interacción y cooperación

interacción y cooperación

coordinación distribuida

- Los procesos distribuidos necesitan a menudo coordinar sus actividades
  - sistemas síncronos y asíncronos
  - tolerancia a fallos
  - exclusión mutua de los procesos distribuidos
  - ejemplo: reservas de billetes de avión
- En los SD para solucionar el problema de la exclusión mutua, no se pueden utilizar:
  - ni variables compartidas
  - ni facilidades dadas por un único núcleo central

→ Solución basada en el paso de mensajes

interacción y cooperación

coordinación distribuida

- Algunos servidores implementan sus propios cerrojos para sincronizar los accesos a los recursos que gestionan
- Otros servidores no incluyen sincronización (p.e. Sun NFS):
  - necesitan servicio de exclusión mutua (p.ej. *daemon lockd*)
  - para este caso se requiere un mecanismo de exclusión mutua distribuida:
    - dar a un único proceso el derecho de acceder temporalmente a los recursos compartidos
- En otros casos se necesita elegir a un único proceso de un conjunto para que desarrolle un papel privilegiado durante un largo tiempo (Redes ethernet o inalámbricas)
  - necesario un algoritmo de elección

Requisitos en la exclusión mutua

- EM1: Seguridad → en todo momento, como máximo hay un solo proceso ejecutando la región crítica
- EM2: Vitalidad → a todo proceso que lo solicita se le concede la entrada/salida en la región crítica en algún momento:
  - evita el abrazo mortal (*deadlock)* e inanición (*starvation*)
- EM3: Ordenación → la entrada en la región crítica debe concederse según la relación sucedió - antes

Algoritmo basado en servidor central

- El servidor central concede permisos en forma de testigo que concede acceso a la sección crítica (SC)
  - al salir de la SC, el proceso devuelve el testigo al servidor
- Suponiendo que no hay caídas y no se pierden mensajes:
  - se cumplen E1 y E2
  - E3 está asegurada en el orden de llegada de los mensajes al servidor
- Rendimiento del algoritmo
  - 2 mensajes para la entrada en la sección crítica
  - 1 mensaje para salir de la sección crítica

interacción y cooperación

coordinación distribuida

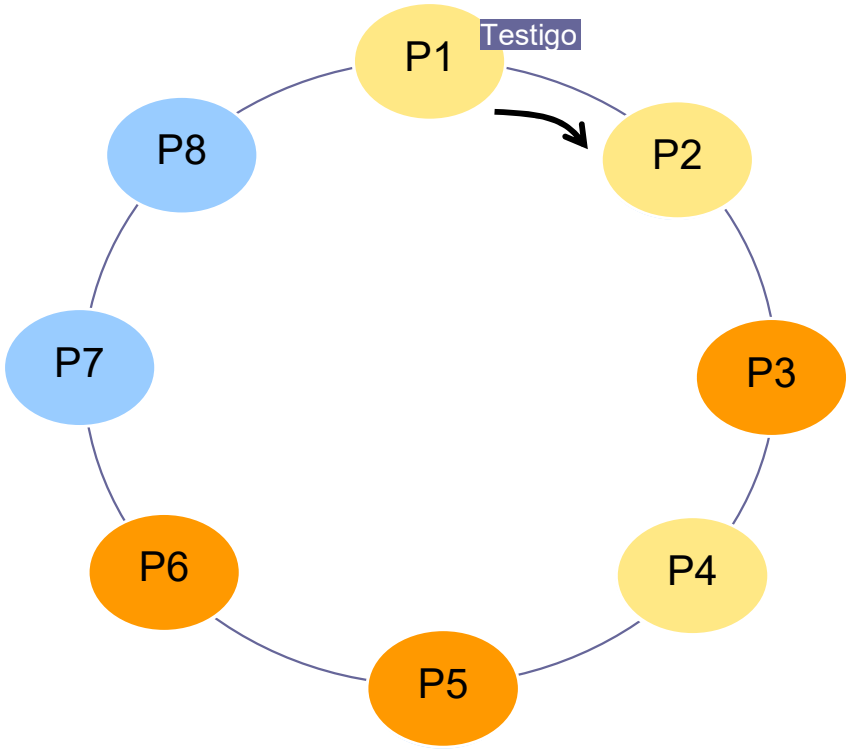
- Problemas:
  - todas las solicitudes se envían al servidor → cuello de botella
  - caída o fallo del servidor → elección de nuevo servidor → E3 no asegurada
  - caída o fallo del proceso en la SC

interacción y cooperación

coordinación distribuida

- Algoritmo basado en Anillo
  - 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 está siempre circulando por el anillo
    - Cuando un proceso recibe el testigo:
      - si no quiere entrar en la SC → lo envía a su vecino
      - si quiere entrar en la SC → 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 Anillo



Anillo de procesos que transfieren un testigo de exclusión mutua

interacción y cooperación

coordinación distribuida

- Problemas:
  - se carga la red aun cuando ningún proceso quiera entrar en la SC
  - si un proceso cae necesita reconfiguración
    - si además tenía el testigo: elección para regenerar el testigo
  - asegurarse de que el proceso ha caído →varios testigos
  - desconexión o ruptura de la red

interacción y cooperación

coordinación distribuida

- Algoritmos basados en relojes lógicos
  - Premisas:
    - cada proceso conoce la dirección de los demás
    - cada proceso posee un reloj lógico
- Algoritmos:
  - Ricart y Agrawala
  - Lamport

interacción y cooperación

coordinación distribuida

- Basados en relojes lógicos: RICART y AGRAWALA
  - Idea básica: cuando un proceso quiere entrar en la sección crítica (SC) → les pregunta a los demás si puede entrar
  - Cuando **todos** los demás le contesten → entra
- El acceso se obtiene a través de un testigo
  - cada proceso guarda el estado en relación a la SC: liberada, buscada o tomada
- Cola de solicitudes en cada proceso
- Mensaje → Tupla <Ti, Pi, SCi>

Algoritmo de RICART y AGRAWALA

- **En la inicialización**
  estado := LIBERADA;

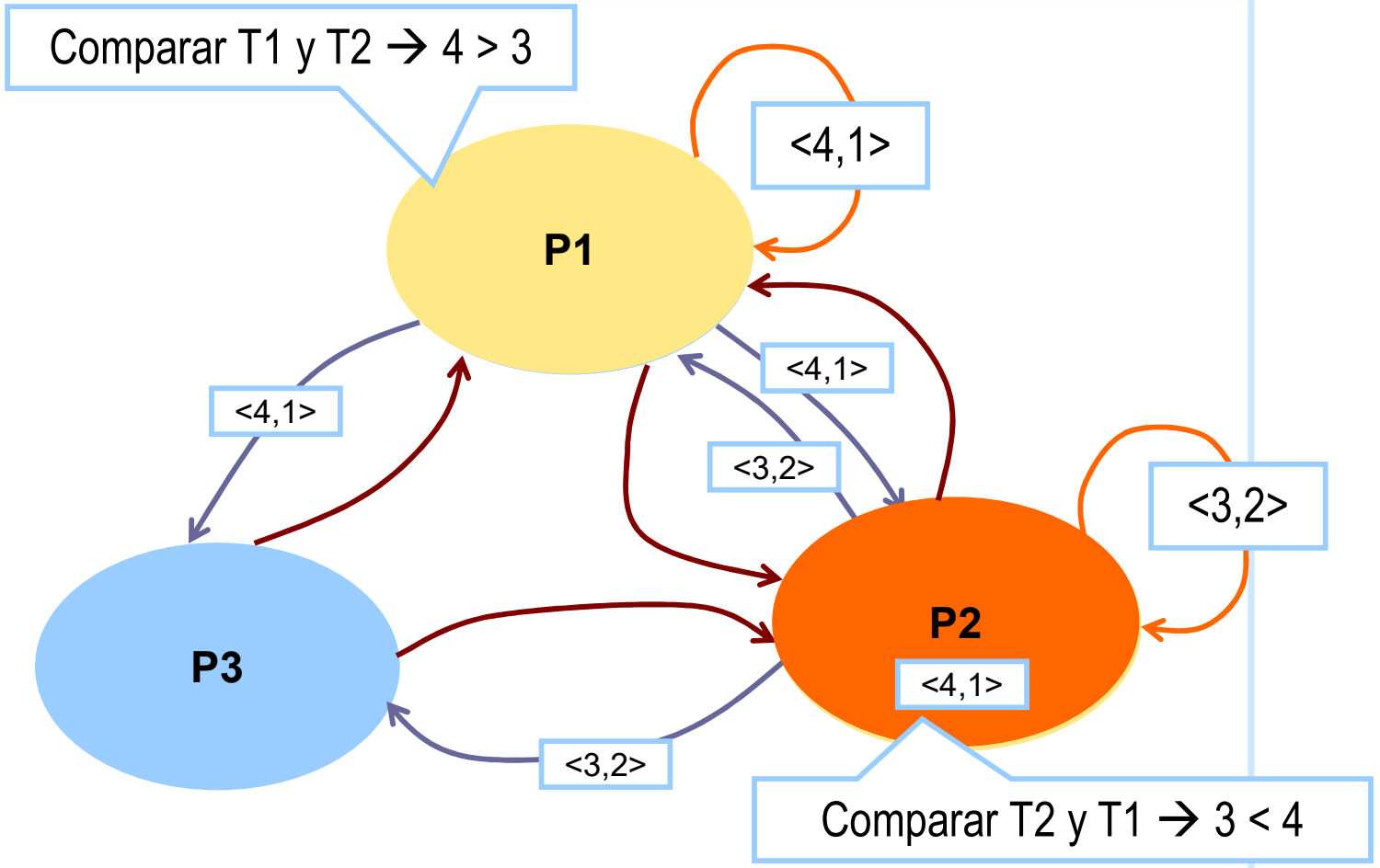
- **Para entrar en la sección crítica**
  estado := BUSCADA;
  Multitransmite *petición* a todos los procesos;
  T := marca temporal de la petición;
  Espera hasta que (número de respuestas recibidas = (*N* – 1));
  estado := TOMADA;

  ⇨ Se aplaza el procesamiento de peticiones

- **Al recibir una petición $<T_i, p_i>$ en el proceso p_j ($i \neq j$)**
  si (*estado* = TOMADA o (*estado* = BUSCADA *y* $(T, p_j) < (T_i, p_i)$))
  entonces
    pone en la cola la *petición* por parte de p_i sin responder;
  sino
    responde inmediatamente a p_i;
  fin si

- **Para salir de la sección crítica**
  estado := LIBERADA;
  responde a cualquiera de las peticiones en la cola;

Basados en relojes lógicos: RICART y AGRAWALA



Algoritmo de multidifusión y relojes lógicos: Ricart y Agrawala

interacción y cooperación

coordinación distribuida

- Número de mensajes necesarios para obtener el recurso:
  - sin soporte *multicast*: *2(n-1)*
  - con soporte *multicast*: *n*
  - el algoritmo fue refinado hasta *n* mensajes sin soporte *multicast* (Raynal, 1988)
- Problemas:
  - Algoritmo más costoso que el del servidor central
  - Pese a ser algoritmos distribuidos, el fallo de cualquier proceso bloquea el sistema
  - Los procesos implicados reciben y procesan cada solicitud:
    - igual o peor congestión que el servidor central

interacción y cooperación

coordinación distribuida

- Discusión de los algoritmos estudiados
  - Ninguno puede tratar el problema de la caída de un computador o proceso
  - El algoritmo de servidor central es el que tiene menor número de mensajes, pero supone un cuello de botella
- Conclusión:
  - es preferible que el servidor que gestiona el recurso implemente también la exclusión mutua

interacción y cooperación

algoritmos de elección

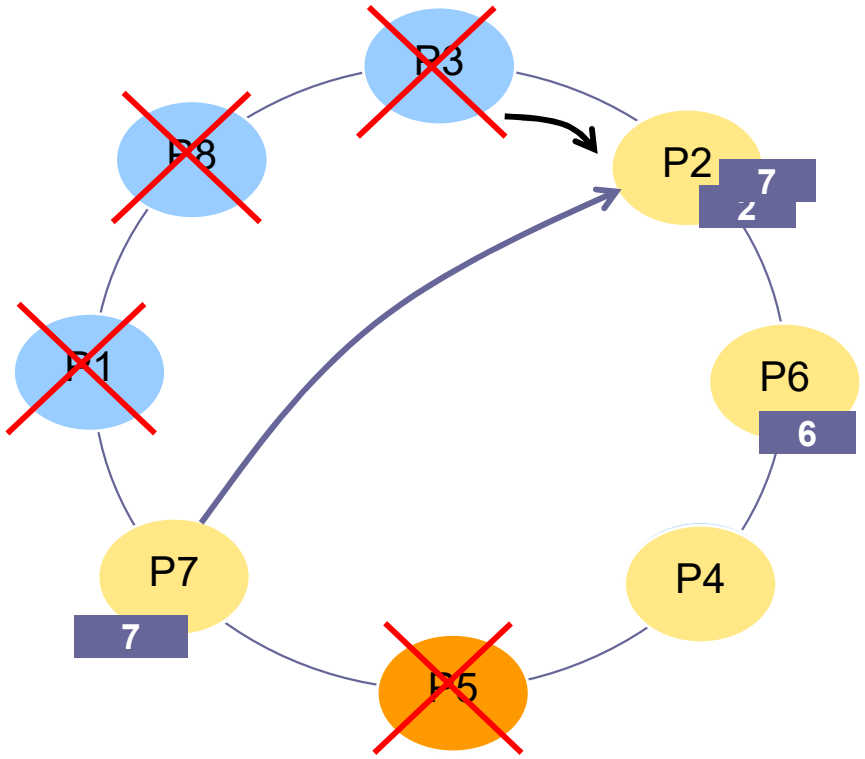
- Procedimiento para elegir a un proceso dentro de un grupo
  - ejemplo: elegir a un proceso que sustituya a uno especial (coordinador, maestro, ...) cuando éste cae
- Principal exigencia: elección única incluso si varios procesos lanzan el algoritmo de elección de forma concurrente
- E1 → Seguridad
- E2 → Vivacidad
- Dos algoritmos:
  - algoritmo basado en anillo: Chang y Roberts
  - algoritmo del matón (bully): Silberschatz
  - Algoritmo de invitación

interacción y cooperación

algoritmos de elección: anillo

- Inicialmente todos los procesos son no-candidatos: cualquiera puede empezar una elección:
  - se marca como candidato
  - envía mensaje de **elección** con su identificador
- Cuando un proceso recibe un mensaje de **elección**:
  - si identificador del mensaje es mayor que el suyo → envía mensaje a sus vecinos
  - si es menor:
    - si es no-candidato → sustituye el identificador y envía mensaje al vecino y se marca como candidato
  - si es el suyo
    - se marca como no-candidato
    - envía mensaje de **elegido** a su vecino añadiendo su identidad
- Cuando un proceso 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

interacción y cooperación

algoritmos de elección: anillo

- Anillo lógico: cada proceso sólo sabe comunicarse con su vecino
- Se elige al proceso con identificador más alto
- Se supone procesos estables durante la elección
- Tanenbaum (1992):
  - variante donde los procesos pueden caer
- Número de mensajes para elegir coordinador:
  - peor caso: lanza elección sólo el siguiente al futuro coordinador → *(3n-1)* mensajes
  - mejor caso: lanza elección el futuro coordinador → *2n* mensajes
- No detecta fallos

interacción y cooperación

algoritmos de elección: bully

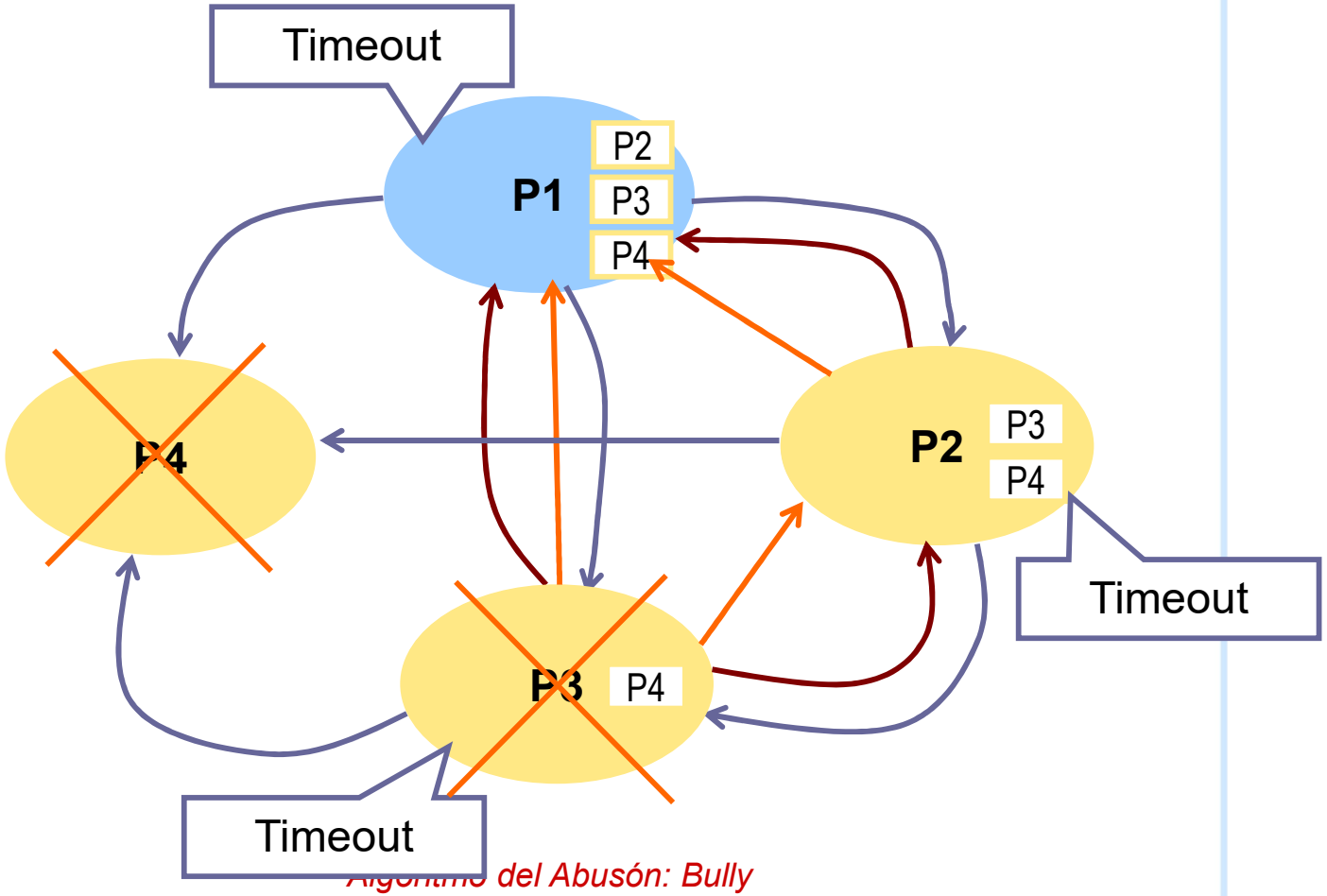
- Requisitos:
  - todos los miembros del grupo deben conocer las identidades y direcciones de los demás miembros
  - se supone comunicación fiable
- El algoritmo selecciona al miembro superviviente con mayor identificador
- Los procesos pueden caer durante la elección
- Hay 3 tipos de mensajes:
  - mensaje de elección: para anunciar una elección
  - mensaje de respuesta a un mensaje de elección
  - mensaje de coordinador: anuncia identidad de nuevo coordinador
- Número de mensajes para elegir coordinador:
  - caso mejor: se da cuenta el segundo más alto → *(n-2) m*ensajes
  - caso peor: se da cuenta el más bajo → $O(n^2)$ mensajes

interacción y cooperación

algoritmos de elección: bully

- Un proceso inicia una elección al darse cuenta de que el coordinador ha caído:
  - envía mensaje de **elección** a los procesos con identificador mayor que el suyo
  - espera algún mensaje de **respuesta**:
    - si vence temporizador →el proceso se erige como coordinador y envía mensaje de **coordinador** a todos los procesos con identificadores más bajos
  - si recibe alguna **respuesta** → espera mensaje de **coordinador**.
    - si vence temporizador, lanza una nueva elección
- Si un proceso recibe un mensaje de **coordinador**:
  - guarda el identificador y trata a ese proceso como nuevo coordinador
- Si un proceso recibe un mensaje de **elección**:
  - contesta con un mensaje de **respuesta** y lanza una elección, (si no ha lanzado ya antes una)
- Cuando un proceso se reinicia:
  - lanza una elección a menos que sea el de identificador más alto (en cuyo caso se erigiría como nuevo coordinador)

Algoritmo "bully": ejemplo



Algoritmo del Abusón: Bully

interacción y cooperación

algoritmos de elección: invitación

- Problemática de los algoritmos anteriores:
  - Se basan en *timeouts*: Retrasos de transmisión pueden causar la elección de múltiples lideres.
  - La perdida de conexión entre dos grupos de procesadores puede aislar permanentemente los procesadores.
- Algoritmo de Invitación, característica:
  - Definición de grupos de procesadores con líder único.
  - Detección y agregación de grupos.
  - Reconocimiento por parte del líder de los miembros del grupo.

interacción y cooperación

algoritmos de elección: invitación

Pasos:

- Si un procesador detecta la perdida del líder, entonces se declara líder y forma su propio grupo.
- Periódicamente el líder de cada grupo busca otros líderes de otros grupos.
- Dos grupos se unen por medio de mensajes de aceptación:
  - Como respuesta a mensajes de invitación.
  - De forma explícita.