Glenn Ricart, National Institutes of Health. Ashok Agrawala, University of Maryland.

An Optimal Algorithm for Mutual Exclusion in Computer Networks

CARLOS MARTÍN FLORES GONZÁLEZ, Carné: 2015183528

Instituto Tecnológico de Costa Rica
Maestría en Computación
Sistemas Operativos Avanzados
Profesor: Francisco Torres Rojas, Ph.D

Se propone un algoritmo que crea exclusión mutua en una red de computadoras cuyos nodos se comunican a través de mensajes y no comparten memoria. Se asume que hay una red de comunicaciones subyacente libre de errores en donde el tránsito podría variar y los mensajes pueden no ser entregados en el orden que fueron enviados. Los nodos se asumen que operan correctamente. Este algoritmo es simétrico, muestra total control distribuido y no es sensitivo a las velocidades relativas de los nodos o a los enlaces de comunicación. El algoritmo usa solo $2 * (N - 1)$ mensajes entre nodos, donde N es el número de nodos y es óptimo en el sentido que un algoritmo simétrico y distribuido no puede usar menor mensaje si las solicitudes son procesadas por cada nodo concurrentemente. Además, el tiempo requerido para obtener la exclusión mutua es mínimo si se asume que los nodos no tienen acceso a información de tiemo y que puden actuar de forma simétrica.

Algoritmo. : *Descripción:* Un nodo entra en su sección crítica después que todos los otros nodos hayan sido notificados de la solucitud(*request*) y hayan enviado una respuesta otorgando su permiso. Un nodo que hace un intento de invocar exclusión mutua envía un mensaje de `REQUEST` a todos los otros nodos. Al recibir el mensaje de `REQUEST` el otro nodo envia ya sea un `REPLY` inmediatamente o posterga una respuesta hasta después que él salga de su propia sección crítica. El algoritmo se basa en el hecho de que un nodo que recibe un mensaje de `REQUEST` puede determinar inmediatamente si el nodo solicitante o bien él mismo debería de ser aceptado a entrar a su sección crítica primero. Al nodo que origina el mensaje de `REQUEST` nunca se le dice el resultado de la comparación. Un mensaje de `REPLY` se retorna inmediatamente si el originador del mensaje `REQUEST` tiene prioridad, de otra forma el `REPLY` se posterga. La decisión del orden de prioridad se hace al comparar un número de sequencia presente en cada mensaje `REQUEST`. Si los números de secuencia son iguales, los números de nodos son comparados para determinar cuál va a entrar primero. *Especificación:* La red consiste de N nodos. Cada nodo ejecuta un algoritmo idéntico pero se refiere a su propio número único de nodo como ME. Cada nodo tiene tres procesos para implementar la exclusión mutua: (1) Uno es despertado cuando se invoca exclusión mutua en nombre de este nodo. (2) Otro recibe y procesa mensajes `REQUEST`. (3) El último recibe y procesa mensajes `REPLY`. Los tres procesos corren asincrónicamente pero operan en un conjunto de variables compartidas. Se usa un semáforo para serializar el accesos a las variables compartidas cuando es necesario. Si un nodo puede generar múltiples solicitudes internas para exclusión mutua, él debe tener un método para serializar esas solicitudes. *Discusión:* Los números de secuencia son similares a los números usado por el *"bakery algorithm"* de Lamport. El nodo con el número más bajo es el siguiente en entra en la sección crítica. Los empates se rompen al comparar el número de nodo. Un `REPLY` se genera cuando el emisor está de acuerdo en permitir el nodo que envió el `REQUEST`

a entrar a su sección crítica primero. Los números de secuencia previenen que los nodos con enumerados con números altos sean cerrados/obstruidos[1] por nodos enumerados con números bajos. Cuando los mensajes de REQUEST del nodo A han sido procesados por todos los otros nodos, ningún otro nodo puede entrar a su sección crítica dos veces antes que el nodo A haya entrado a su región crítica. Los números de secuencia y los números de nodo forman un ordenamiento visual entre los nodos solicitantes. Ningún nodo tiene más información que una lista de algunos o todos los otros nodos que siguiéndolo en el orden virtual. Sin embargo, el sistema como un todo define un único ordenamiento virtual basado en la disciplina de *first-come-first-served*.

Afirmaciones: *Exclusión Mutua:* se logra cuando ningún par de nodos está simultáneamente en su sección crítica. Para cualquier par de nodos, uno debe dejar su sección crítica antes que el otro pueda entrar. La exclusión mutua se logra. *Deadlock:* El sistema de nodos se dice que está *deadlocked* cuando ningún nodo está en su sección crítica y ningún nodo solicitante puede proceder con su propia sección crítica. El *deadlock* es imposible. *Starvation:* ocurren cuando on nodo debe esperar indefinidamente en entrar a su sección crítica aunque otros nodos estén entrando y saliendo de sus propias secciones críticas. El *Starvation* es imposible.

Message Traffic: Este algoritmo requiere un mensaje hacia (REQUEST) y un mensaje desde (REPLY) cada otro nodo para cada entrada a una sección crítica. Si la red consiste de N nodos, $2*(N-1)$ mensajes son intercambiados. Este número es el mínimo requerido cuando los nodos actúan independientemente y concurrentemente. Por lo tanto, el algoritmo es óptimo con respecto al número de mensajes intercambiados. *Procesamiento Concurrente:* para un algoritmo simétrico y totalmente distribuido debe existir al menos un mensaje hacia adentro y un mensaje hacia afuera de cada nodo. Si ningún mensaje entra/sale de algún nodo, ese nodo no es necesario para el algoritmo, entonces el algoritmo no es simétrico o no está totalmente distribuido. Además, para permitir que el algoritmo opere concurrentemente en todos los nodos, los mensajes que entran de los nodos no deben esperar por el mensaje generado en la conclusión del procesamiendo en otros nodos. Esto podría indicar que dos mensajes separados por nodo son requeridos. El nodo solicitante no necesita enviar y recibir mensaje a sí mismo, sin embargo, un total de $2 * (N - 2)$ mensajes son necesarios. Este número debe ser un mínimo para cualquier algoritmo paralelo, simétrico y distribuido. *Procesamiento Serial:* si los nodos no actúan independientemente es posible reducir el número de mensajes usando procesamiento serial nodo-a-nodo. La primer condición discutida previamente aún se mantiene asi que un número mínimo mensajes N son requeridos. Si el algoritmo que se presenta en el artículo se modificara de tal forma que los mensajes son enviados de nodo a nodo secuencialmente, se logra también el número mínimo teórico.

Delay in Granting Critical Sections. Este algoritmo también otorga exclusión mutua con un retraso mínimo si algunas suposiciones generales son hechas. *Definition of delay:* el retraso involucrado en otorgar la sección crítica es el tramo de tiempo que inicia con el nodo solicitante pidiendo la sección crítica y que finaliza cuando el nodo entra en la sección crítica. El tiempo de ejecución de las instrucciones en el algoritmo se asumen despreciables en comparación con los tiempos de transmisión de mensajes. *Assumptions:* (1) No se dispone de información que limite los tiempos de transmisión o los tiempos reales de tránsito. A partir de esta suposición se toma un tiempo de viaje de ida y vuelta para determinar el estado de otro nodo. (2) Ningún nodo posee el recurso de la sección crítica cuando no ha sido solicitado. (3) Nodos no anticipan solicitudes. *Bounds:* (1) *minimun delay time per request:* Antes que un nodo entre a su sección crítica, tiene que estar seguro que ningún otro nodo está entrando. Para hacer eso tiene que determinar el estado actual de cualquier otro nodo que puede tener precedencia si hay una superposición de tiempo (*time overlap*) y ambos nodos se dicen que están solicitando concurrentemente. De acuerdo a las suposiciones anteriores, ninguna solicitud puede ser servida en menos de un tiempo que toma un viaje de ida y vuelta (*round-trip*). (2) *Minimun delay time with conflict:* Debido a que la regla para romper los empates no sabe cuál nodo hizo la solicitud de primero, la mitad del tiempo otorgar la sección

[1] "Shut-up"

crítica no puede ser hecho hasta que el nodo que hizo la última solicitud haya recibido todas sus respuestas de ida y vuelta (*round-trip*). (3) *System throughput:* Una vez que un nodo a liberado el recurso de su sección crítica, ningún otro nodo puede entrar a su sección crítica en menos de tiempo que el que toma un viaje de transmisión de un sólo sentido. Este es el tiempo mínimo necesario para notificar a los otros nodos que el procesamiento de la sección crítica se ha completado. (4) El algoritmo cumple con estos tres *bounds*.

Modifications: se pueden hacer algunas modificaciones para tomar ventaja de diferentes ambientes. (1) *Implicit Reply:* El mensaje de `REPLY` lleva solo un *bit* de información. Cuando el tiempo de transmisión de un mensaje entre nodos tiene un límite alto, la sensación de la respuesta puede ser cambiado tal que ninguna respuesta entre ese periódo indique una respuesta implícita. Un mensaje explícito, llamada `DEFERRED` se envía cuando un `REPLY` normalmente no se envía. El número de mensajes requeridos por el esquema implícito varía entre $1*(N-1)$ y $3*(N-1)$ dependiendo del número de mensajes `DEFERRED` enviados. Cuando hay poca contención por el recurso de la sección crítica, el número de mensajes se acerca a $1*(N-1)$. (2) *Broadcast Messages:* cuando la estructura enter nodos permite mensajes *broadcast*, el mensaje `REQUEST` inicial puede ser enviado usando ese mecanismo. El tráfico de los mensajes se redue a N mensajes, un *broadcast* REQUEST y $(N-1)$ `REPLY`s. (3) *CRing Structure:* El número de mensajes se puede cortar a N al procesar las solicitudes serialmente a través de un circuito lógico que consiste de todos los nodos en lugar de permitir que el procesamiento proceda concurrentemente. N es el número mínimo de mensajes requeridos por cualquier algoritmo distribuido simétrico cuando no hay disponible *broadcasting* y la información no es enviada via canales de tiempo (*timming channels*). El algoritmo debe ser modificado reemplazando el mensaje `REPLY` con un eco de mensaje `REQUEST`. Como el mensaje `REQUEST` viaja a través de nodos de circuitos, puede ser retrasado en varias paradas. Cuando se recibe en el nodo iniciador, la exclusión mutua se logra y el procesamiento de la sección crítica puede iniciar.

Considerations for Practical Networks: Es más conviente tener números de nodo de un rango mayor a 1...N. El algoritmo puede ser cambiado para mapear enteros 1...N en los números de nodo reales indexando una tabla `NAMES[1...N]`. La comparación de números de nodos debería entonces ser realizada comprando los valores contenidos en `NAMES`. Nuevos nodos pueden ser insertados en el grupo participante del algoritmo de exclusión mutua. Se les debe de asignar un número único de nodo y un valor apropiado para su variable `Highest_Sequence_Number`. Si un nodo falla, la falla podría ser detectada en aproximadamente el mismo tiempo en que un nodo se re-une al grupo. Si un nodo quiere dejar el grupo puede hacerlo notificando a todos los otros nodos su intención. Los otros nodos deberían de dar acuse de recibo de este mensaje. Mientras espera por el acuse de recibo, el nodo saliente no puede solicitar exclusión mutua y debe continuar enviar mensajes de `REPLY` a cada mensaje `REQUEST` que recibe. En práctica algunos nodos fallarán y no van a responder a los mensajes enviados directamente a él. Para prevenir esta situación, un mecanismos de recuperación se puede agregar. La detección del tiempo de espera (*timeout*) del nodo que falla depende del conocimiento de un límite superior en el tiempo el cual pueda transcurrir antes que un nodo que en funcionamiento responda a un mensaje y a una estimación del tiempo máximo de procesamiento dentro de la sección crítica. El único mensaje en el algoritmo original que demanda por una respuesta es el mensaje de `REQUEST`.

1. ¿CUÁL ES EL PROBLEMA QUE PLANTEA EL *PAPER*?

Cómo brindar una solución para asegurar exclusión mutua en ambientes distribuidos.

2. ¿POR QUÉ EL PROBLEMA ES INTERESANTE O IMPORTANTE?

Aunque en el artículo no se habla mucho del problema como tal (se aborda la solución de forma directa), el asegurar exclusión mutua adecuadamente en ambientes distribuidos es vital para contar con sistemas que sean

confiables y consistentes. Cuando se coopera o compite por un recurso compartido se necesitan de mecanismos para orquestar y sincronizar el uso sin que se causen efectos secundarios dentor del sistema.

3. ¿QUÉ OTRAS SOLUCIONES SE HAN INTENTADO PARA RESOLVER ESTE PROBLEMA?

Aunque muchos han considerado implementaciones para exclusión mutua, el único algorimo para exclusión mutua en una red de computadoras fue propuesto por Lamport. Requiere que aproximadamente $3 * (N - 1)$ mensajes sean intercambiados por invocación de sección crítica.

4. ¿CUÁL ES LA SOLUCIÓN PROPUESTA POR LOS AUTORES?

Se propone un algoritmo que crea exclusión mutua en una red de computadoras cuyos nodos se comunican a través de mensajes y no comparten memoria. Se asume que hay una red de comunicaciones subyacente libre de errores en donde el tránsito podría variar y los mensajes pueden no ser entregados en el orden que fueron enviados. Los nodos se asumen que operan correctamente. Este algoritmo es simétrico, muestra total control distribuido y no es sensitivo a las velocidades relativas de los nodos o a los enlaces de comunicación. El algoritmo usa solo $2 * (N - 1)$ mensajes entre nodos, donde N es el número de nodos y es óptimo en el sentido que un algoritmo simétrico y distribuido no puede usar menor mensaje si las solicitudes son procesadas por cada nodo concurrentemente. Además, el tiempo requerido para obtener la exclusión mutua es mínimo si se asume que los nodos no tienen acceso a información de tiemo y que puden actuar de forma simétrica.

5. ¿QUÉ TAN EXITOSA ES ESTA SOLUCIÓN?

El algoritmo presentado en este artículo no depende que los mensajes sean entregados en el orden en que fueron enviados. Si tal condición existe, existe un límite en el número de veces en que los otros nodos pueden entrar a su sección crítica antes de que un nodo solicitante A pueda. Si la entrega no está garantizada en el orden de transmisión, el análisis del peor caso muestra que $N(N + 1),2 - 1$ nodos puede entrar en su sección crítica antes que el nodo A pueda.

Si la entrega está garantizada en el en order de transmisión, ningún nodo puede entrar a su sección crítica más de dos veces entre el tiempo en que A selecciona un número de secuencia y a A se le permite entrar en su sección crítica. No más de $2 * (N - 1)$ secciones críticas son posibles antes que A pueda entrar.

Cuando entrega en orden es utilizada, un nuevo nodo puede asumir que su `Highest_Sequence_Number` está sincronizado cuando ha escuchado el cuarto mensaje de `REQUEST` desde el mismo nodo.