IIC2523
Sistemas Distribuidos

Hernán F. Valdivieso López
(2025 - 2 / Clase 09)

Exclusión Mutua Distribuida

Temas de la clase

1. Algoritmos de Exclusión Mutua Distribuida
2. Servidor Central
3. *Token Ring*
4. *Multicast* de Ricart & Agrawala

Algoritmos de Exclusión Mutua Distribuida

Algoritmos de Exclusión Mutua Distribuida

- Es esencial para evitar la interferencia y asegurar la consistencia cuando múltiples procesos acceden a recursos compartidos.
- En un computador, se logra con mecanismos como *locks* y semáforos.
- En un sistema distribuido, los procesos y el recurso no se encuentran en el mismo equipo
  - No podemos usar *locks* y semáforos como hemos aprendido antes.
  - Solo podemos coordinarnos mediante mensajes.

Algoritmos de Exclusión Mutua Distribuida

- Es esencial para evitar la interferencia y asegurar la consistencia cuando múltiples procesos acceden a recursos compartidos.
- En un computador, se logra con mecanismos como *locks* y semáforos.
- En un sistema distribuido, los procesos y el recurso no se encuentran en el mismo equipo
  - No podemos usar *locks* y semáforos como hemos aprendido antes.
  - Solo podemos coordinarnos mediante mensajes.
- Los algoritmos se pueden catalogar en 2 mecanismos:
  - Basado en *token.*
  - Basado en permiso.

Algoritmos de Exclusión Mutua Distribuida

- Existen 3 requisitos que se buscan de los algoritmos de exclusión mutua:
  - ***Safety***: Como máximo un solo nodo accede al recurso crítico a la vez.
  - ***Liveness***: Las solicitudes para entrar y salir de la sección crítica finalmente tienen éxito.
    - Implica la ausencia de *deadlock* y de inanición.
    - Inanición es el aplazamiento indefinido de la entrada de un nodo que la ha solicitado.
  - ***Ordering***: Si una solicitud para ingresar al recurso crítico ocurrió antes de otra, entonces el ingreso al recurso crítico se concede en ese orden.

Servidor Central

Algoritmo 1 - Servidor Central

Nodo K - Rol Servidor

Cola de peticiones

Fin [G H] Inicio

2. Quiero el recurso crítico

1. Quiero el recurso crítico

Nodo G

Nodo V

Nodo C

Nodo H

Algoritmo 1 - Servidor Central

Nodo K - Rol Servidor

Cola de peticiones

Fin [C G H] Inicio ✅

3. Tu turno de acceder al recurso crítico

Nodo G

1. Necesito recurso crítico

2. Ya terminé de usar recurso crítico

Nodo V

Nodo C

Nodo H

Algoritmo 1 - Servidor Central

Resumen algoritmo

1. Un nodo que quiere acceso al recurso crítico envía un mensaje de solicitud al servidor.
    - Si el recurso no está siendo accedido por otro, el servidor envía un mensaje de concesión de permiso.
    - Si el recurso está ocupado, pone la solicitud en una cola.
2. El nodo espera respuesta del servidor para acceder al recurso crítico.
3. Al salir, envía un mensaje de liberación al servidor para que elimine su solicitud de la cola.
4. En caso de caída del nodo que tiene acceso, el servidor puede detectar eso y considerarlo como mensaje de liberación.

Algoritmo 1 - Servidor Central

Características

- Se elige un nodo para actuar como servidor y solo él otorga permisos para acceder al recurso crítico.
- Uso de una cola para registrar las solicitudes por orden de llegada.
- Se puede implementar mecanismos para que el servidor detecte caídas de nodos y dar permiso de recurso crítico al siguiente nodo.
- Asumiendo que no hay fallas del nodo servidor, se garantiza *Safety* y *Liveness*, pero no *ordering*... ¿por qué?

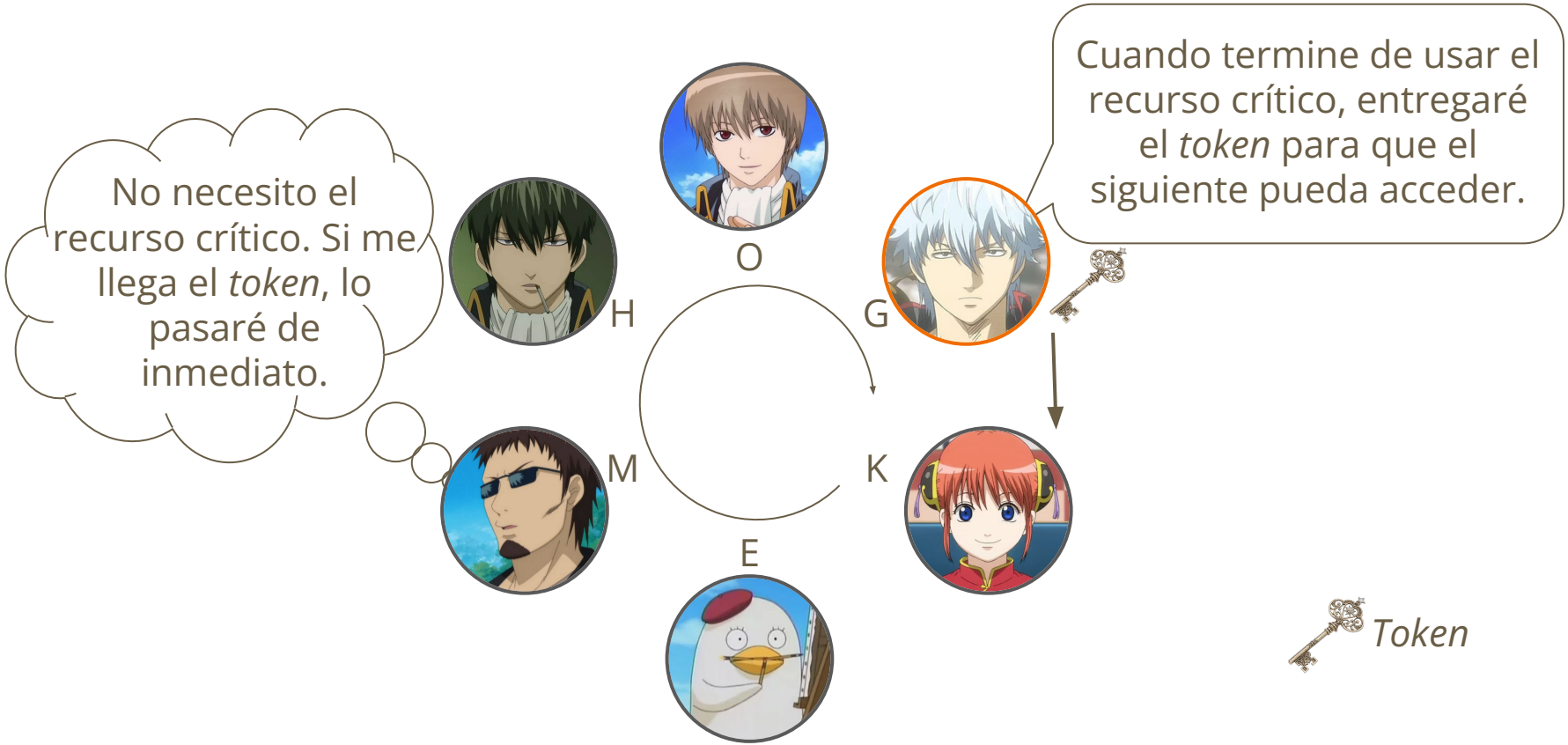
Algoritmo 1 - Servidor Central

Desventajas

- Existe un punto único de fallo. Si el servidor central falla, todo el sistema puede detenerse.
- Los nodos clientes tienen dificultad en distinguir entre un servidor caído y una espera por otro nodo.
- Cuello de botella de rendimiento para sistemas grandes o de alta concurrencia haciendo peticiones a un solo nodo.

Token Ring

Algoritmo 2 - *Token-Ring*



Algoritmo 2 - *Token-Ring*

Resumen algoritmo

1. El *token* se pasa en una única dirección alrededor del anillo.
2. Si un nodo recibe el *token* y no necesita la sección crítica, lo reenvía inmediatamente a su vecino.
3. Si un nodo necesita la sección crítica, espera a recibir el *token* y lo retiene.
4. Al salir de la sección crítica, el nodo envía el *token* a su vecino.

Algoritmo 2 - *Token-Ring*

Características

- Solo necesitas conocer al nodo vecino, no toda la red.
- Mediante la entrega de mensaje especial (*token*) se confiere la exclusión al recurso.
- No existe cuello de botella ante un solo nodo.
- Asumiendo que no hay fallas de nodos, se garantiza *Safety* y *Liveness*, pero no *ordering.* ¿por qué?

Algoritmo 2 - *Token-Ring*

Desventajas

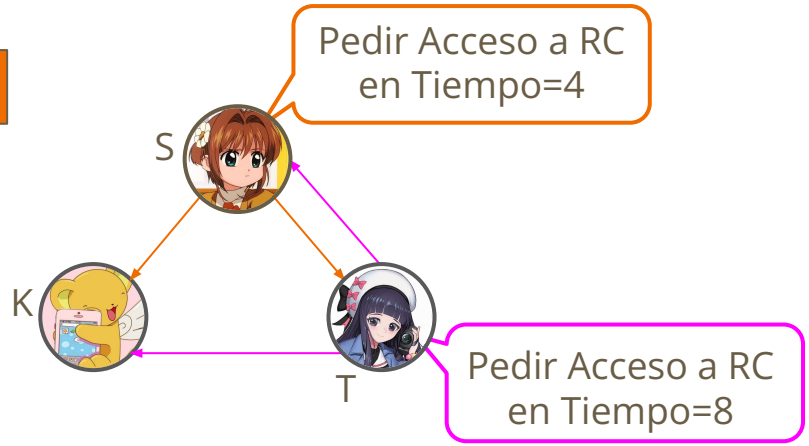
- La pérdida del *token* es un problema grave que requiere un procesamiento distribuido complejo para recrearlo y asegurar que sea único.
- Requiere que los nodos no fallen para funcionar correctamente. En otro caso se debe rehacer el anillo.
- El tiempo de respuesta puede ser largo si el *token* tiene que dar una vuelta completa al anillo (hasta N-1 mensajes).

Multicast de Ricart & Agrawala

Algoritmo 3 - *Multicast* de Ricart & Agrawala

RC = Recurso Crítico

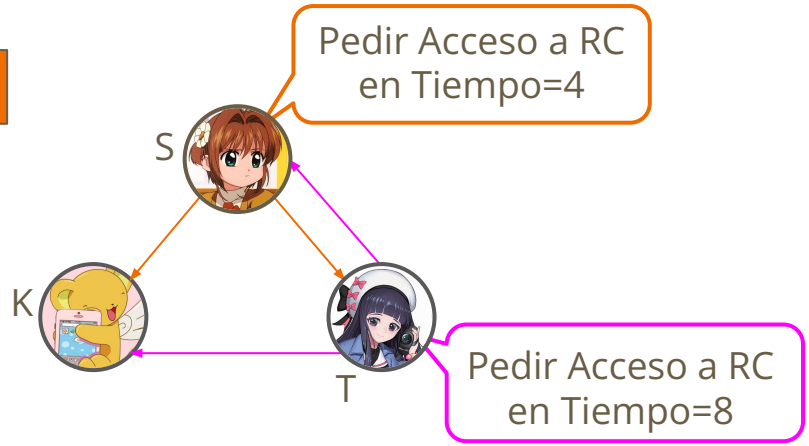
1



Algoritmo 3 - *Multicast* de Ricart & Agrawala

RC = Recurso Crítico

1

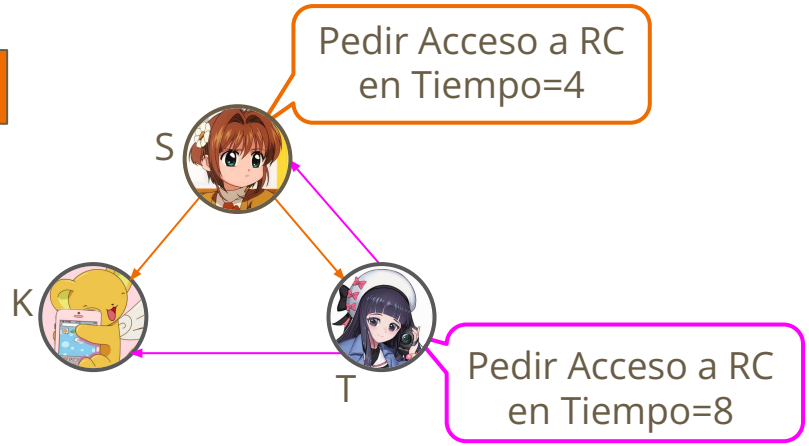


2

Algoritmo 3 - *Multicast* de Ricart & Agrawala

RC = Recurso Crítico

1



2

3

Algoritmo 3 - *Multicast* de Ricart & Agrawala

RC = Recurso Crítico

Algoritmo 3 - *Multicast* de Ricart & Agrawala

Resumen Algoritmo

1. Un nodo N_i que quiere entrar a la sección crítica, incrementa su reloj lógico y envía un mensaje $<Reloj_i, N_i>$ a todos los demás $N-1$ nodos de la red.
2. Cuando un nodo N_x recibe una solicitud $<Reloj_i, N_i>$:
    a. Si N_x no está en la sección crítica y no quiere entrar, le da permiso a N_i.
    b. Si N_x está en la sección crítica, pone en cola la solicitud de N_i.

Algoritmo 3 - *Multicast* de Ricart & Agrawala

Resumen Algoritmo

1. Un nodo N_i que quiere entrar a la sección crítica, incrementa su reloj lógico y envía un mensaje <$Reloj_i$, N_i> a todos los demás N-1 nodos de la red.
2. Cuando un nodo N_x recibe una solicitud <$Reloj_i$, N_i>:
    a. Si N_x no está en la sección crítica y no quiere entrar, le da permiso a N_i.
    b. Si N_x está en la sección crítica, pone en cola la solicitud de N_i.
    c. Si N_x quiere entrar y su propia solicitud tiene un $Reloj_x$ menor que la de N_i (solicitó antes el recurso), pone en cola la solicitud de N_i.
    d. Si N_x quiere entrar y su propia solicitud tiene un $Reloj_x$ mayor a N_i (solicitó después el recurso), le da permiso a N_i.

Algoritmo 3 - *Multicast* de Ricart & Agrawala

Resumen Algoritmo

1. Un nodo N_i que quiere entrar a la sección crítica, incrementa su reloj lógico y envía un mensaje $<Reloj_i, N_i>$ a todos los demás $N-1$ nodos de la red.
2. Cuando un nodo N_x recibe una solicitud $<Reloj_i, N_i>$:
   a. Si N_x no está en la sección crítica y no quiere entrar, le da permiso a N_i.
   b. Si N_x está en la sección crítica, pone en cola la solicitud de N_i.
   c. Si N_x quiere entrar y su propia solicitud tiene un $Reloj_x$ menor que la de N_i (solicitó antes el recurso), pone en cola la solicitud de N_i.
   d. Si N_x quiere entrar y su propia solicitud tiene un $Reloj_x$ mayor a N_i (solicitó después el recurso), le da permiso a N_i.
3. N_i entra a la sección crítica cuando ha recibido respuestas de todos los $N-1$ nodos.
4. Al salir, N_i envía respuestas a todas las solicitudes que había puesto en cola.

Algoritmo 3 - *Multicast* de Ricart & Agrawala

Características

- Los nodos que requieren la sección crítica envían un mensaje *multicast* con su solicitud, y solo pueden entrar cuando todos los demás nodos han respondido.
- Utiliza relojes lógicos para establecer un orden total de las solicitudes.
- Si un nodo se cae, ya no se le espera una respuesta de él.
  - Cuando revive, primero debe sincronizar su reloj lógico.
- Garantiza *Safety*, *liveness* y *ordering* según reloj lógico.
  - Si se ocupa reloj físico, deben estar correctamente sincronizados para asegurar *ordering*.

Algoritmo 3 - *Multicast* de Ricart & Agrawala

Desventaja

- Alta complejidad de mensajes. Se requiere $2 \times (N-1)$ mensajes por cada entrada a la sección crítica. Esto puede ser ineficiente para sistemas grandes o de alta frecuencia de uso.
- Todos los nodos están involucrados en cada decisión de acceso.

Poniendo a prueba lo que hemos aprendido 👀

Una empresa de sensores de humo ofrece la instalación de un sistema de red privada en edificios, en el cual los mensajes entre nodos siempre demoran el mismo tiempo en llegar a su destino. Además, cuentan con un sofisticado sistema de "nodos de respaldo": si un nodo de la red principal falla, su nodo de respaldo lo reemplaza inmediatamente y sin que la red principal lo note, lo que garantiza que la red principal nunca experimente realmente la caída de un nodo. Finalmente los sensores utilizan únicamente relojes de cuarzo para medir el tiempo, que se sincronizan una vez cada dos año mediante el protocolo NTP.

Se necesita garantizar que solo un sensor pueda acceder a la alarma de humo a la vez para activarla, respetando el orden de las solicitudes y asegurando que todos los sensores eventualmente tengan la oportunidad de acceder. ¿Cuál de los siguientes mecanismos **garantiza** todos estos requisitos?

a. Servidor Central.

b. Token Ring.

c. *Multicast* de Ricart & Agrawala.

d. Ninguno de los anteriores.

Poniendo a prueba lo que hemos aprendido 👀

Una empresa de sensores de humo ofrece la instalación de un sistema de red privada en edificios, en el cual los mensajes entre nodos siempre demoran el mismo tiempo en llegar a su destino. Además, cuentan con un sofisticado sistema de "nodos de respaldo": si un nodo de la red principal falla, su nodo de respaldo lo reemplaza inmediatamente y sin que la red principal lo note, lo que garantiza que la red principal nunca experimente realmente la caída de un nodo. Finalmente los sensores utilizan únicamente relojes de cuarzo para medir el tiempo, que se sincronizan una vez cada dos año mediante el protocolo NTP.

Se necesita garantizar que solo un sensor pueda acceder a la alarma de humo a la vez para activarla, respetando el orden de las solicitudes y asegurando que todos los sensores eventualmente tengan la oportunidad de acceder. ¿Cuál de los siguientes mecanismos **garantiza** todos estos requisitos?

a. **Servidor Central .**

b. *Token Ring.*

c. *Multicast* de Ricart & Agrawala.

d. Ninguno de los anteriores.

Próximos eventos

Próxima clase

- Sesión de dudas para los algoritmos de Coordinación.
  - No es materia nueva, es repasar brevemente todos los que vimos.
  - Se aprovechará de repartir los *stickers* a quienes les debo.
  - Pueden aprovechar de hacer el control 3 en la misma clase.
  - Si está colapsado/a por semana pre-receso y no tiene que retirar *stickers*, puede usar esta clase para descansar 😴 o priorizar otros cursos.

Evaluación

- El jueves se entrega el control 3.
- Sábado nos vemos a las 10:00 para la prueba.
- La tarea 2 se publica de vuelta de Receso, será simular 2 algoritmos.

IIC2523
Sistemas Distribuidos

Hernán F. Valdivieso López
(2025 - 2 / Clase 09)

Créditos (animes utilizados)

Detective Conan

Gintama

Sakura Card Captor