**Algoritmo de exclusión mutua (Distribuido).**

Algoritmo de exclusión mutua(Ricart-Agrawala).

**Un algoritmo distribuido.**

(El algoritmo de Ricart y Agrawala ) Este algoritmo requiere de un orden total de todos los eventos en el sistema. Para cualquier pareja de mensajes debe quedar claro quién ocurrió primero.

Cuando un proceso desea entrar a la región crítica, construye un mensaje con el nombre de ésta, su número de proceso y la hor actual, y la envía a todos los demás procesos y de forma conceptual a él mismo. Se puede utilizar comunicación en grupo confiable.

Cuando un proceso recibe un mensaje de solicitud de otro proceso para entrar a una región crítica, debe distinguir tres casos:

1. Si el receptor no está en la región crítica y no desea entrar a ella, envía de regreso un mensaje ok al emisor.
2. Si el receptor desea entrar a la región crítica, no responde, sino que forma la solicitud en una fila.
3. Si el receptor desea entrar a la región crítica, pero no lo ha logrado todavía, compara la marca de tiempo en el mensaje recibido con la marca contenida en el mensaje que envió a cada uno. La menor de las marcas gana. Si el mensaje recibido es menor, el receptor envía un mensaje ok. Si su propio mensaje tiene una marca menor, el receptor forma la solicitud en una fila y no envía nada.

Un proceso puede entrar a la región crítica si obtiene permiso de todos. Si es así, entra y cuando sale envía mensajes ok a todos los procesos de su fila y elimina los elementos de la fila.

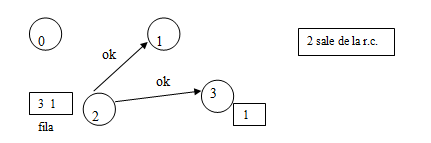
Con este método la exclusión mutua queda garantizada sin bloqueo ni inanición. El número de mensajes necesarios por entrada es de 2(n-1), donde n es el número total de procesos en el sistema.

Ejemplo: Supongamos que tenemos 4 procesos numerados del 0 al 3, los procesos 1 y 3 desean entrar a la región crítica, 0 no desea entrar a la región crítica y 2 esta dentro de la región crítica. Los procesos 1 y 3 envían mensaje casi al mismo tiempo, pero la marca de 3 es menor a la de 1 (1 tiene marca de tiempo igual a 3 y 3 tiene marca de tiempo igual a 2). Cuando 0 recibe el mensaje de 3 le otorga el permiso, como la marca de tiempo de 3 es menor que la de 1 entonces 1 debe enviar un mensaje de ok a 3, 2 esta dentro de la región crítica por lo que forma el mensaje de 3 en una fila.

Cuando 0 recibe el mensaje de 1 le responde con un ok, 3 al recibir el mensaje de 1 checa su propia marca de tiempo, como su marca de tiempo es menor que la de 1 entonces forma la solicitud de 1 es una fila y no responde nada. El proceso 2 al recibir mensaje de 1, forma el mensaje en una fila ya que 2 esta en la r.c.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | [http://1.bp.blogspot.com/-8xhgW8UA4Oc/UbZC7z9ddXI/AAAAAAAAAI4/taanZUWaTqA/s320/6.png](http://1.bp.blogspot.com/-8xhgW8UA4Oc/UbZC7z9ddXI/AAAAAAAAAI4/taanZUWaTqA/s1600/6.png) |  |  |
|  |  |  |

El proceso 3 tiene el permiso de 0 y 1 para entrar a la región crítica, y el proceso 1 tiene el permiso de 0. Cuando 2 sale de la región crítica envía mensajes de ok a los procesos de su fila, en este caso, a los procesos 3 y 1.

[](http://1.bp.blogspot.com/-Be3dbFvy6Ek/UbZDZITyNnI/AAAAAAAAAJA/S1oYxOiSyaQ/s1600/7.png)

 El proceso 3 ya tiene el permiso de todos por lo tanto puede entrar a la r.c., mientras que el proceso 1 debe esperar a que 3 salga de la región crítica. Cuando 3 salga de la región crítica debe enviar un mensaje de ok al proceso 1.

Ahora existen n puntos de falla. Sino hay respuesta, se interpretará como ua negación de permiso, aún cuando el proceso haya muerto, podría haber bloqueos.

Una posible modificación es la siguiente, cada vez que se haga una solicitud debe haber una respuesta negativa o positiva.

Funciona bien con comunicación en grupo, donde se tenga un registro de los integrantes del grupo, quiénes entran y quiénes han salido o han fallado. Funciona bien con grupos pequeños.

El algoritmo es más lento, más complejo y más caro y menos robusto que el centralizado. Pero sirve para estimular a investigadores a producir algoritmos útiles.

Fuente: <http://distribui2.blogspot.com.es/2013/06/algoritmo-de-exclusion-mutua-distribuido.html>