Sistemas Distribuidos

Módulo 3

Sincronización en Sistemas Distribuidos

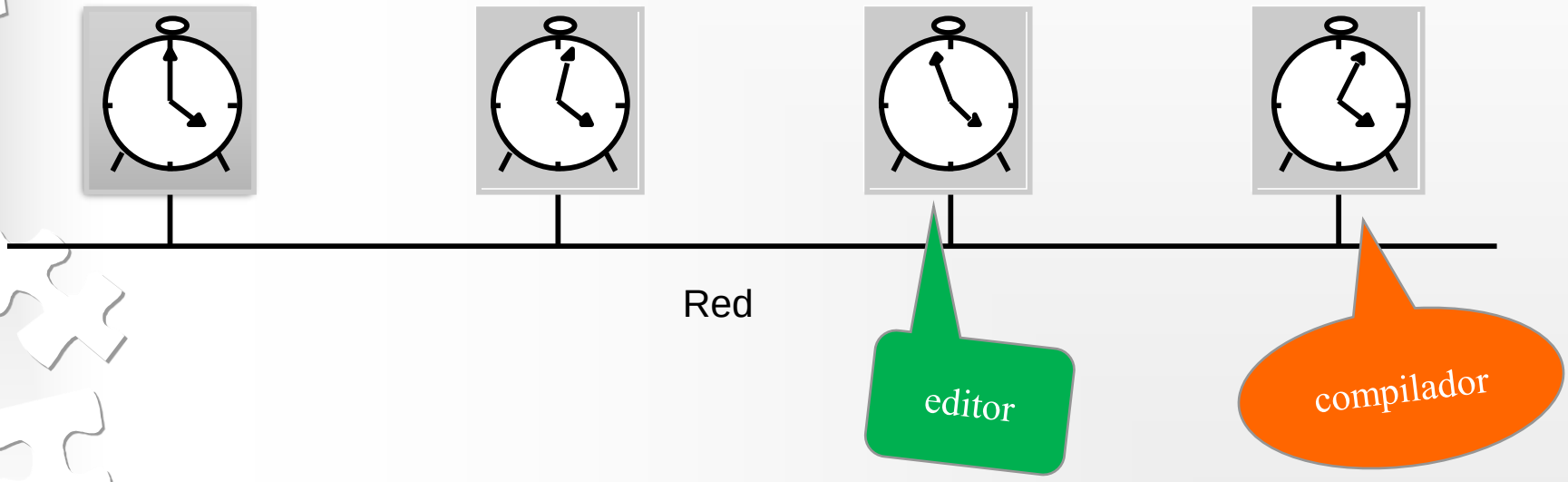
Agenda

1. Sincronización de Relojes
    1. Algoritmos
    2. Relojes lógicos
2. Estado Global
3. Exclusión Mutua
4. Algoritmos de Elección
5. Alcance de Acuerdos
6. Fallas de Comunicación y Procesos
7. Interbloqueos

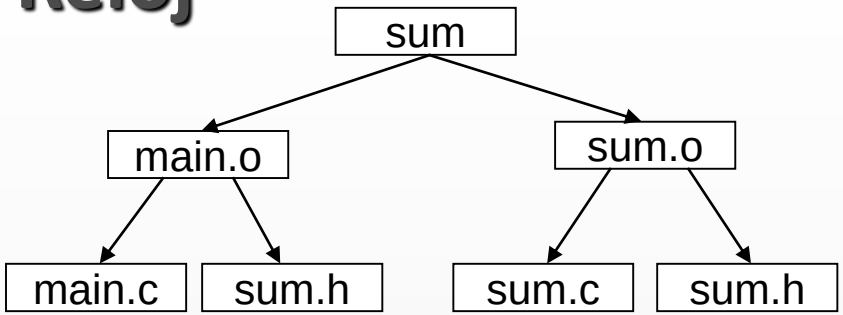
Agenda

1. Sincronización de Relojes
    1. Algoritmos
    2. Relojes lógicos
2. Estado Global
3. Exclusión Mutua
4. Algoritmos de Elección
5. Alcance de Acuerdos
6. Fallas de Comunicación y Procesos
7. Interbloqueos

Sincronización de Reloj

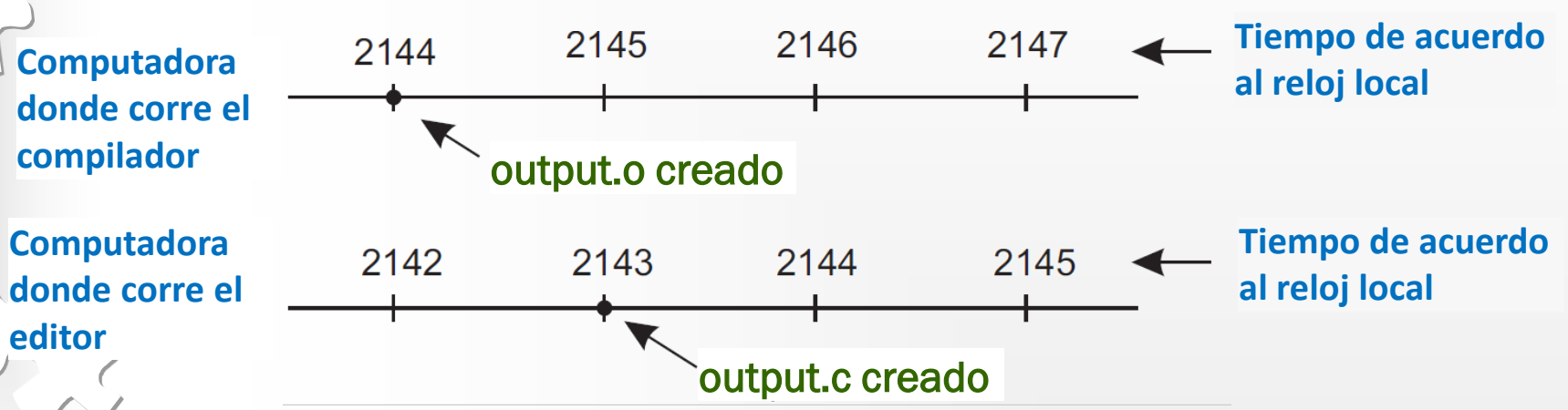


Sincronización de Reloj



Archivo	Última modificación	Nodo
sum	10:03	Compilador
main.o	09:56	Compilador
sum.o	09:35	Compilador
main.c	10:15 (10:02)	Editor
sum.c	09:14	Editor
sum.h	08:39	Editor

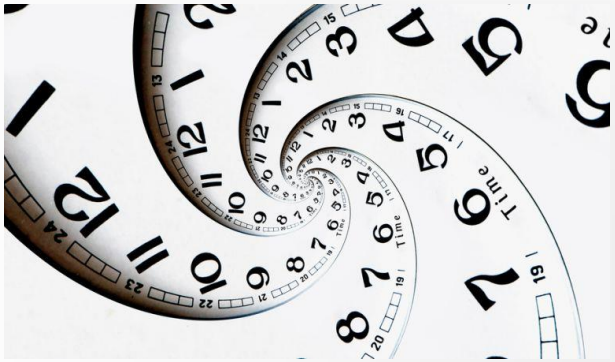
Sincronización de Reloj



Cuando cada máquina tiene su propio reloj, un evento que ocurre después de otro no puede ser asignado en tiempo anterior.

Sincronización de Reloj

¿Es posible sincronizar los relojes en un sistema distribuido?



Sincronización de Reloj

No es posible lograr ordenamiento de eventos si no existe una sincronización entre los relojes de diferentes sitios.

Implementación de Relojes: Se tiene un oscilador de cristal de cuarzo, un registro ***CONTADOR*** y un registro ***CONSTANTE***.

El registro *constante* almacena un valor dependiente de la frecuencia de oscilación del cristal.

Cuando registro ***CONTADOR*** llega a ***cero*** produce una interrupción ***tick*** *de reloj* (**clock tick**) en la computadora y el registro ***CONTADOR*** es reiniciado.

El reloj de la computadora debe sincronizarse con el tiempo real (relojes externos)

Sincronización de Reloj

Deriva de relojes

Con el pasaje del tiempo, el reloj de una computadora puede derivar respecto al tiempo real.

La deriva aproximada de un cristal es de:

1 seg cada 1000000 seg (11.6 días)

El valor del tiempo para un reloj *p* es $C_p(t)$.

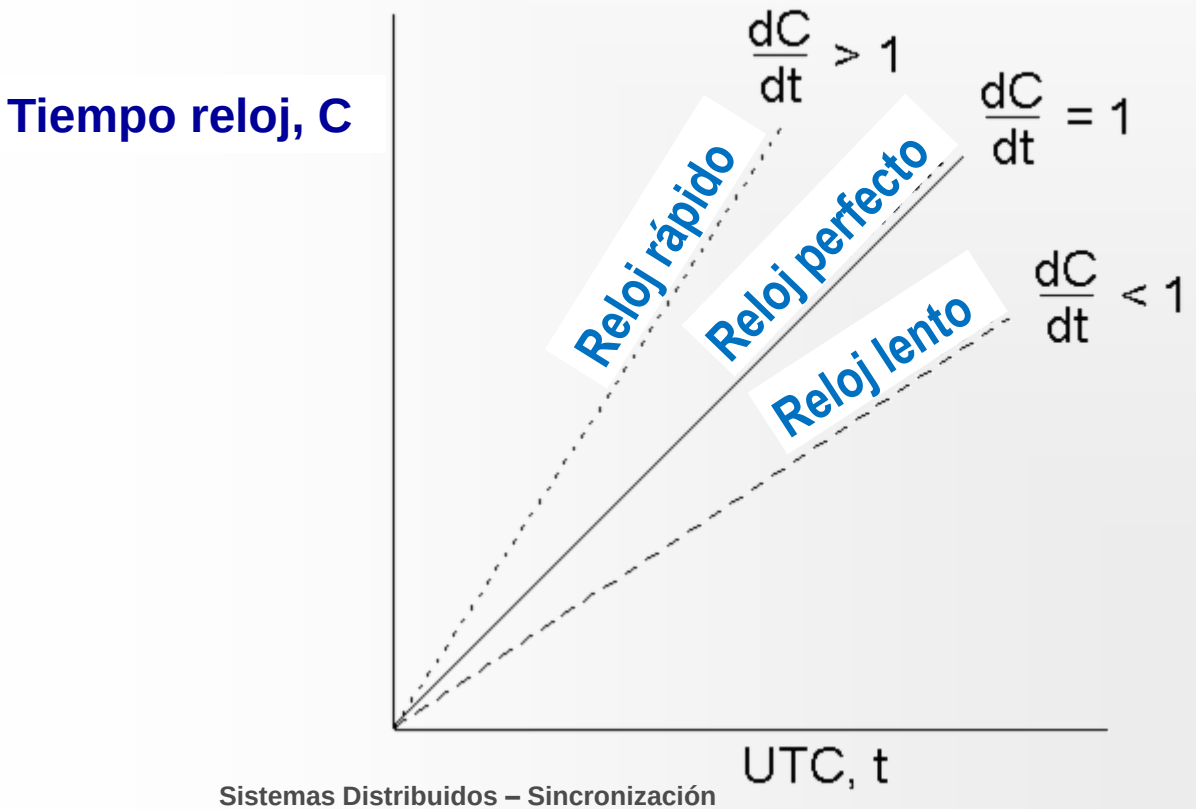
Si todos los relojes están sincronizados (caso ideal) entonces:

$$C_p(t)=t \; \forall p \text{ y } \forall t \Rightarrow dC/dt=1$$

Sincronización de Reloj

Si el ***máximo ritmo de deriva*** permitido es ρ, entonces se dice que el ***reloj funciona sin falla*** si: $1-\rho \leq dC/dt \leq 1+\rho$

Deriva de relojes



Sincronización de Reloj

Esta sincronización puede hacerse de dos maneras:

- **EXTERNA** - Sincronización de los relojes de las computadoras con relojes (externos) de tiempo real.
- **INTERNA** - Sincronización mutua de los relojes de los diferentes nodos del sistema.

Debe tenerse en cuenta que:

- Los retardos y cargas de las redes hacen impredecibles el cálculo del costo de los mensajes de actualización.
- El tiempo nunca debe ir hacia atrás porque puede causar serios problemas.

Sincronización de Reloj - Algoritmos

Algoritmos Centralizados

En este tipo de algoritmo, un nodo tiene el tiempo real (de cualquier fuente). Los algoritmos centralizados sufren:

- Único punto de falla
- Malo para la escalabilidad

Algoritmos Distribuidos

La sincronización se puede realizar por capas con servidores en cada una o entre los nodos pares.

Sincronización de Reloj

En **SISTEMAS SINCRÓNICOS** entre 2 procesos.

- p1 (emisor) envía tiempo ***t*** de su reloj en un mensaje ***m***
- p2 (receptor) actualiza su reloj al tiempo $t + T_{TRANS}$

T_{TRANS}:tiempo de transferencia

min: tiempo mínimo de transferencia

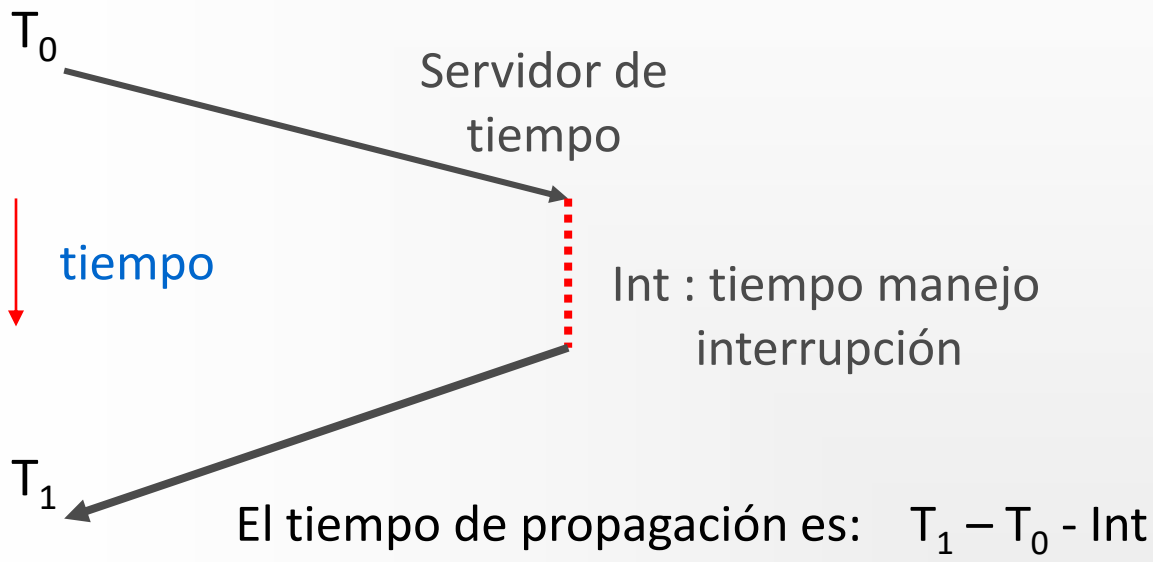
max: tiempo máximo de transferencia

u = (max-min)

t + (max-min)/2 con desvío de a lo sumo ***u/2***

Sincronización de Reloj - Algoritmos

Algoritmo Cristian con servidor de tiempo pasivo, 1989.



Sincronización de Reloj - Algoritmos

Algoritmo Centralizado con Servidor de Tiempo Activo

El servidor de tiempo activo, periódicamente hace un *broadcast* con su valor de tiempo y los nodos que lo reciben ajustan sus relojes considerando un retardo tipo.

Tiene algunos desajustes.

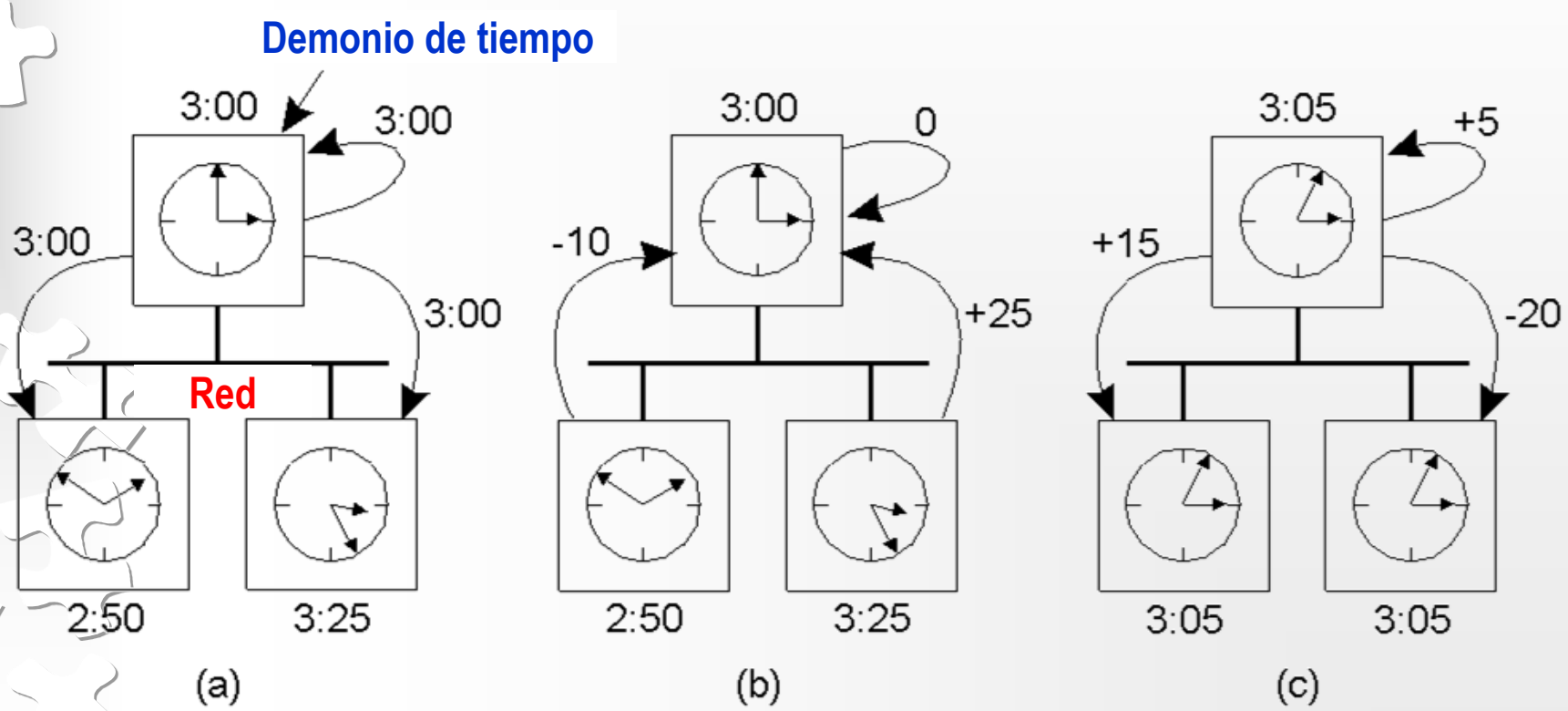
Sincronización de Reloj - Algoritmos

El **ALGORITMO BERKELEY** mejora el anterior.

1. El servidor envía un mensaje a cada uno de los nodos.
2. Estos le envían al servidor sus tiempos.
3. El servidor promedia todos (dentro de determinado intervalo, incluido el propio). Este es el valor al cual se tienen que ajustar todos los relojes. El servidor envía solo la diferencia que los otros nodos deben usar para corregir sus relojes.

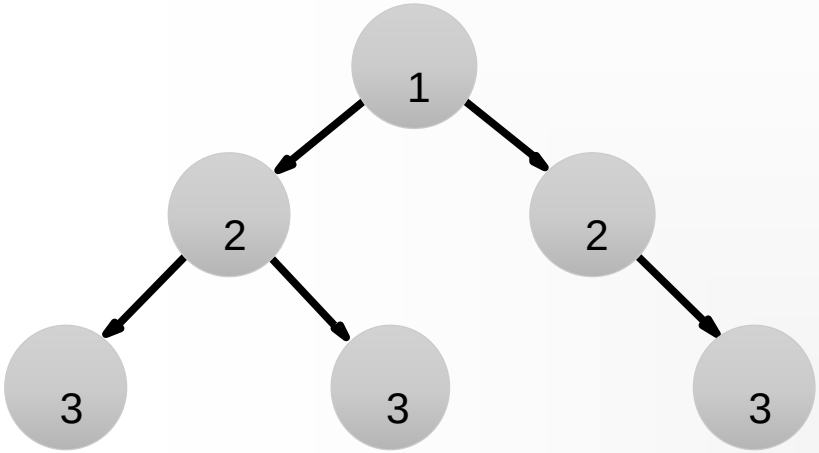
Sincronización de Reloj - Algoritmos

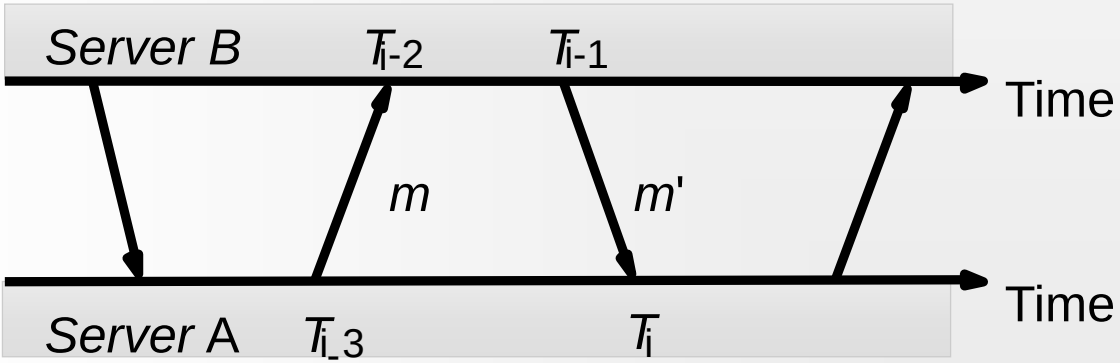
El **ALGORITMO BERKELEY** mejora el anterior:



Sincronización de Reloj - Algoritmos

Network Time Protocol





$$d_i = T_{i-2} - T_{i-3} + T_i - T_{i-1}$$

$$o_i = (T_{i-2} - T_{i-3} + T_{i-1} - T_i) / 2$$

Sincronización de Reloj - Algoritmos

ALGORITMOS DISTRIBUIDOS CON PROMEDIO GLOBAL.

Cada nodo hace un broadcast con su **tiempo local** a todos los demás, cuando el tiempo local es T_0+iR para algún entero i, donde T_0 es un tiempo fijo en el pasado y R es un parámetro del sistema que depende de factores tales como el número de nodos, la máxima deriva permitida, etc.

Cada nodo promedia y establece su tiempo.

Sincronización de Reloj - Algoritmos

ALGORITMO DISTRIBUIDO CON PROMEDIO LOCALIZADO.

Los globales no escalan bien.

Se agrupan nodos y se calculan tiempos en ellos como si fueran globales.

La agrupación puede ser en anillo, solamente se consulta a los vecinos.

Sincronización de Reloj – Relojes Lógicos

Relojes Lógicos

Problema: no hay reloj común

Solución: relojes lógicos

Ordenamiento: Lamport: ***"sucede antes"***

1) Si ***a*** y ***b*** son eventos en el mismo proceso y ***a*** ocurre antes que ***b***; entonces

$$a \rightarrow b \text{ es verdadero}$$

2) Si ***a*** es el evento de envío de mensaje por un proceso y ***b*** el de recepción del mismo mensaje por otro proceso ; entonces:

$$a \rightarrow b \text{ es verdadero}$$

Sincronización de Reloj – Relojes Lógicos

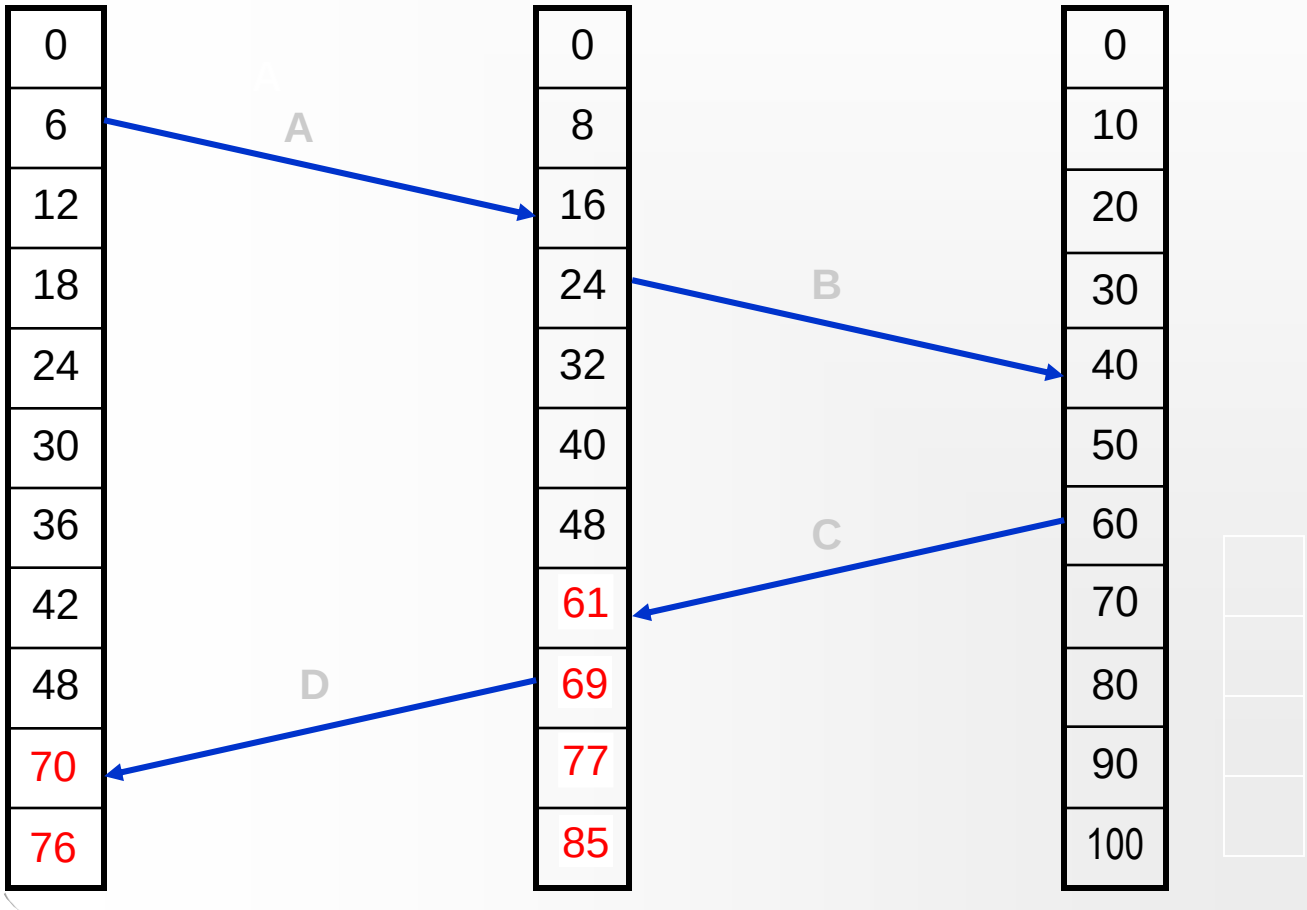
Algoritmo de Lamport

Se asocia un contador **C** a un evento *a* o *b*: **C(a), C(b)**

1. Si *a* sucede antes que *b* en el mismo proceso, **C(a) < C(b)**
2. Si a y b representan el envío y recepción de un mensaje, **C(a) < C(b)**
3. Para otros eventos *a* y *b*, **C(a) ≠ C(b)**

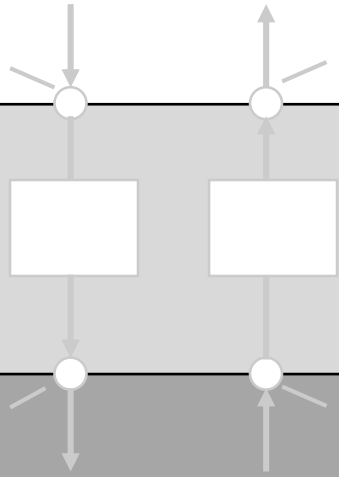
Sincronización de Reloj – Relojes Lógicos

Algoritmo de Lamport



Sincronización de Reloj - Relojes Lógicos

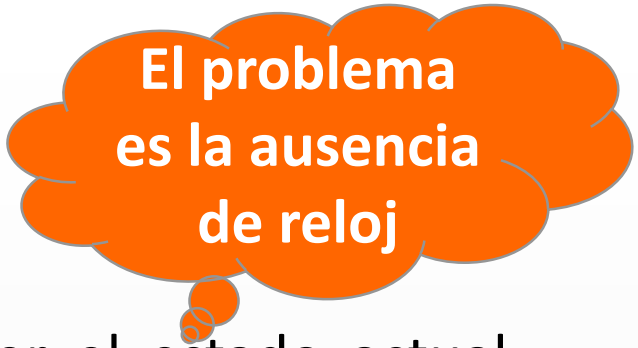
Algoritmo de Lamport

Capa de Aplicación		
Aplicación envía msg		Msg es entregado a la aplicación
Ajustar el reloj local y estampilla msg		Ajustar el reloj local
Capa de Middleware		
Middleware envía el msg		El msg es recibido
Capa de Red		

Agenda

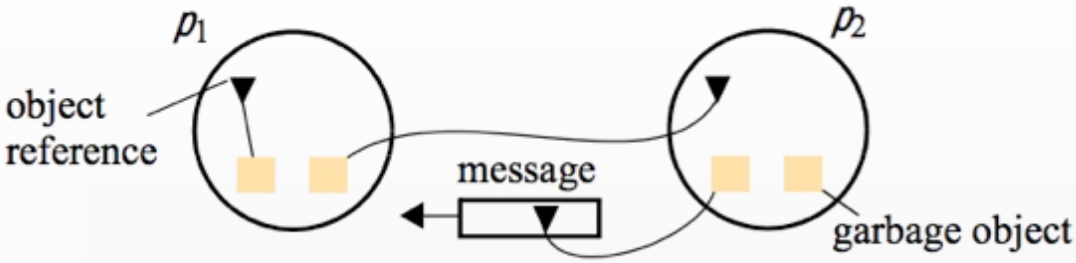
1. Sincronización de Relojes
    1. Algoritmos
    2. Relojes lógicos
2. Estado Global
3. Exclusión Mutua
4. Algoritmos de Elección
5. Alcance de Acuerdos
6. Fallas de Comunicación y Procesos
7. Interbloqueos

Estado Global Distribuido

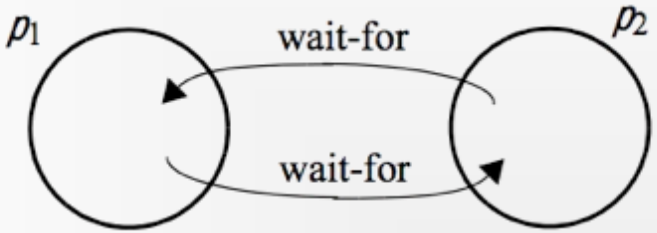


- El sistema operativo no puede conocer el estado actual (corriente) de todos los procesos en el sistema distribuido.
- Un proceso solo puede conocer los estados corrientes de todos los procesos en el sistema local.
- Los procesos remotos solo conocen la información sobre estados que es proporcinada por los mensajes recibidos
  - Generalmente estos mensajes presentan información de estado del pasado.

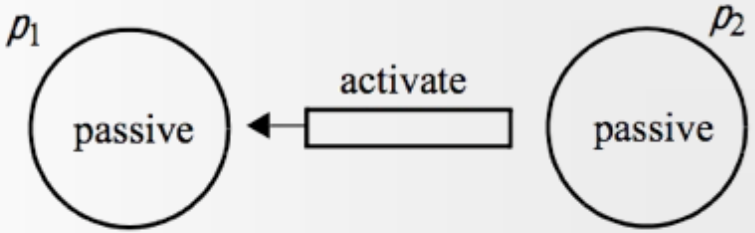
Estado Global Distribuido - Propiedades



(a) Garbage collection



(b) Deadlock

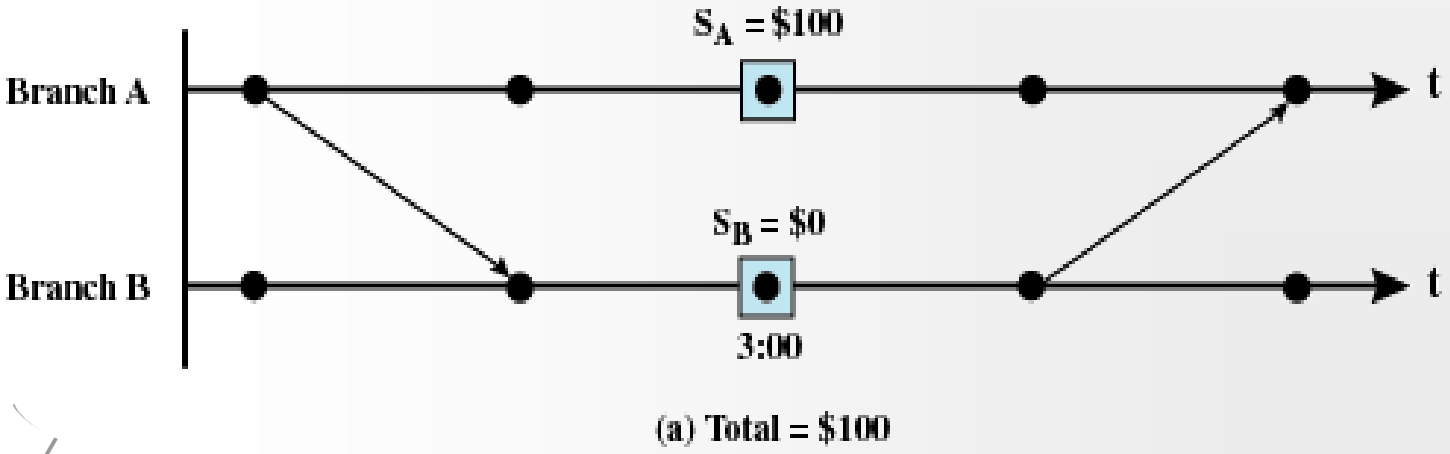


(c) Termination

Estado Global Distribuido

Ejemplo

- Una cuenta de banco está distribuida sobre dos sucursales.
- La cantidad total de la cuenta es la suma de cada sucursal.
- El balance de la cuenta es determinado a las 3 PM.
- Para requerir la información se envían mensajes

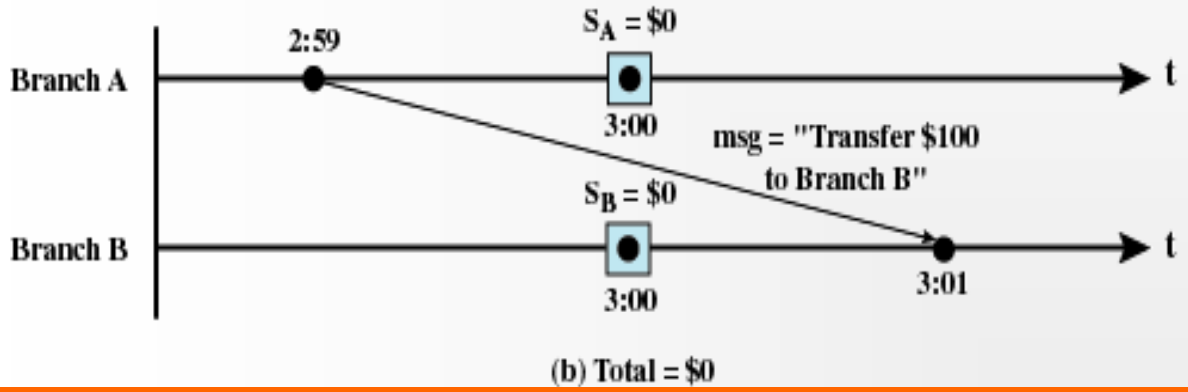


(a) Total = $100

Estado Global Distribuido

Ejemplo

- Si al momento de la determinación del balance, el balance de la sucursal A está en tránsito a la sucursal B,
- El resultado es una lectura falsa.



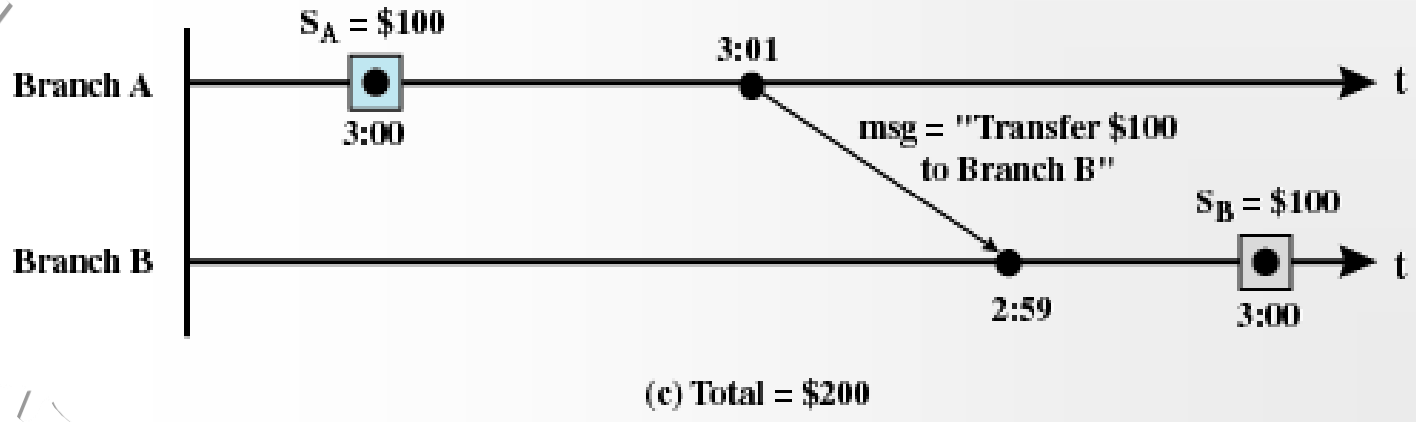
(b) Total = $0

- Todos los mensajes en tránsito deben ser examinados en el tiempo de observación
- Debe haber consistencia total del balance de ambas sucursales y la cantidad en el mensaje.

Estado Global Distribuido

Ejemplo

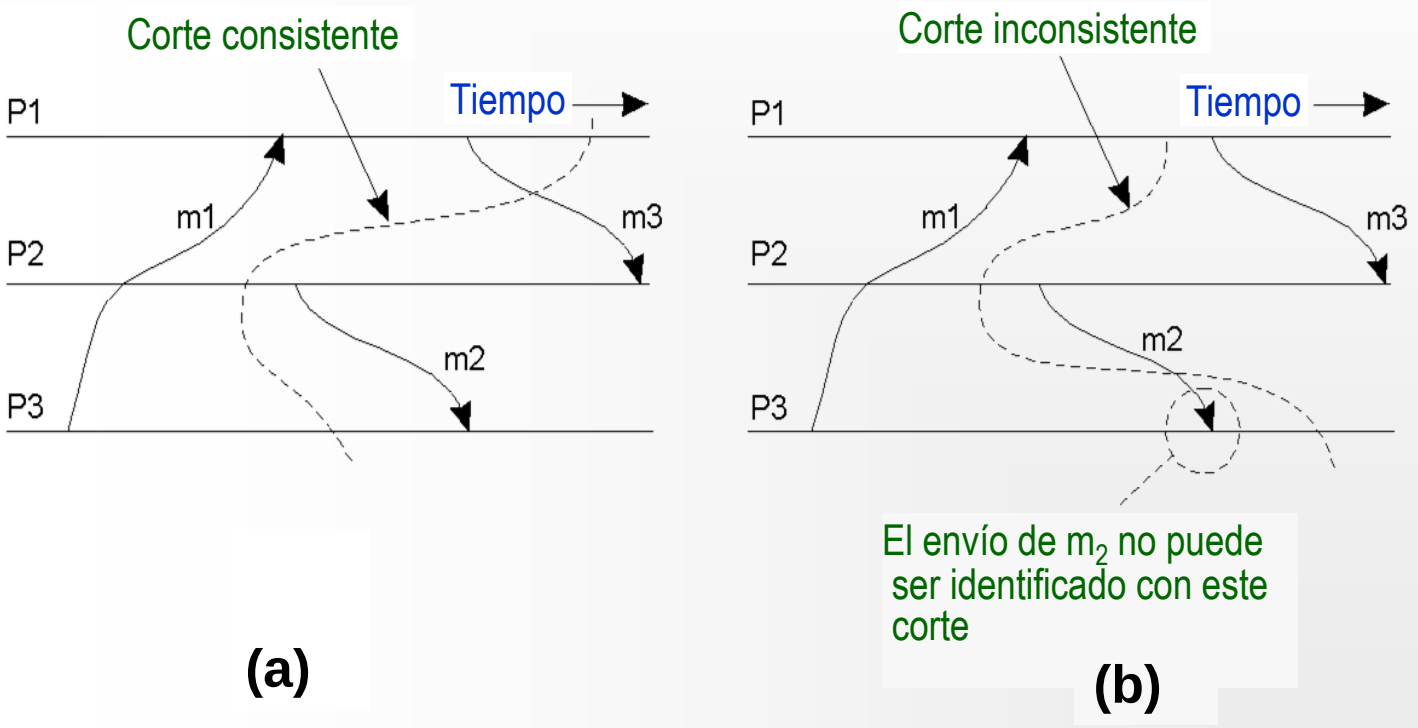
- Si los relojes de las dos sucursales no están perfectamente sincronizados.
- Desde la sucursal A se transfiere el monto a las 3:01.
- El monto llega a la sucursal B a las 2:59 (hora de B)
- A las 3:00 la cantidad es contada dos veces.



(c) Total = $200

Estado Global Distribuido

Estado Global: Problema



a) Un corte consistente

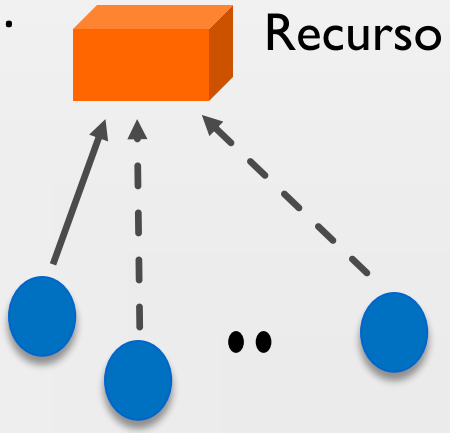
b) Un corte inconsistente

Agenda

1. Sincronización de Relojes
    1. Algoritmos
    2. Relojes lógicos
2. Estado Global
3. Exclusión Mutua
4. Algoritmos de Elección
5. Alcance de Acuerdos
6. Fallas de Comunicación y Procesos
7. Interbloqueos

Exclusión Mutua Distribuida

- Suposiciones
  - El sistema consiste de *n* procesos; cada proceso P_i reside en un procesador diferente.
  - Cada proceso tiene una sección crítica que requiere exclusión mutua.
- Dificultades que deben enfrentarse cuando se diseña el protocolo:
  - Interbloqueo o “abrazo mortal” (deadlock).
  - Inanición (starvation)



Exclusión Mutua Distribuida

- Requerimiento
  - Exclusión. Si P_i está ejecutando en su sección crítica (SC), entonces no hay otro proceso ejecutando en su SC.
  - Progreso.
    - Si varios procesos están esperando para entrar en la SC, mientras ninguno de ellos está en la misma, alguno de ellos deberá entrar en un tiempo finito.
    - El comportamiento de un proceso fuera de la SC o del protocolo que gobierna el acceso, no tiene influencia sobre el protocolo de exclusión mutua (hay independencia).
  - Espera Limitada. No existe proceso privilegiado.

Exclusión Mutua Distribuida

La exclusión mutua en un ambiente distribuido puede ser conseguida mediante dos familias de algoritmos:

- Basados en permiso
- Basados en ficha (token)

Como caso especial se considera la forma centralizada dentro de un ambiente distribuidos

Exclusión Mutua Distribuida

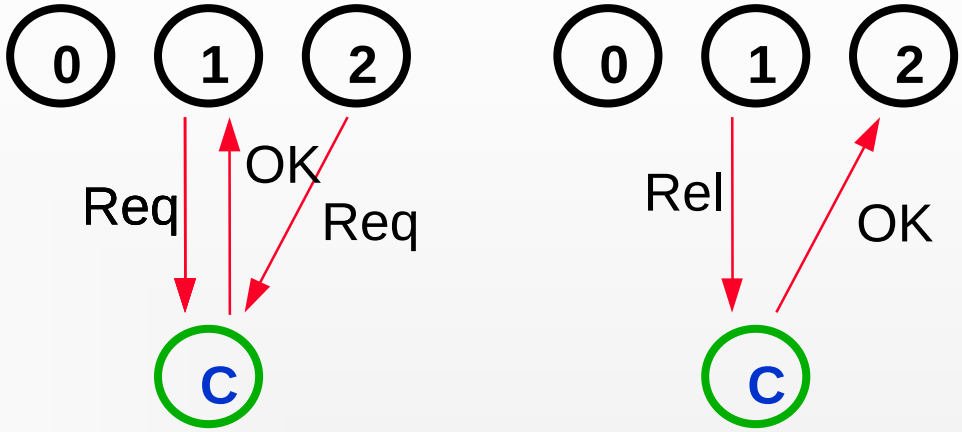
Algoritmos centralizados para Exclusión Mutua

- Un nodo es asignado como **nodo de control**.
- Este nodo de control accede a todos los objetos compartidos.
- Solo el nodo de control toma decisiones sobre la alocación de los recursos compartidos.
- Toda la información necesaria es concentrada en el nodo de control.
- Si el nodo de control falla, la exclusión mutua se cae.

Exclusión Mutua Distribuida – Algoritmo Centralizado

VENTAJAS

- No hay inanición¿?
- No hay bloqueo
- Fácil de implementar



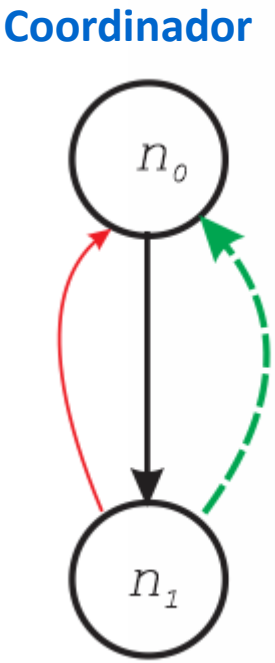
DESVENTAJAS

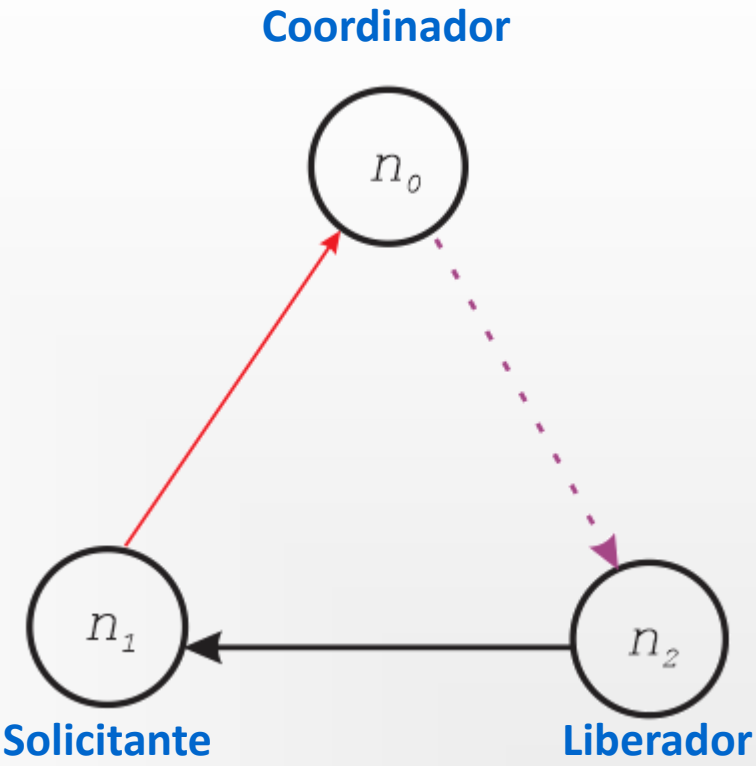
- Único punto de falla
- El coordinador es un “cuello de botella”

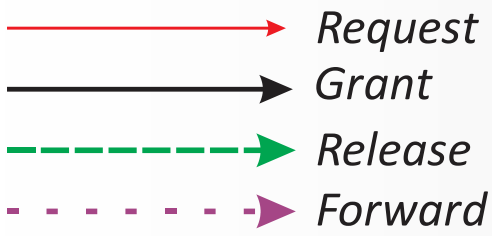
COMPLEJIDAD

Tres mensaje por entrada en la sección crítica

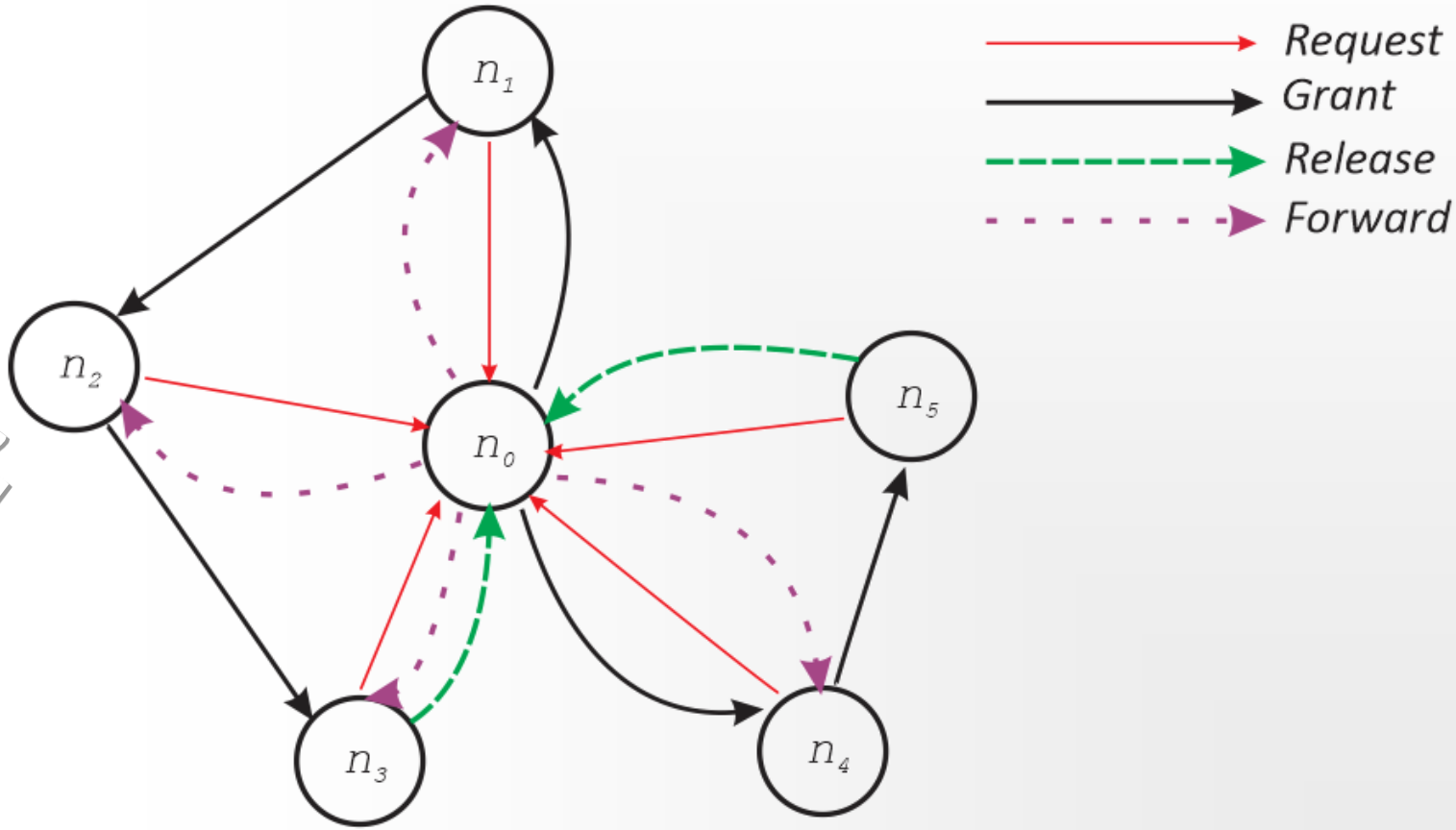
Exclusión Mutua Distribuida – Algoritmo Centralizado







Exclusión Mutua Distribuida – Algoritmo Centralizado



Exclusión Mutua Distribuida

Algoritmos Distribuidos - Características

En general todos los algoritmos distribuidos debieran cumplir con las siguientes pautas:

- Todos los nodos tiene igual cantidad de información, en promedio.
- Cada nodo tiene solo una visión parcial del sistema total y debe tomar decisiones en base a esta información.
- Todos los nodos tienen igual responsabilidad sobre la decisión final.

Exclusión Mutua Distribuida

Algoritmos Distribuidos - Características

- Todos los nodos realizan el mismo esfuerzo, en promedio, para llegar a la decisión final.
- Falla en un nodo, en general, no resulta en un colapso total del sistema.
- No existe un reloj común con el cual regular el tiempo de los eventos.

Exclusión Mutua Distribuida

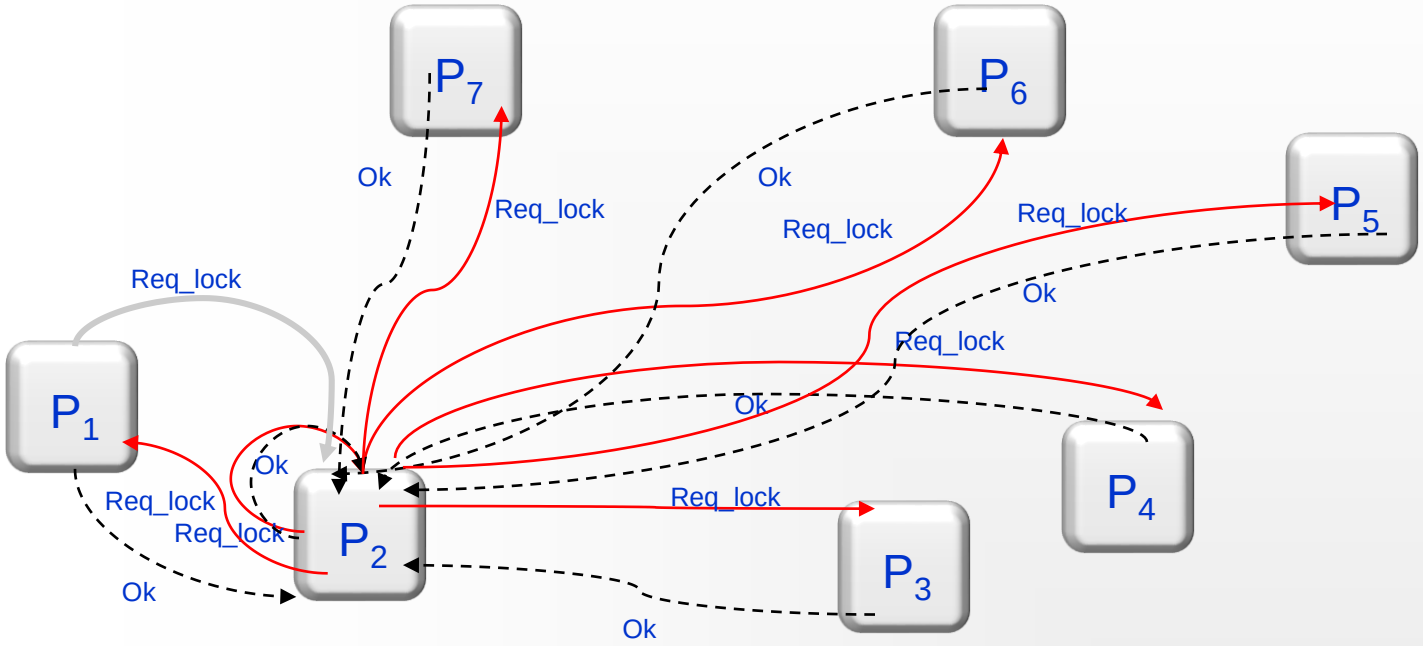
Algoritmo Distribuido Ricart-Agrawala

El proceso que quiere entrar en su ***sección crítica*** envía un mensaje a los demás *n-1* procesos y a sí mismo

Posibles respuestas de los ***n-1*** procesos restantes:

- Si el receptor no está en su sección crítica envía un OK al proceso emisor.
- Si el receptor está en la sección crítica no contesta. Pone el pedido en cola.
- Si quiere entrar en la sección crítica compara las estampillas de los mensajes, la menor (más antigua) gana.

Exclusión Mutua Distribuido: Ricart-Agrawala



Exclusión Mutua Distribuida: Ricart-Agrawala

El número de mensajes es ***2(n-1)*** para ingresar

DESVENTAJA: hay *n* puntos de fallas

- Está libre de interbloqueos.
- Está libre de inanición dado que la entrada en la sección crítica está planificada de acuerdo a un ordenamiento basado en estampillas de tiempo.

Exclusión Mutua Distribuida Ricart - Agrawala

Consecuencias indeseables

- Los procesos necesitan conocer la identidad de todos los otros procesos en el sistema, lo cual hace el agregado y remoción dinámica de procesos más complejo.
- Si uno de los procesos falla el esquema completo colapsa. Esto puede manejarse con un continuo monitoreo del estado de todos los procesos del sistema.
- Este protocolo es adecuado para conjuntos pequeños y estables de procesos cooperativos.

Exclusión Mutua Distribuida - Maekawa

Este algoritmo está basado en permisos, obteniendo los permisos de un quorum de procesos.

Los objetivos de este algoritmo es reducir la cantidad de mensajes necesarios para el ingreso en la sección crítica.

Condiciones para el armado del Quorum

- $P_i \in V_i$
- $V_i \cap V_j \neq \emptyset$
- $|V_i| = k$
- Cada proceso pi está contenido en M de los conjuntos de quorums V_i

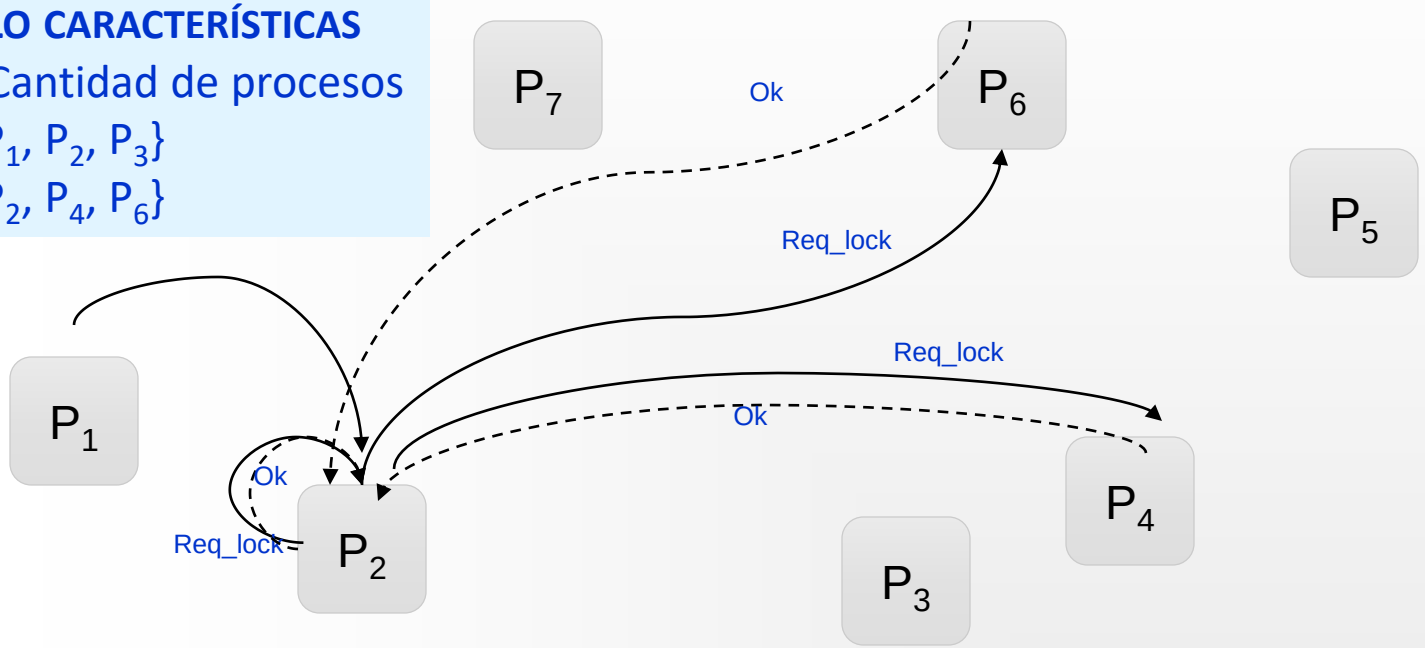
Exclusión Mutua Distribuida - Maekawa

EJEMPLO CARACTERÍSTICAS

N=7 Cantidad de procesos

S1={P_1, P_2, P_3}

S2={P_2, P_4, P_6}



Para ingresar a la sección crítica requiere ***2(k-1)*** mensajes, donde k es la cantidad de miembros del quorum.

Para salir de la sección crítica requiere ***(k-1)*** mensajes.

Exclusión Mutua Distribuida

Algoritmos de Pasaje de Ficha

- Se pasan una ficha (token) entre los procesos participantes.
- La ficha es una entidad que en algún momento es retenida por un proceso.
- El proceso que retiene la ficha puede entrar a su sección crítica sin pedir permiso.
- Cuando un proceso deja su sección crítica, pasa la ficha a otro proceso.

Exclusión Mutua Distribuido

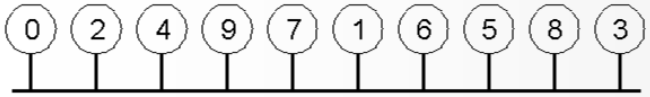
Algoritmo de Pasaje de Ficha en Anillo

Los procesos tienen una conexión lógica en anillo.

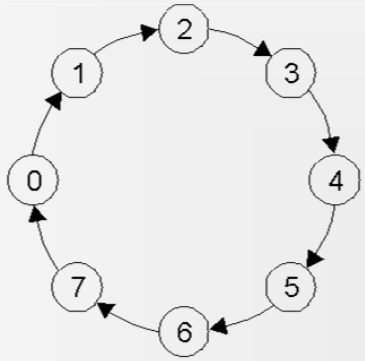
Una ficha recorre los procesos en un solo sentido en forma circular.

El proceso que quiere entrar en su ***sección crítica*** espera tener la ficha y la retiene mientras procesa su sección crítica.

Desventaja: puede perderse la ficha o caer un proceso.
Ventaja: no hay inanición.



(a)



(b)

a) Un grupo de procesos sin orden en la red. b) Un anillo lógico construido en software.

Exclusión Mutua Distribuida

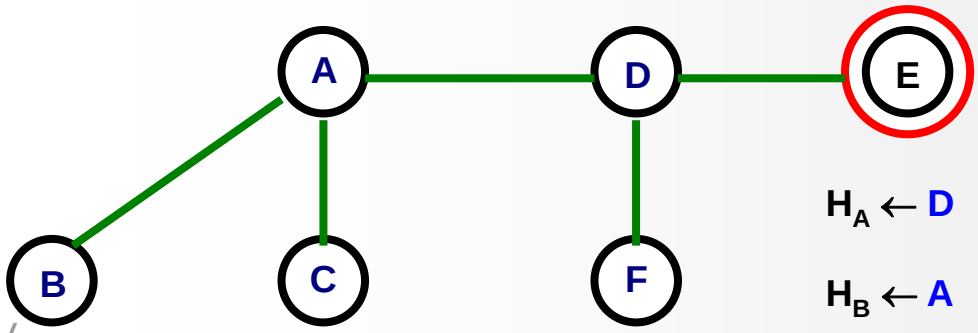
Algoritmo Distribuido para Exclusión Mutua basado en Topología Arbórea

(Kenny-Raymond, 1981)

La posesión del *ticket* implica permiso para entrar en la sección crítica.

Cada nodo se comunica con el vecino solamente.

Cada nodo tiene la ubicación del ticket H_i.



$H_A \leftarrow D$ $H_C \leftarrow A$ $H_E \leftarrow$ "self"

$H_B \leftarrow A$ $H_D \leftarrow E$ $H_F \leftarrow D$

Agenda

1. Sincronización de Relojes
    1. Algoritmos
    2. Relojes lógicos
2. Estado Global
3. Exclusión Mutua
4. Algoritmos de Elección
5. Alcance de Acuerdos
6. Fallas de Comunicación y Procesos
7. Interbloqueos

Algoritmos de Elección

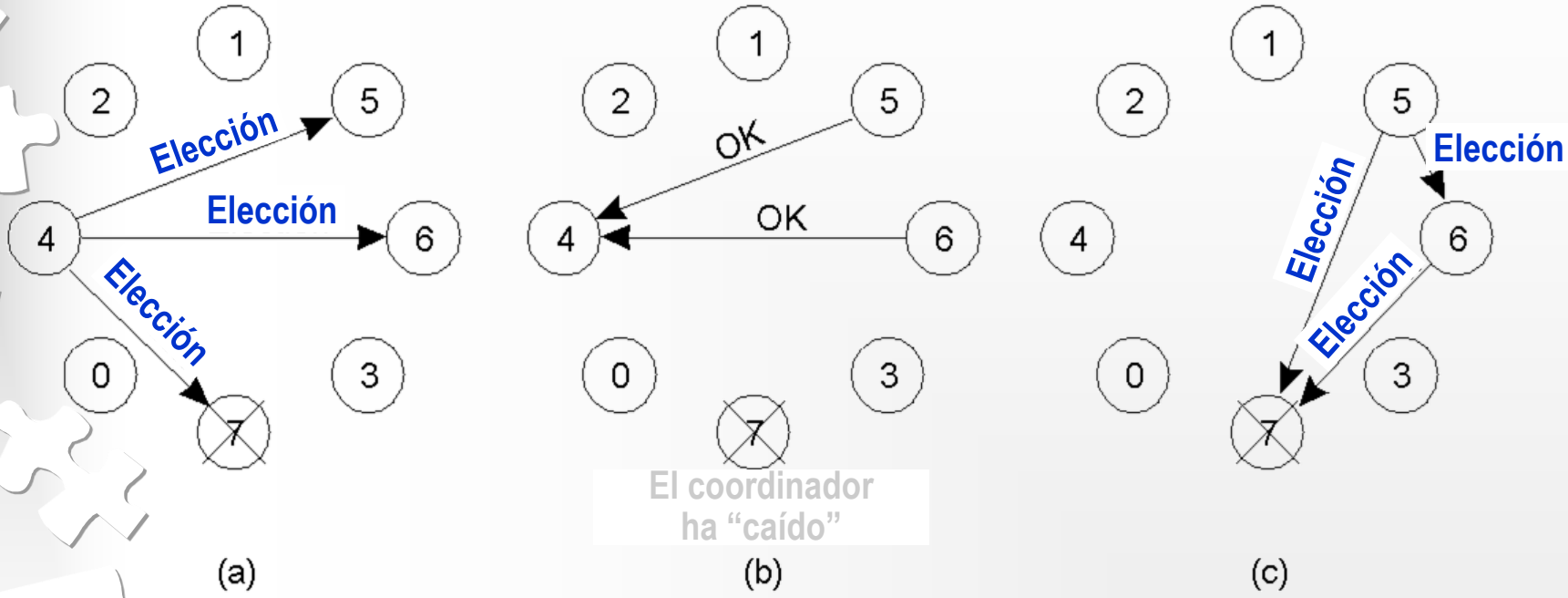
Cuando el coordinador no responde a los requerimientos, se inicia un proceso de elección.

ALGORITMO DE "BULLY" (GARCÍA-MOLINA)

P_i envía un mensaje ELECCION a cada proceso con número mas grande.

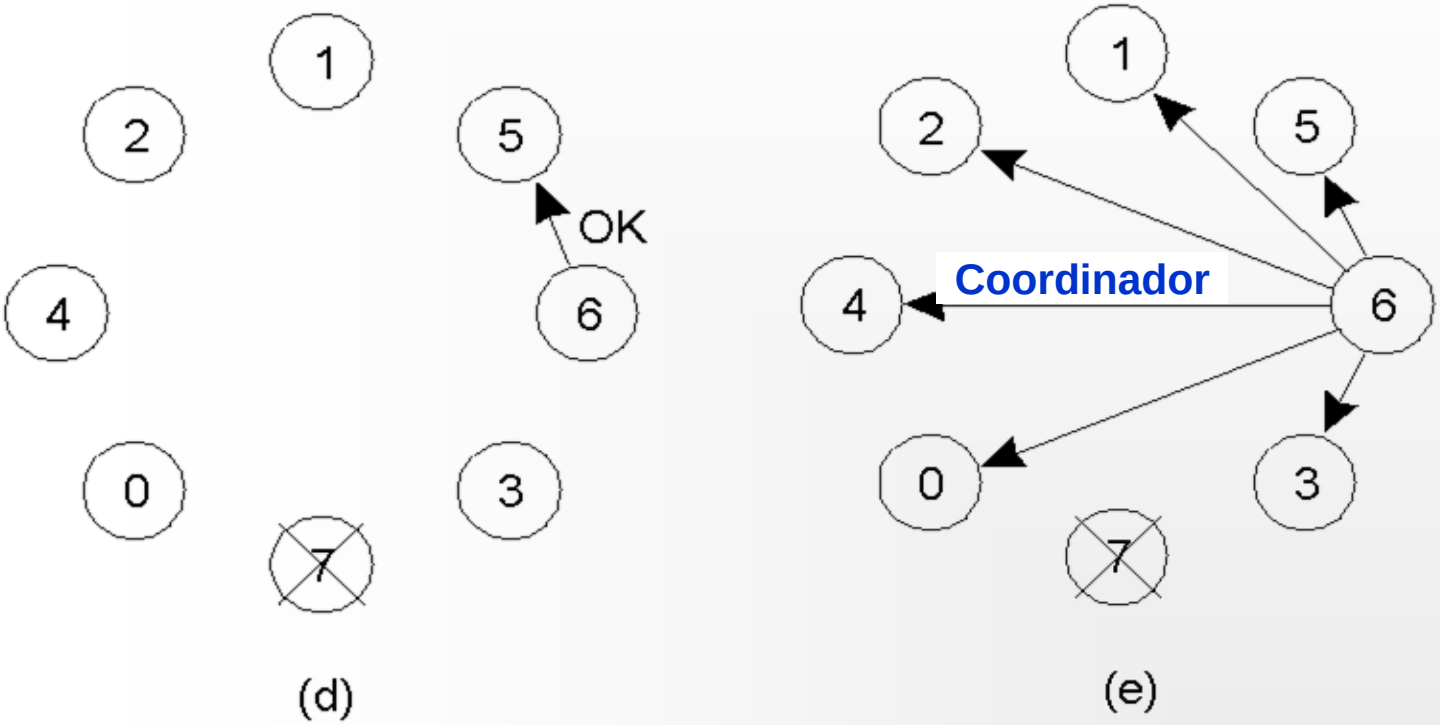
- Si no responde nadie P_i gana la elección.
- Si un proceso mayor P_j responde, P_i queda afuera y P_j *broadcast* que él es el coordinador.

Algoritmos de Elección



a) El proceso 4 inicia una elección
b) Los procesos 5 y 6 responden, entonces 4 para
c) Ahora 5 and 6, cada uno, inicia una elección

Algoritmos de Elección



d) El proceso 6 le dice al 5 que pare

e) El proceso 6 gana y la avisa a todos

Algoritmos de Elección

Algoritmo del anillo

Cada proceso conoce su sucesor (vecino en el anillo)

Algún proceso detecta que el coordinador no funciona entonces envía un mensaje *elección* a su sucesor, incorporando su número,.

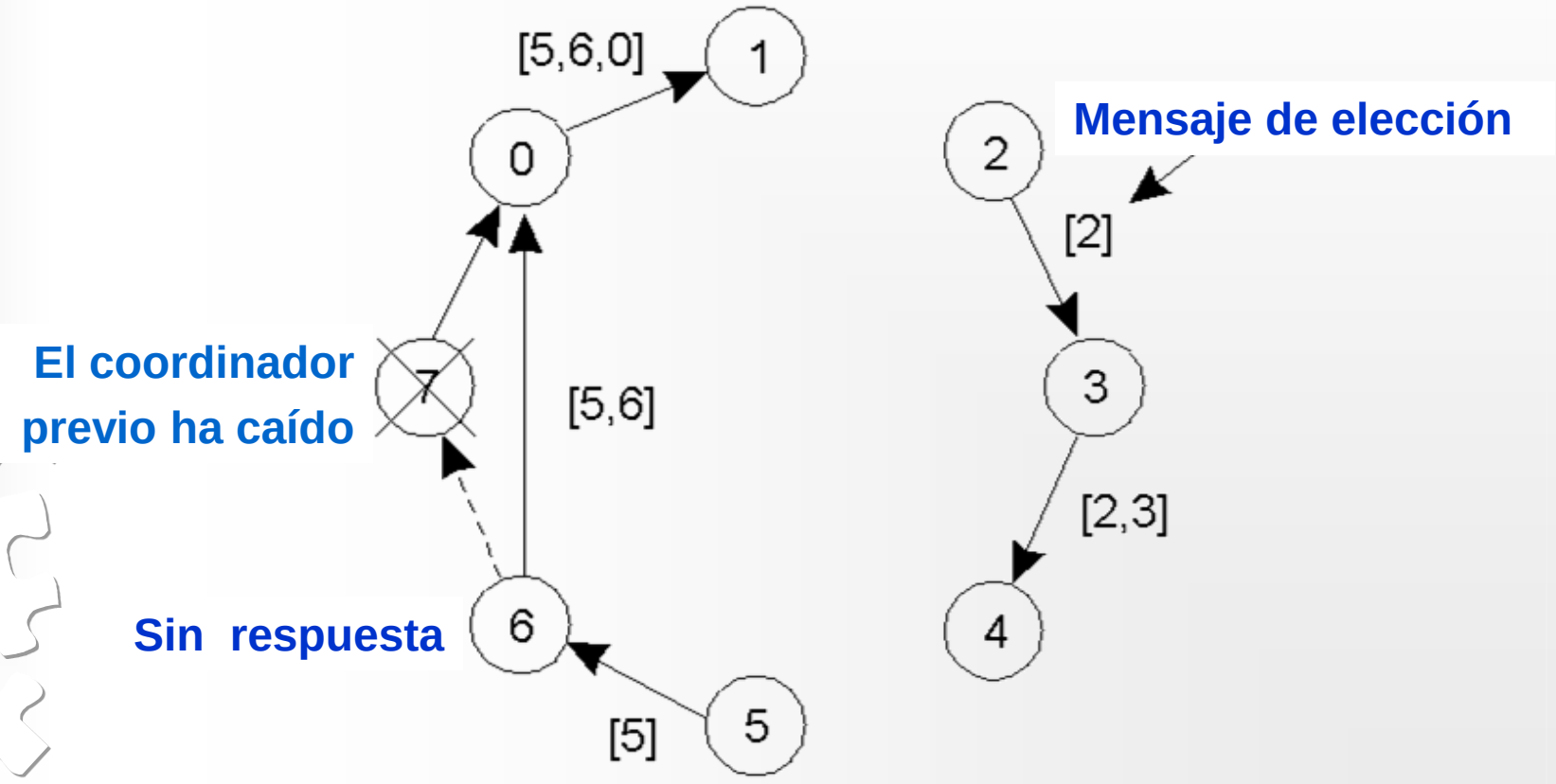
En cada paso cada proceso agrega su número.

El mensaje vuelve (lo reconoce porque está su propio número).

El siguiente mensaje es con el nombre del coordinador.

Algoritmos de Elección

ALGORITMO DE ANILLO



Agenda

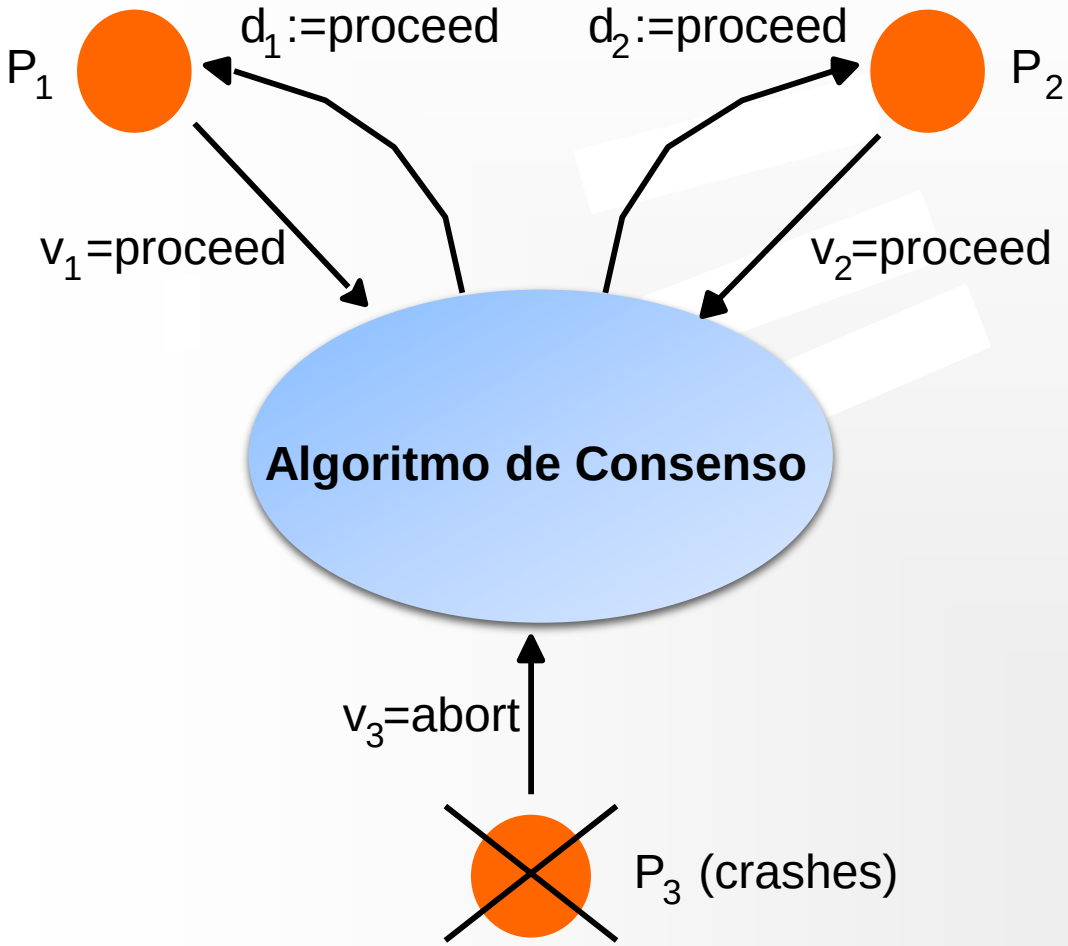
1. Sincronización de Relojes
    1. Algoritmos
    2. Relojes lógicos
2. Estado Global
3. Exclusión Mutua
4. Algoritmos de Elección
5. Alcance de Acuerdos
6. Fallas de Comunicación y Procesos
7. Interbloqueos

Alcance de Acuerdo

- Hay aplicaciones donde un conjunto de procesos desea acordar sobre un "**valor**" común.

- Tales acuerdos pueden no tener lugar debido a:
  - Fallas en el medio de comunicación
  - Procesos que fallan:
    - Los procesos pueden enviar mensajes incorrectos o mal escritos a otros procesos.
    - Un subconjunto de procesos puede colaborar con otro en un intento de derrotar al esquema.

Alcance de Acuerdo



Alcance de Acuerdo

Requerimientos para un algoritmo de consenso.

- **TERMINACIÓN**: Finalmente, cada proceso correcto establece su valor de decisión.
- **ACUERDO**: El valor de decisión es el mismo para todos los procesos correctos. Si p_i y p_j son correctos y están en el estado *decidido*, entonces $d_i = d_j$, (i, j = 1, 2, …, N).
- **INTEGRIDAD**: Si los todos los procesos correctos proponen el mismo valor, entonces cualquier proceso correcto en el estado *decidido* ha elegido ese valor.

Agenda

1. Sincronización de Relojes
    1. Algoritmos
    2. Relojes lógicos
2. Estado Global
3. Exclusión Mutua
4. Algoritmos de Elección
5. Alcance de Acuerdos
6. Fallas de Comunicación y Procesos
7. Interbloqueos

Acuerdo en Sistemas con Falla en las Comunicaciones

- El proceso P_i en el sitio *A*, ha enviado un mensaje al proceso P_j de *B*; para proceder, P_i necesita saber si P_j ha recibido el mensaje.
- Detección de fallas usando un esquema de time-out:
  - Cuando P_i envía un mensaje, también especifica un intervalo de tiempo durante el cual espera recibir un mesaje de ACK de P_j.
  - Cuando P_j recibe el mensaje, inmediatamente envía ACK a P_i.
  - Si P_i recibe el ACK dentro del tiempo especificado en el intervalo de tiempo, concluye que P_j ha recibido su mensaje. Si un time-out ocurre, P_j necesita retransmitir su mensaje y esperar por su ACK.
  - Continue hasta que P_i reciba un ACK, o sea notificado por el sistema que B está caído.

Acuerdo en Sistemas con Falla en las Comunicaciones

- Suponga que P_j también necesita saber que P_i ha recibido su mensaje de ACK, para decidir sobre como proceder.
  - En la presencia de falla, no es posible cumplir con esta tarea.
  - No es posible en un medio distribuido por los procesos P_i y P_j estar de acuerdo completamente sobre sus respectivos estados.

Acuerdo en Sistemas con Falla en Procesos

- El medio de comunicación es confiable, pero los procesos pueden fallar de modo impredictible.
- Considere un sistema de **n** procesos, de los cuales no más de **m** están fallados. Suponga que cada proceso P_i tiene un valor privado V_i.
- Se corre un algoritmo que permite que cada P_i no fallado construya un vector $X_i = (A_{i,1}, A_{i,2}, \ldots, A_{i,n})$ tal que:
  - Si P_j es un proceso no fallado, entonces $A_{ij} = V_j$.
  - Si P_i y P_j son procesos no fallados, entonces $X_i = X_j$.
- La solución comparte las siguientes propiedades.
  - Un algoritmo correcto puede ser corrido solo si $\boldsymbol{n \geq 3m + 1}$.
  - El retardo del peor caso para el alcance de acuerdo es proporcional a $m+1$ mensajes.

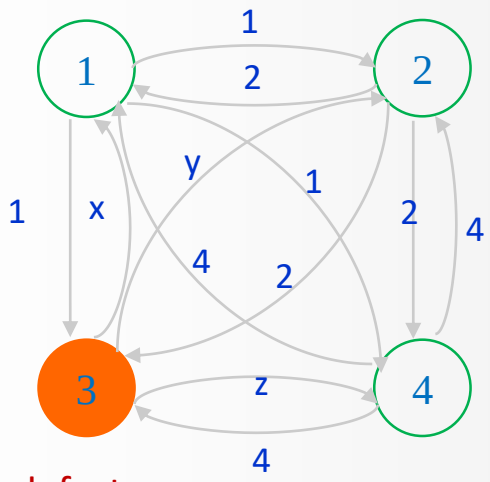
Acuerdo en Sistemas con Falla en Procesos

- Un algoritmo para el caso donde m=1 y n=4 requiere dos rondas de intercambio de información:
  - Cada proceso envía su valor privado a los otros 3 procesos.
  - Cada proceso envía la información obtenida en la primer ronda a todos los otros procesos.
- Si un proceso fallado no envía mensajes, un proceso no fallado puede elegir un valor arbitrario y pretender que ese valor fue enviado por ese proceso.
- Completadas las dos rondas, un proceso no fallado puede construir su vector $X_i = (A_{i,1}, A_{i,2}, A_{i,3}, A_{i,4})$ como sigue:
  - $A_{i,j} = V_j$.
  - Para $j \neq i$, si al menos dos de los tres valores reportados por el proceso P_j coinciden, entonces se usa el valor de mayoría para iniciar el valor A_{ij}. Sino se usa (*nil*).

Acuerdo en Sistemas con Falla en Procesos

Consideremos la siguiente situación: tres generales leales y un traidor. Tienen que llegar a un acuerdo sobre los valores propuestos.

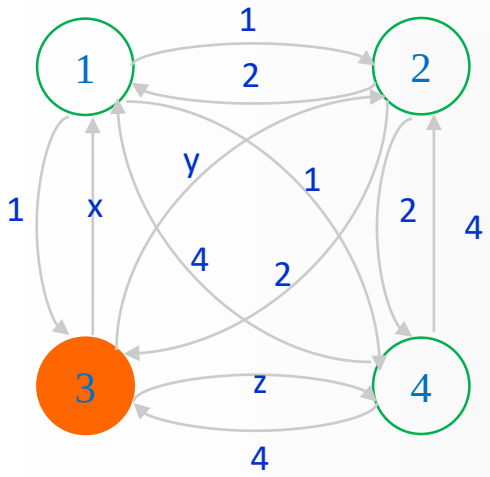
1er. Paso - El general anuncia el poder de sus tropas (en unidades de 1 kilosoldados).



Proceso defectuoso

Acuerdo en Sistemas con Falla en Procesos

2do. Paso - Los vectores que cada general arma basados en lo enviado en el primer paso



Proceso defectuoso

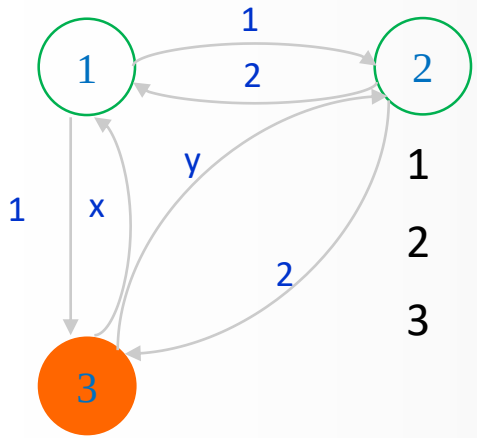
1	Got(1,2,x,4)
2	Got(1,2,y,4)
3	Got(1,2,3,4)
4	Got(1,2,z,4)

3er. Paso - Los vectores que cada uno recibe en paso 3.

1 Got	2 Got	4 Got
(1,2,y,4)	(1,2,x,4)	(1,2,x,4)
(a,b,c,d)	(e,f,g,h)	(1,2,y,4)
(1,2,z,4)	(1,2,z,4)	(i,j,k,l)

Acuerdo en Sistemas con Falla en Procesos

Lo mismo que en el caso previo, excepto que ahora con dos generales leales y un traidor.

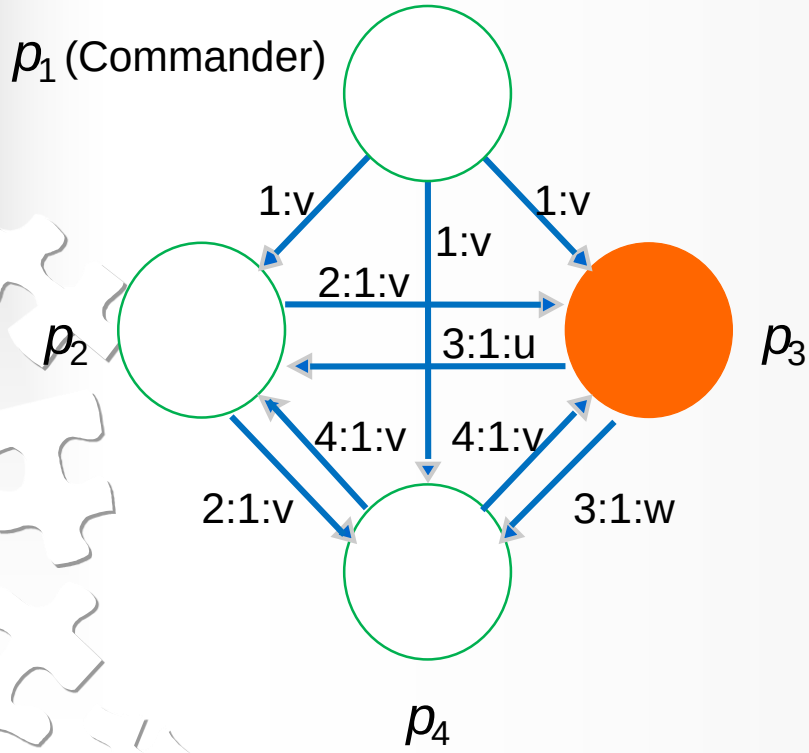


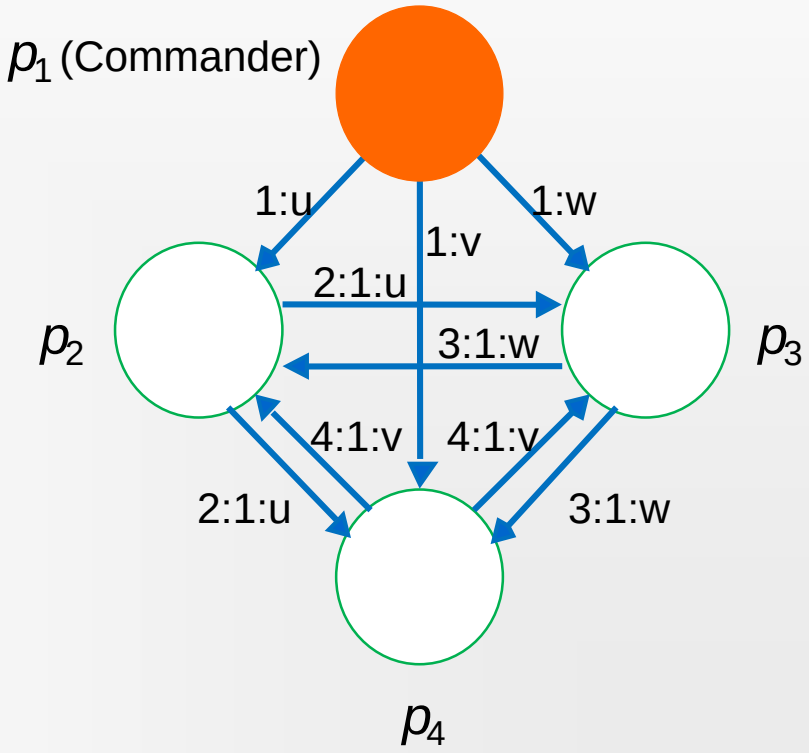
Proceso defectuoso

1	Got(1,2,x)	1 Got	2 Got
2	Got(1,2,y)	(1,2,y)	(1,2,x)
3	Got(1,2,3)	(a,b,c)	(e,f,g)

Acuerdo en Sistemas con Falla en Procesos

Problema de los **GENERALES BIZANTINOS**: hay un ***proceso especial (p_1 Commander)*** que brinda el valor a los otros para que lleguen a un acuerdo.





Agenda

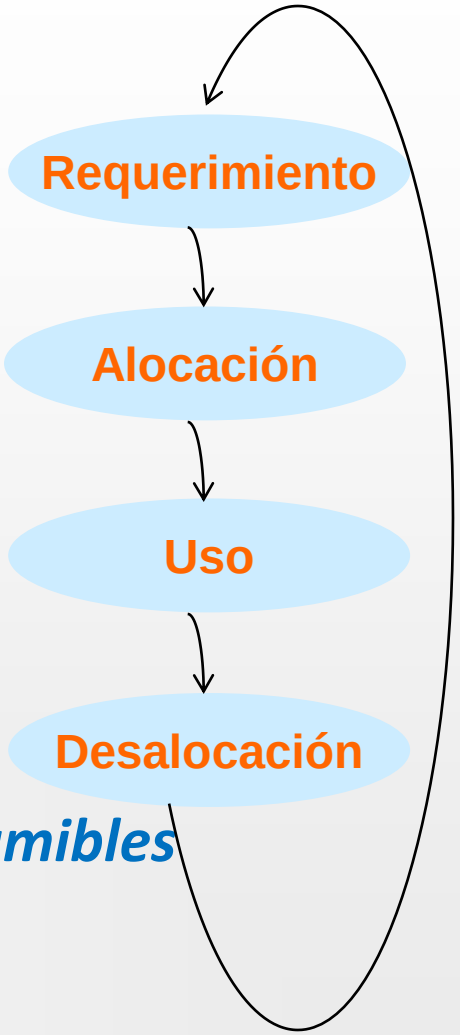
1. Sincronización de Relojes
    1. Algoritmos
    2. Relojes lógicos
2. Estado Global
3. Exclusión Mutua
4. Algoritmos de Elección
5. Alcance de Acuerdos
6. Fallas de Comunicación y Procesos
7. Interbloqueos

Interbloqueos Distribuidos

Interbloqueos (Deadlocks)

El uso de un recurso implica:

- Requerimiento del recurso
- Alocación del recurso
- Desalocación del recurso



Los recursos pueden ser ***reusables*** o ***consumibles***

Interbloqueos Distribuidos

Condiciones *Necesarias* para un Interbloqueo

- Exclusión Mutua
- Retención y espera
- No apropiación
- Espera circular

Si una de ellas no se cumple no hay interbloqueo

Interbloqueos Distribuidos

Modelamiento de interbloqueos

***Grafo dirigido*:** es un par (N,E) donde N es un conjunto no vacío de nodos y E es un conjunto de lados dirigidos. Un lado dirigido es un par (*a,b*) donde $a,b \in N$.

***Camino*:** Un camino es una secuencia de nodos (*a,b,c,…,i,j*) de un grafo dirigido tales que (*a,b*), (*b,c*), …, (*i,j*) son lados dirigidos.

***Ciclo*:** Un ciclo es un camino donde el primer y último nodo son los mismos.

Conjunto alcanzable*:** el conjunto alcanzable de un nodo ***a es el conjunto de todos los nodos ***b*** tales que que existe un camino de ***a*** a ***b***

***Nudo*:** Un nudo es un conjunto no vacío K de nodos tales que el conjunto alcanzable de cada nodo en K es exactamente el conjunto K

Interbloqueos Distribuidos

Estrategias

- Prevención
- Evasión
- Detección

La prevención se hace naturalmente.

La evasión es muy costosa.

Ambas son estrategias “pesimistas”, solo se intenta la detección y la recuperación.

Interbloqueos Distribuidos

Detección y recuperación de interbloqueos

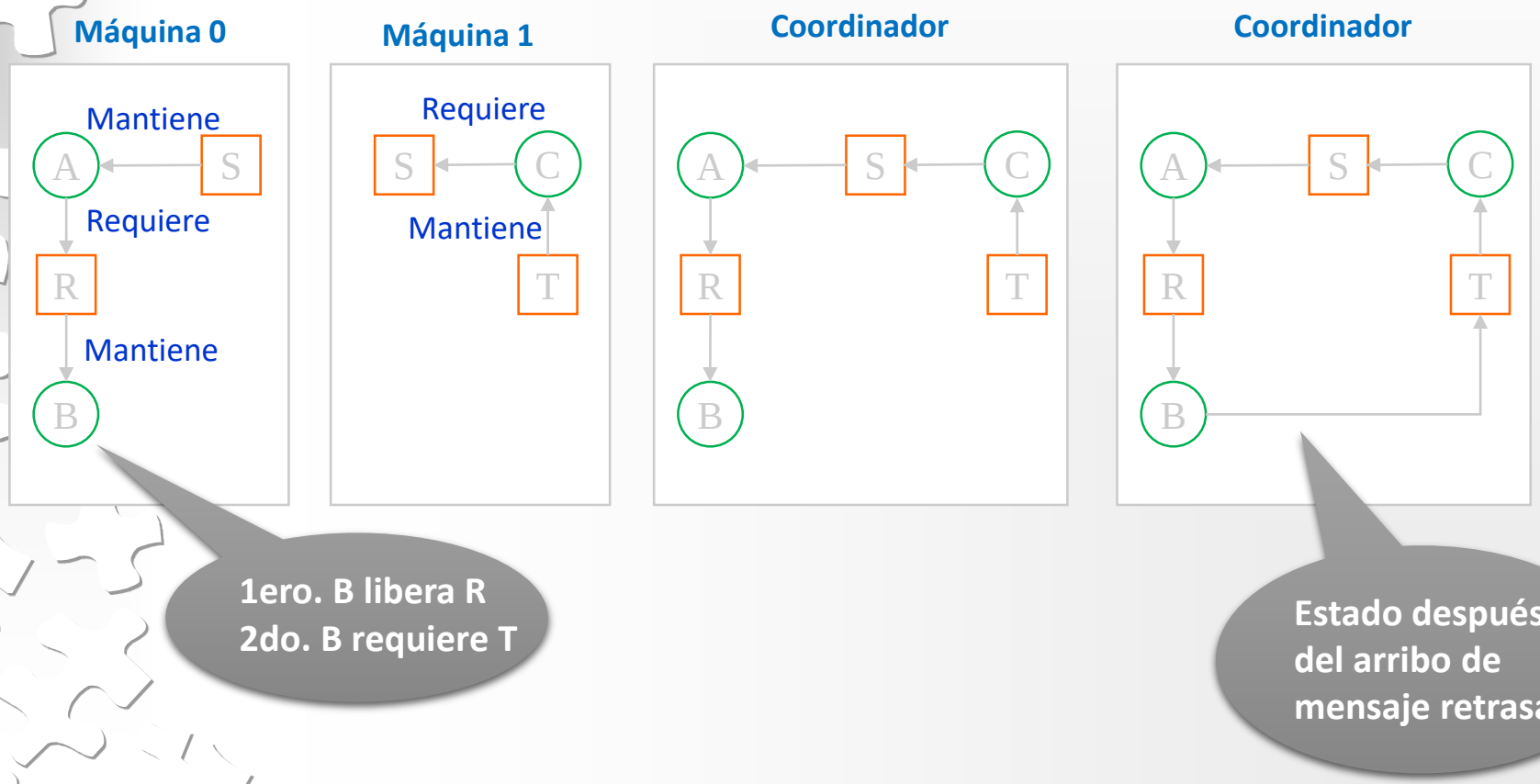
Se mantiene el grafo de espera, se busca la presencia de ciclos.

- Centralizados: Debe construirse el grafo en diferentes puntos en tiempo. Aparece o desaparece un nuevo borde, periódicamente o para buscar ciclo.
- Distribuidos: Cada sitio tiene una copia global. Pasaje de mensaje de prueba.
- Jerárquicos: Se arma un árbol de nodos con una jerarquía establecida.

Problema: se pueden producir interbloqueos “fantasmas”

Interbloqueos Distribuidos

Centralizado – Ejemplo: Bloqueo Fantasma



Interbloqueos Distribuidos – Modelos de Requerimiento

- **Interbloqueos de recursos**
- **Interbloqueos de comunicaciones**

La diferencia real entre ambos tipos es que el primero usa la condición **AND** y la segunda la condición **OR**.

Condición **AND**: Un proceso que requiere recursos para su ejecución puede proceder cuando ha adquirido todos.

Condición **OR**: un proceso que requiere recursos para su ejecución puede proceder cuando ha adquirido alguno de ellos.

Se usa OR para interbloqueo de comunicaciones porque un proceso puede estar esperando un mensaje de más de una fuente (proceso). No es determinístico.

El modelo **AND-OR** es otra generalización.

Interbloqueos Distribuidos

Condiciones de Interbloqueo

Interbloqueos de recursos única instancia (**condición AND**)

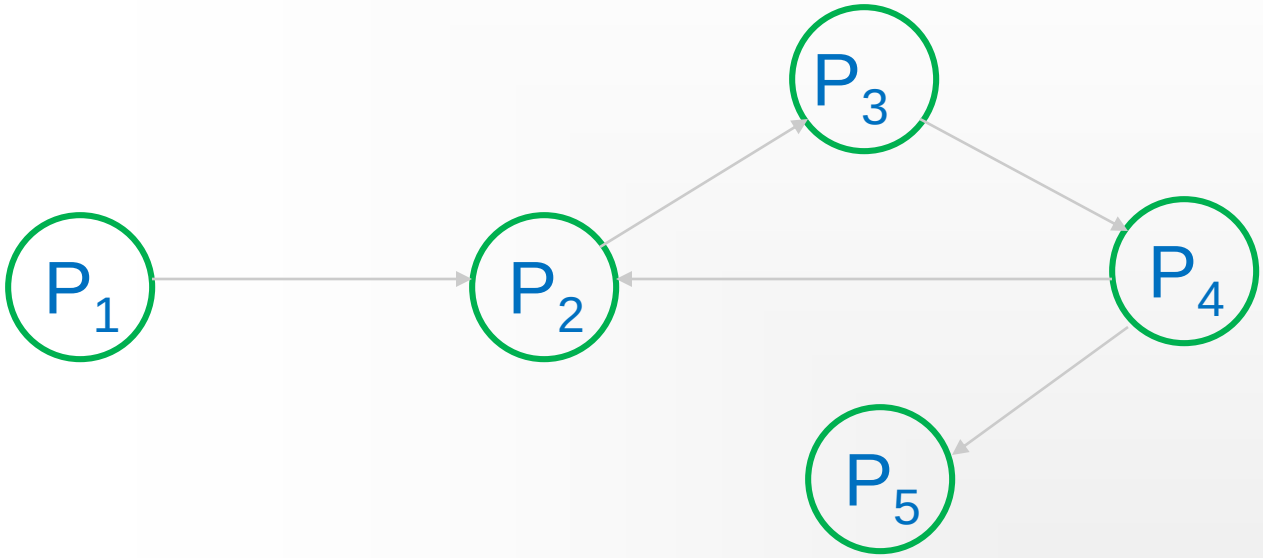
- Con detectar ciclo es suficiente

Interbloqueos de comunicaciones (**condición OR**). Con la detección de ciclos no es suficiente

- Es necesario detectar nudos.

Problema: interbloqueos fantasmas

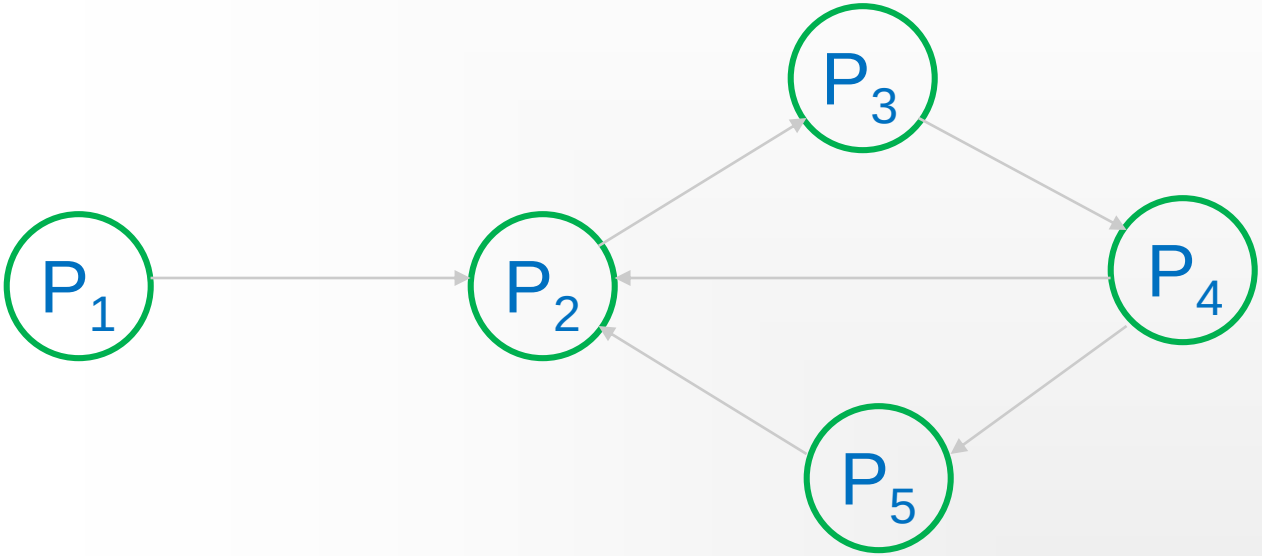
Interbloqueos Distribuidos



$P_4 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4$, conjunto $S=\{P_2,P_3,P_4\}$ es un ciclo

No hay interbloqueo (no hay nudo)

Interbloqueos Distribuidos



Hay interbloqueo (nudo: {2,3,4,5})

Bibliografía:

- Sinha, P. K.; “Distributed Operating Systems: Concepts and Design”, IEEE Press, 1997.
- Wu, Jie; “ Distributed System Design”. CRC Press, 1999.
- Tanenbaum, A.S.; van Steen, Maarten; “Distributed Systems: Principles and Paradigms”. 3rd. Edition, 2017 and 2nd Edition, Prentice Hall, 2007.
- Coulouris,G.F.; Dollimore, J. y T. Kindberg; “Distributed Systems: Concepts and Design”. 5th Edition Addison Wesley, 2011.