Tema 5. Interacción y Cooperación

- Los procesos distribuidos han de coordinar sus actividades
  - Sistemas síncronos y asíncronos
  - tolerancia de fallos
  - exclusión mutua de procesos distribuidos
  - ejemplo: reserva de billete de avión

- Para solucionar la exclusión mutua:
  - Ø variables compartidas
  - Ø facilidades dadas por un único nucleo central

- Algunos servers implementan sus cerrojos para sincronizar los accesos a los recursos que gestionan

- Otros no incluyen sincronización
  - Necesitan un servicio de exclusión mutua (daemon locks)
  - Para este curso, mecanismo de exclusión mutua distribuida
    + Dar a un solo proceso el derecho a acceder a los recursos compartidos.

- A veces se necesita elegir a un solo proceso de un conjunto para desarrollar el papel privilegiado por un largo tiempo, con un algoritmo de elección

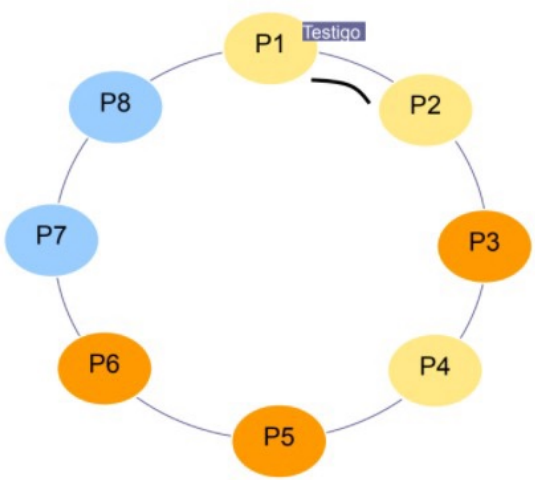
- Requisitos de exclusión mutua:
  - EM1: Seguridad en todo momento (máximo 1 proceso ejecutando la región crítica)
  - EM2: Vitalidad a todo proceso que lo solicita, se le concede la entrada/salida en la región crítica cuando lo necesita (evita deadlock* y starvation*)
    - * Abrazo mortal y inanición
  - EM3: Ordenación en la entrada a la región crítica se debe conceder según cuando sucedió la relación

- Algoritmo basado en servidor central:
  - Este concede permisos en forma de testigo, el cual concede acceso a la sección crítica (SC)
    - + Cuando sale de la SC, el testigo es devuelto
  - Suponiendo que NO HAY CAÍDAS NI PÉRDIDAS DE MENSAJES
    - + Se cumple E1 y E2
    - + E3 está asegurada en el orden de llegada de los mensajes
  - Rendimiento del algoritmo
    - + 2 mensajes para la entrada en la SC
    - + 1 mensaje para salir de la SC

- Problemas:
  - Cuello de botella: todas las solicitudes se envían al servidor
  - Caída del servidor: se selecciona un nuevo servidor → E3 No asegurada
  - Caída o fallo en el proceso en la SC

- Algoritmo basado en servidor central
  - La exclusión se logra por la obtención de un testigo
  - Anillo lógico: se crea dando a cada proceso la dirección de su vecino
    - + El testigo circula siempre por el anillo
    - + Cuando un proceso recibe el testigo
      - * Si no quiere entrar a la SC lo envía a su vecino
      - * Si quiere entrar, lo retiene
    - + Al salir de la SC: lo envía a su vecino
  - Se verifican E1 y E2 pero no se asegura E3
  - Obtención del recurso → necesita entre 1 y n-1 mensajes

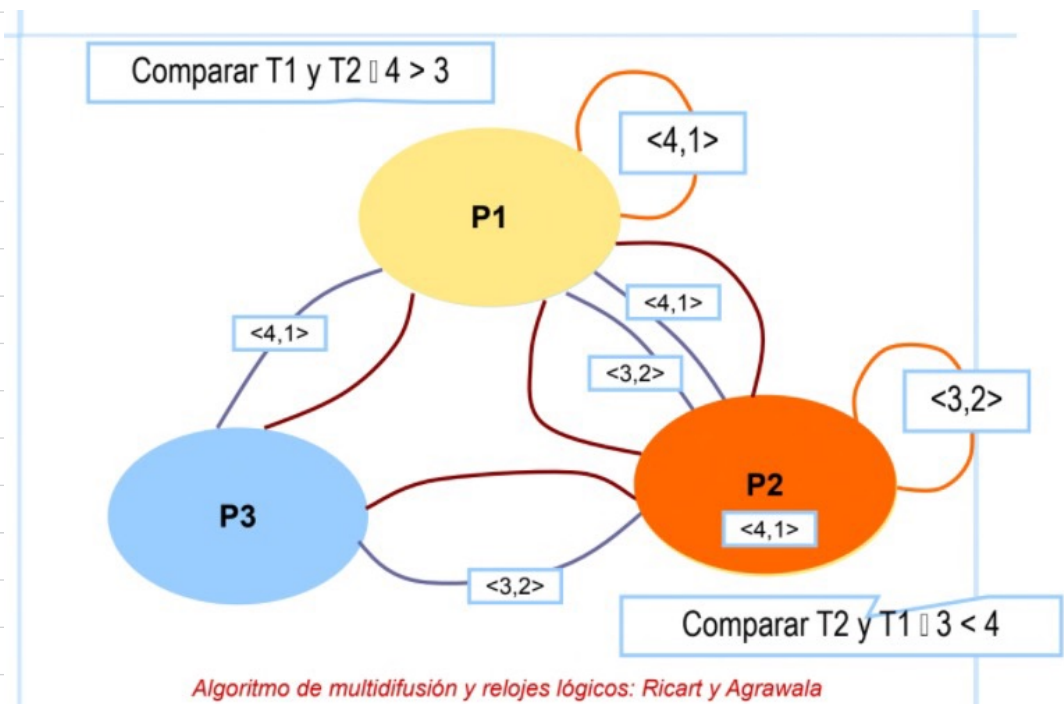
ALGORITMO BASADO EN EL ANILLO ⇒



Anillo de procesos que transfieren un testigo de exclusión mutua

- Problemas:
  - Se carga la red aunque ningún proceso quiera entrar a la SC
  - Si un proceso cae necesita reconfiguración
    - + si tenía testigo hay una elección para regenerarlo
  - Asegurarse de que el proceso ha caído → varios testigos
  - Desconexión o ruptura de red

- Algoritmos basados en relojes lógicos (- Ricart y Agrawala, - Lamport)
  - Premisas
    - cada proceso conoce la dirección del resto
    - cada proceso contiene un reloj lógico
  - Ricart y Agrawala
    - Conceptos: cuando un proceso quiere entrar en la SC, pregunta al resto si puede entrar.
    - Cuando TODOS contesten, entra.
  - El acceso es a través de un testigo.
    - Cada proceso guarda el estado en relación a la SC → liberada, buscada o tomada
  - Cola de sustituciones en cada proceso
  - Mensaje: Tupla <Ti, Pi, SCi>



Algoritmo de multidifusión y relojes lógicos: Ricart y Agrawala

- Nº de mensajes necesarios para obtener el recurso:
  - Sin soporte multicast: $2(n-1)$
  - Con soporte multicast: n
  - Este algoritmo fue refinado hasta n mensajes sin soporte multicast
- Problemas
  - Más costoso que el servidor central
  - Pero a ser distribuido, si falla algún proceso, se bloquea el sistema
  - Los procesos reciben y procesan cada solicitud
    + = o peor congestión que el servidor central
- Conclusión:
  - Ninguno puede tratar la caída de un computador o proceso
  - El algoritmo con menor número de mensajes es el de servidor central, pero supone cuello de botella
  - Es preferible que el servidor que gestiona el recurso implemente la exclusión mutua
- Procedimiento para elegir a un proceso dentro de un grupo
  - Elegir a un proceso que sustituya a uno en concreto cuando cae
- Exigencia principal: elección única aún si varios procesos lanzan el algoritmo de elección de forma concurrente
  - E1: seguridad
  - E2: vivacidad
  - Algoritmos:
    + Basado en el anillo (Chang y Roberts)
    + Del Matón (Bully → Silberschatz)
    + De invitación

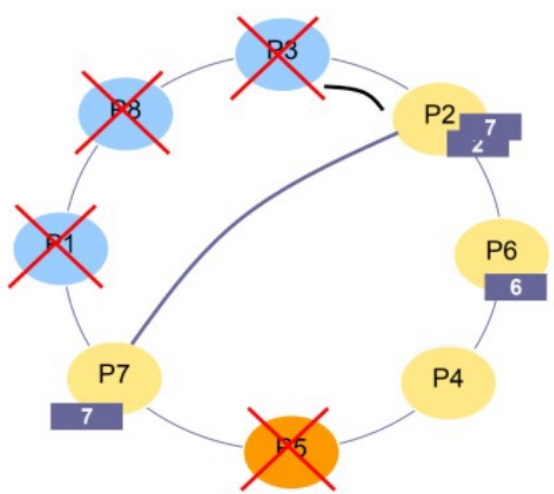
Algoritmo de elección: Anillo

- Todos los procesos son NO-candidatos:
  - se marca como candidato
  - envía un mensaje de elección con su identificador

id > su id	id = su id	id < su id
- mensaje a sus vecinos	- Se marca como no-candidato - envía "elegido" a su vecino con su identidad	- si es no-candidato → sustituye el id y envía un msg al vecino como candidato

- Recibe un mensaje de "elegido":
  se marca como NO-candidato.
  lo envía a su vecino

Algoritmo basado en anillo: ejemplo



Anillo de procesos que transfieren un testigo de exclusión mutua

- Anillo lógico: cada proceso sabe comunicarse con su vecino
  - Se coge el del id más alto
  - Se suponen procesos estables durante la elección
- Tan en baum
  - Variante donde los procesos pueden caer

- Nº de mensajes para elegir coordinador:
  - Peor caso: lanza elección solo el siguiente al futuro coordinador ($3n-1$ mensajes)
  - Mejor caso: lanza elección al futuro coordinador ($2n$ mensajes)
- No detecta fallos

Algoritmo de elección: Bully

- Requisitos:
  - todos los miembros del grupo deben conocer las identidades y direcciones de los demás miembros
  - Se supone comunicación fiable
- Se selecciona al miembro superviviente con mayor id
- Los procesos pueden caer durante la elección
- Hay 3 tipos de mensajes
  - De elección: anunciar elección
  - De respuesta: responder a un mensaje de elección
  - De coordinador: anuncia la id del nuevo coordinador

caso mejor:
- se da cuenta el segundo más alto ($n-2$ mensajes)

- caso peor
se da cuenta el más bajo ($O(n_2)$ mensajes)

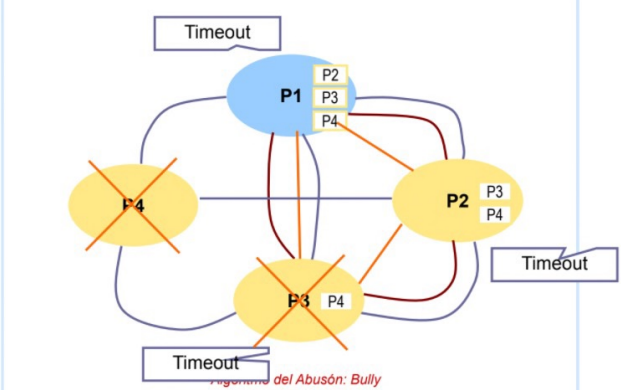
- Un proceso inicia una elección al darse cuenta de que el coordinador ha caído:
  - envía un mensaje de elección a los procesos con id > su id
  - espera algún mensaje de respuesta:
    + si vence el tiempo: el proceso se erige como coordinador y envía mensaje de coordinador a todos los procesos con id < su id
    + si recibe respuesta, espera mensaje del coordinador, pero si vence el tempo ritador, lanza una nueva elección

Si recibe un mensaje:

- De coordinador:
  + guarda el id
  + trata el proceso como nuevo coordinador
- De elección:
  + contesta con un mensaje de respuesta
  + lanza una elección

- Si un proceso se reinicia, lanza una elección a menos que sea el id más alto (él es el nuevo coordinador)

- **Algoritmo "bully": ejemplo**



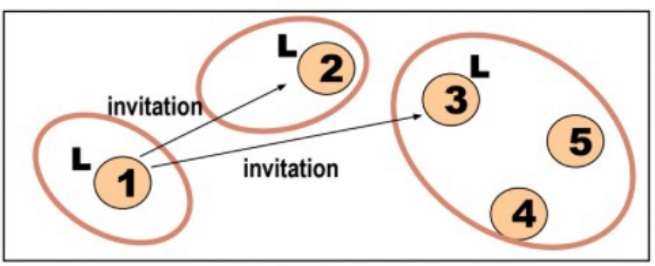
Algoritmo del Abusón: Bully

- Problemas:
  - Se basan en timeouts → puede causar la elección de multiples líderes.
  - La perdida de conexión entre 2 grupos de procesadores puede aislar a estos permanentemente

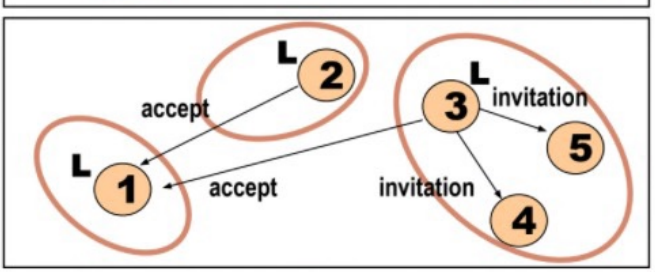
Algoritmo de invitación

- Características:
  - Definición de grupos de procesadores con 1 líder
  - Detección y agregación de grupos
  - Reconocimiento por parte del líder del grupo

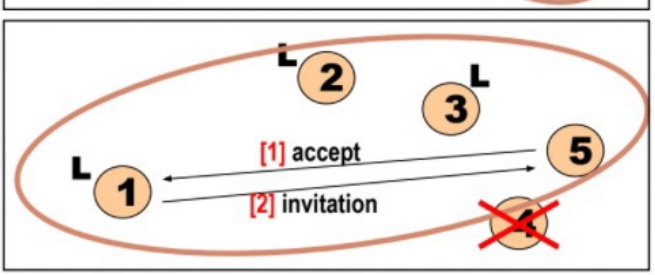
PASOS:



Si un procesador detecta la pérdida de líder se declara líder y forma su propio grupo



periodicamente el líder busca líderes de otros grupos



Dos grupos se unen por medio de mensajes de aceptación
- Explícitamente
- Respondiendo a mensajes de invitación