entrar en la sección crítica es la forma más simple de conseguir exlusión mutua.





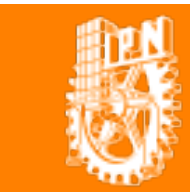


Algoritmo basado en un anillo

Anillo

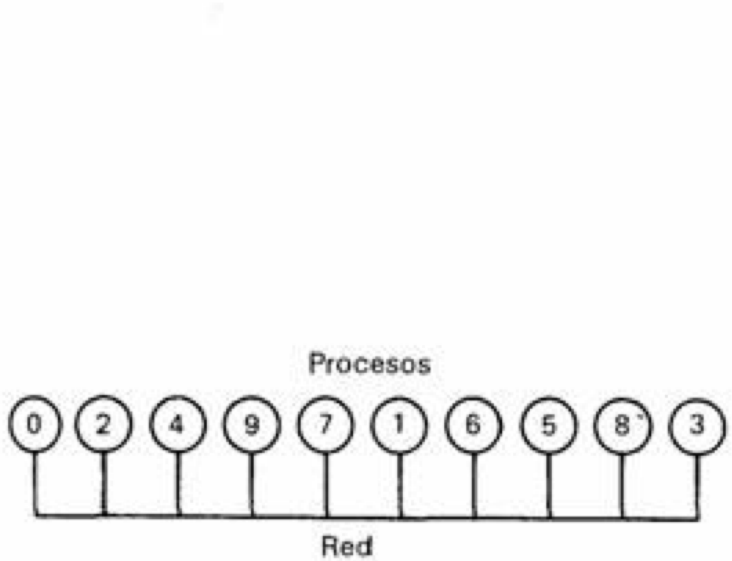
Una de las formas más simples de conseguir la esclusión mutua entre los N procesos sin que se necesite un proceso adicional es organizarlos en un anillo lógico.

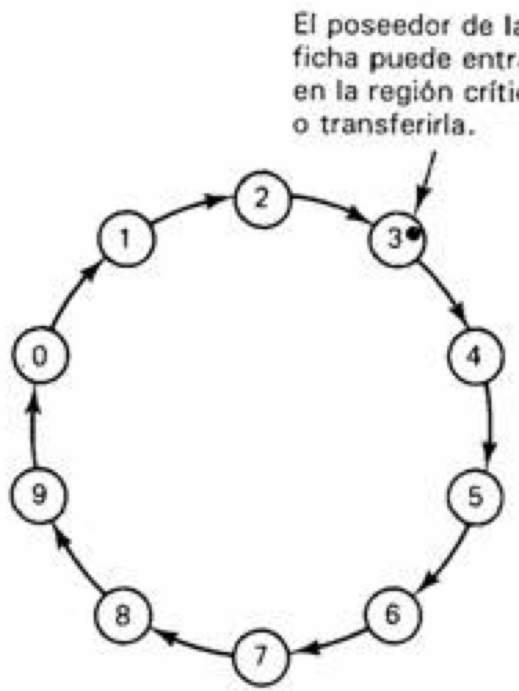
- Se requiere que cada proceso p_i tenga un canal de comunicación hacia el siguiente proceso del anillo $p_{(i+1)modN}$.
- La idea se basa en conseguir la exclusión obteniendo un testigo mediante un mensaje que se pasa de un proceso a otro en una única dirección alrededor del anillo.
- La topología en anillo no tiene por qué estar relacionada con las interconexiones físicas subyacentes.



Algoritmo basado en un anillo

Ejemplo





Algoritmo con multidifusión y relojes lógicos

Ricart y Agrawala[1981]

Desarrollaron un algoritmo para implementar la exclusión mutua entre N procesos de importancia pareja basado en la técnica de multidifusión.

- La idea básica es que los procesos que necesitan entrar en una sección crítica envían un mensaje de petición mediante multidifusión.
- Un proceso puede entrar en la región critica solamente cuando el resto de los procesos haya respondido al mensaje.
- Cada proceso p_i tiene un reloj lógico.
- Los mensajes de solicitud tienen la forma $< T, p_1 >$:
  - T es la marca de tiempo del emisor p_i.
  - p_i es el identificador del proceso que solicita el acceso.

Algoritmo con multidifusión y relojes lógicos

- Cada proceso tiene una variable de estado:
  1. *Liberada* : el proceso está fuera de la región critica.
  2. *Buscada* : el proceso quiere entrar a la región crítica.
  3. *Tomada* : el proceso está dentro de la región crítica.
- Si un proceso solicita acceso y el estado de los procesos es *Liberada*, todos los procesos responden inmediatamente a quien solicita el acceso y éste obtendra el acceso.
- Si algun proceso estuviese en el estado *Tomada* no responderá a las peticiones hasta que haya finalizado con la sección crítica.
- Si dos o más procesos solicitan la entrada al mismo tiempo, la petición que tenga la marca de tiempo más baja será la que consiga el acceso.

Algoritmo con multidifusión y relojes lógicos

Ejemplo