Tema 6: Algoritmos Distribuidos

Exclusión Mutua (EM)

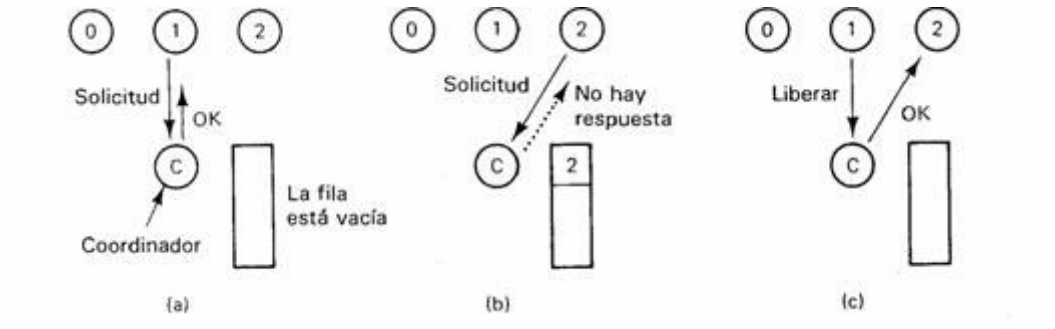
Se refiere al acceso exclusivo que tenemos a los recursos, a una sección crítica.

EXAMEN! Requisitos

- **Seguridad**: Sólo un proceso se puede ejecutar en la sección crítica y no debería haber interbloqueos, además se debe respetar la organización y relación de quién sucede antes de quién.
- **Viveza**: No hay inanición de procesos, es decir que si estoy esperando por entrar en la sección crítica seguramente lo haré en algún momento.

Algoritmo Centralizado

Funciona con un proceso coordinador



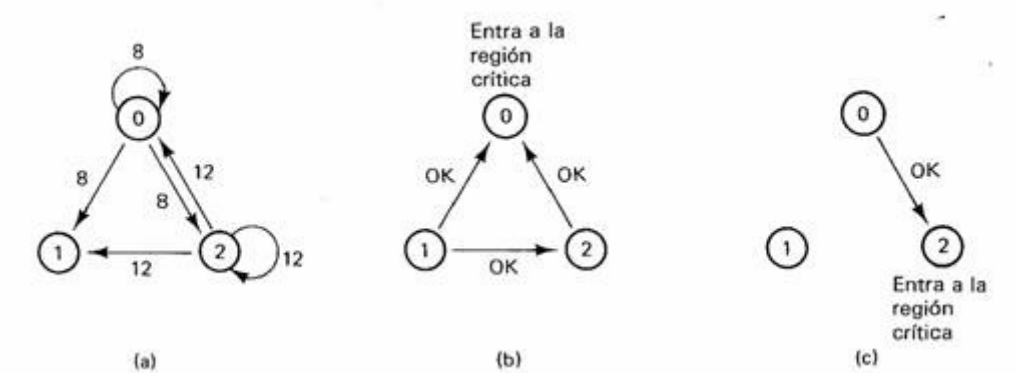
Ventajas: *Garantiza la ausencia de interbloqueos, es justo al ser un FIFO, y enviamos pocos mensajes.*

Desventajas: *Tiene poca tolerancia a fallos y puede haber un cuello de botella, esto sucedería porque tenemos varios procesos enviando 2 mensajes a un sólo proceso (coordinador).*

Algoritmo de Ricart y Agrawala

Aquí tenemos un orden de entrada en función del id del proceso (además de la marca de tiempo del reloj lógico), funciona de tal forma que cuando un proceso quiere acceder a un recurso compartido, éste envía su id al todos los procesos

incluso a si mismo, en el caso que otro proceso quiera acceder también en ese mismo instante al recurso compartido haría lo mismo, enviaría su id al resto de procesos. Los procesos que no quieren acceder simplemente devuelven un ok , pero los procesos que están "compitiendo" por el recurso tienen que comparar quién es el menor, si ellos mismos o el id que han recibido de su competidor, en el caso que sea el de su competidor no les queda más que enviar su ok y esperar en la cola, cuando el proceso con id menor haya terminado le envía un ok al siguiente proceso y ahora éste que estaba en la cola ya puede acceder al recurso.



La marca de tiempo del reloj lógico se utiliza para ver qué proceso entra primero y el id del proceso para desempatar en el caso que tengan la misma marca de tiempo.

***Ventajas**: Descentralizado, sin interbloqueos, sin inanición.*

***Desventajas**: Poca tolerancia a fallos, mucho más trafico de red.*

Algoritmo de anillo

Para esta implementación hacemos uso de un testigo (Token) donde el que lo tenga bloqueará el recurso y nadie más podrá acceder, cuando el proceso que tenga el token lo libere, éste token pasará al siguiente proceso, y así estará dando vueltas pasando por cada uno de los procesos.

***Ventajas**: Descentralizado, sin interbloqueos, sin inanición.*

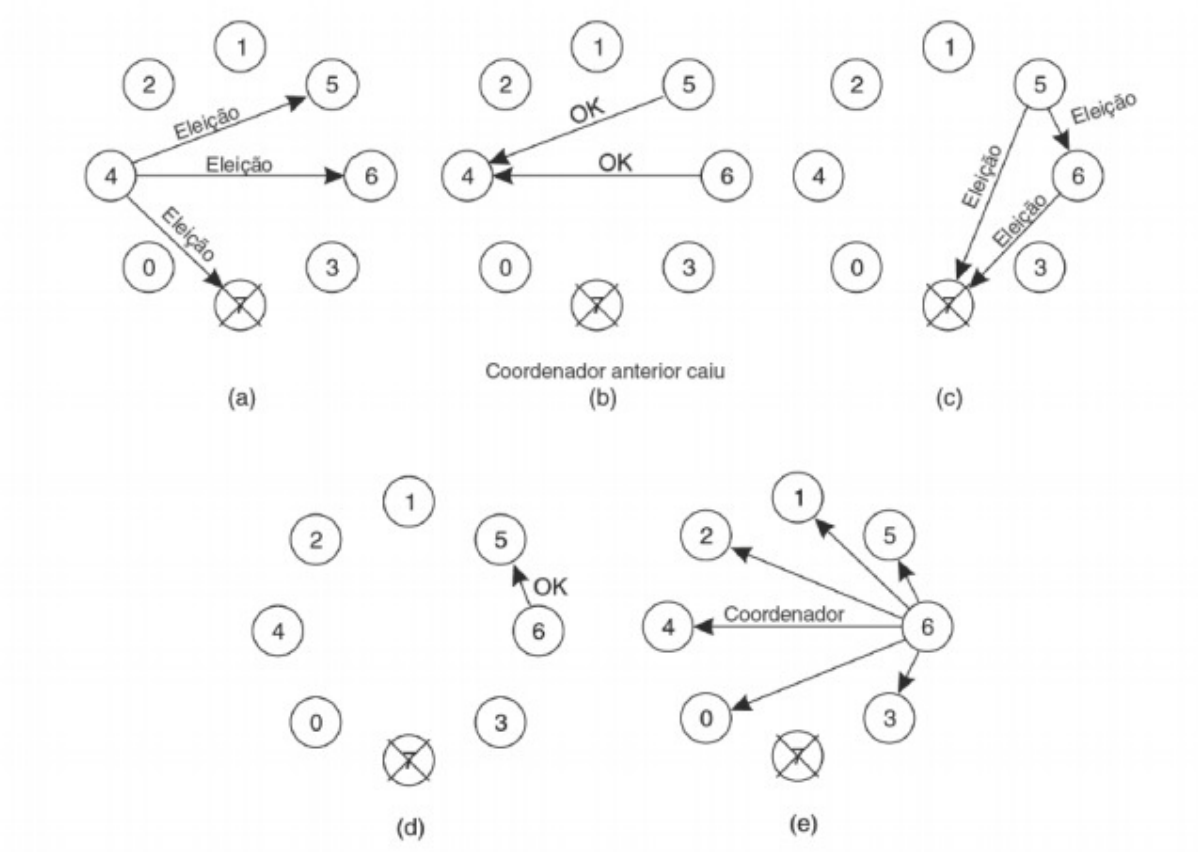
***Desventajas**: Poca tolerancia a fallos, mucho más trafico de red aunque conocido, sin respetar el orden de entrada a la sección crítica.*

Elección del coordinador (EC)

Algoritmo Bully (Distribuido)

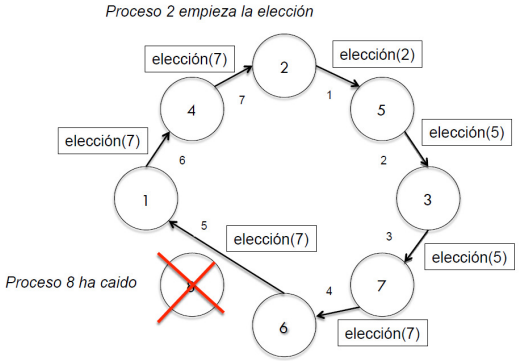
En este algoritmo nos basamos en que podemos calcular la caída de un proceso coordinador teniendo en cuenta que tenemos 2 * tiempo de envío + tiempo de ejecución supondremos que cuando esta suma de tiempos se cumple y el proceso no responde se ha caído y se inicia el proceso de elección del nuevo coordinador que se elegirá por el id más alto.

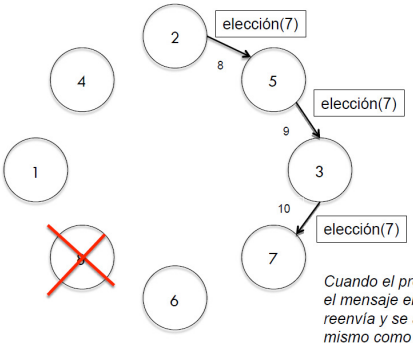
Funciona de tal forma que cuando un proceso detecta que se ha caído el coordinador, envía un mensaje de elección a todos los procesos que tengan un id mayor que el, y éstos si no están caídos le responden con un ok (este ok significa que están vivos y son mayores que él), en este caso como yo que soy el que envío tengo un id menor quedo excluido, este paso se va repitiendo entre los procesos que tengan siempre mayor id que uno mismo hasta que quede sólo uno y nadie le responda con un ok, cuando esto sucede envío un broadcast a todos los procesos señalándoles que soy el nuevo coordinador.

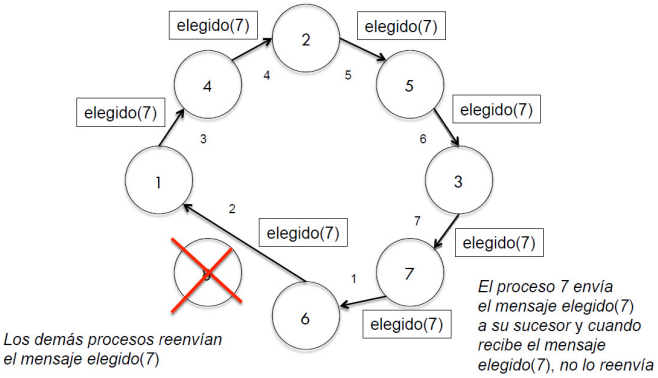


Algoritmo del anillo

Similar al algoritmo del anillo de EM con la diferencia que en vez de un token, cuando un proceso A detecta que se ha caído el coordinador, entonces este proceso enviará su id al siguiente proceso B, si este tiene un id mayor que el del proceso A entonces enviará el suyo, si no lo es reenviará el que le llegó desde A, y así sucesivamente, el proceso acaba cuando el proceso X con mayor id recibe de mensaje su propio id (lo que significa que ha ido pasándose por todos los nodos y ninguno ha podido superarlo), entonces este proceso X se convertirá en coordinador.







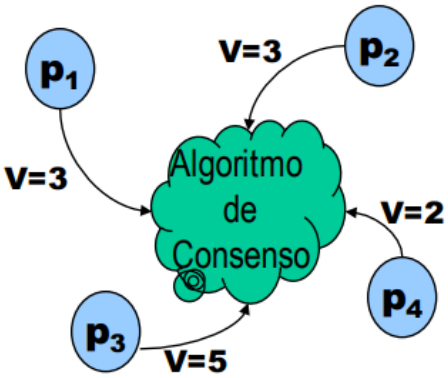
Consenso (Co)

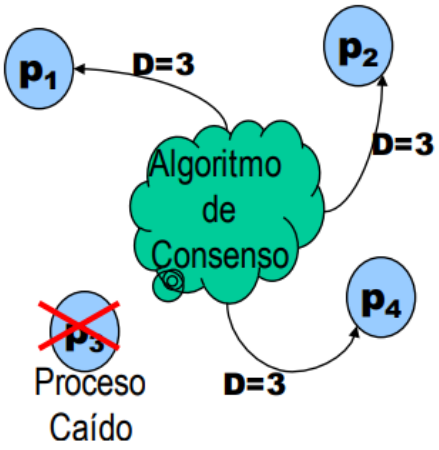
Un algoritmo de consenso es un mecanismo que permite a los usuarios o máquinas coordinarse en un entorno distribuido. Debe garantizar que todos los agentes del sistema puedan ponerse de acuerdo respecto a una fuente única de verdad, incluso en el caso de que algunos de ellos fallen.

Los procesos intercambian candidatos y cada elemento elige el mayoritario. Debe ser común.

EXAMEN! Propiedades del algoritmo de consenso

- **Acuerdo**: Todos tienen que compartir el resultado
- **Terminación**: Se debe decidir algún valor entre todos
- **Integridad**: Todos tienen que participar



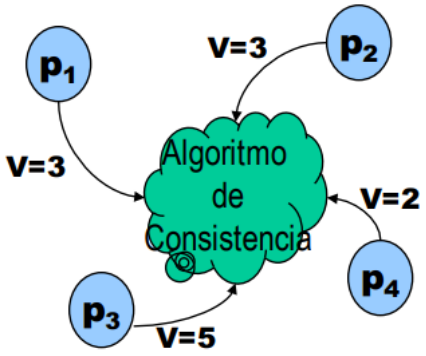


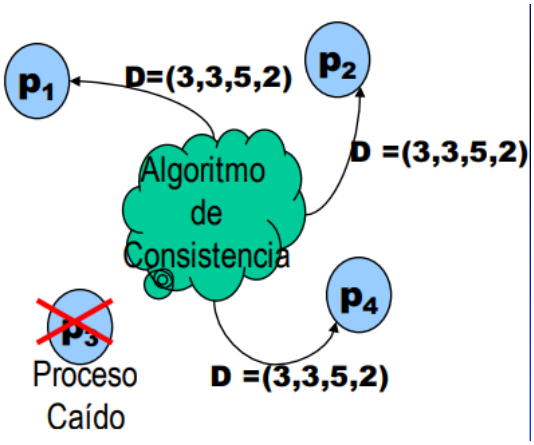
Coherencia Interactiva (CI)

Cada proceso aplica un valor diferente y se debe identificar el vector de valores usado por cada proceso.

Propiedades de la coherencia interactiva

- **Terminación**: Cada proceso correcto fija un vector valores.
- **Acuerdo**: El vector decidido es igual para todos los procesos correctos
- **Integridad**: La posición i-esima del vector se corresponde con el valor propuesto por el proceso p_i





Generales Bizantinos (GB)

Error bizantino: Un proceso genera valores de forma arbitraria.

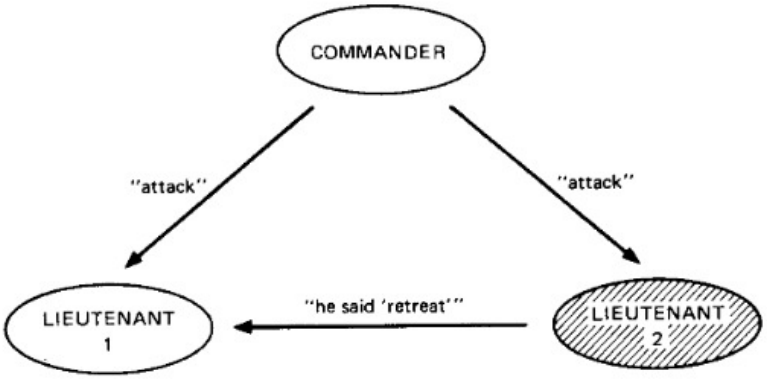


Fig. 1. Lieutenant 2 a traitor.

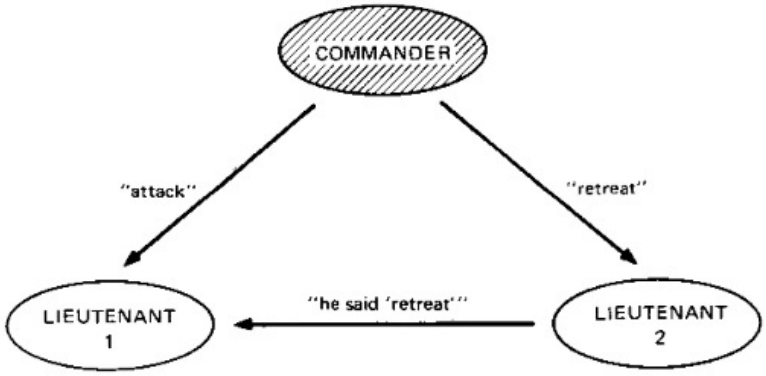


Fig. 2. The commander a traitor.

Terminación Distribuida

Se dice que ha finalizado una tarea distribuida cuando **todos** los procesos terminan y **no** hay ningún mensaje en tránsito.

El algoritmo de Dijkstra-Scholten consiste en un árbol en el cual los hijos de la raíz principal le van avisando enviando mensajes de terminación hacia sus padres.