Programación Matemática y Software (2019) 11 (1): 15-25 ISSN: 2007-3283

Avances en Algoritmos de Exclusión Mutua en Sistemas Distribuidos

Advances in Mutual Exclusion Algorithms in Distributed Systems

Daniel Sánchez Ruiz[1], Hilda Castillo Zacatelco[1], Claudia Zepeda Cortés[1], Rafael de la Rosa Flores[1],Ana Patricia Cervantes Márquez[1], Misael Limón Martínez[1], José Luis Carballido Carranza[1]

[1]Facultad de Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla, Av San Claudio y 14 Sur, Cd Universitaria, Puebla, Puebla, México, C.P. 72592.

* Correo-e: daniel.sanchez.4712@gmail.com

PALABRAS CLAVE:

Exclusión mutua, Sistema Distribuido, Tolerancia a Fallas.

RESUMEN

En este artículo se presenta un análisis de algunos trabajos recientes sobre algoritmos de exclusión mutua en sistemas distribuidos. Estos algoritmos pueden clasificarse en algoritmos basados en permisos y algoritmos basados en tokens. En este trabajo se analizan únicamente los que están basados en tokens por la diversidad que existe de ellos. Para cada algoritmo analizado se exponen ventajas y desventajas las cuales dan pie a nuevas investigaciones. Además, se presentan aplicaciones en donde estos algoritmos son utilizados y se realiza una comparación entre las propuestas.

KEYWORDS:

Mutual exclusion, Distributed System, Fault Tolerance.

ABSTRACT

This paper presents an analysis of some recent works on mutual exclusion algorithms in distributed systems. These algorithms can be classified into permission-based algorithms and token-based algorithms. In this paper we analyze only those that are based on tokens because of the diversity that exists among them. For each analyzed algorithm, advantages and disadvantages are exposed which give rise to new investigations. In addition, applications where these algorithms are used are presented, and a comparison between the proposals is made.

Recibido: 3 de agosto 2018• **Aceptado:** 5 de diciembre de 2018 • **Publicado en línea:** 28 de febrero de 2019

1 INTRODUCCIÓN

En la actualidad los sistemas computacionales son usados en múltiples áreas como la investigación científica, bases de datos, sistemas de autenticación, sistemas de realidad virtual o realidad aumentada. De manera particular, para el manejo de grandes cantidades de datos como predicciones del clima, juegos en línea, cómputo en la nube, redes par a par y en el manejo de bases de datos distribuidas, se utilizan los sistemas distribuidos. Un sistema distribuido es un conjunto de computadoras que aparentan ser una única computadora ante cualquier usuario [1]. La complejidad con estos sistemas empieza desde el momento mismo que tiene que diseñarse, ya que existen múltiples características que un sistema distribuido debe de cumplir, como lo son la confiabilidad, disponibilidad, apertura e integridad de información entre otras. Dentro de la integridad de información, un problema relevante son las condiciones de competencia que se generan al tener recursos compartidos. Por esta razón, el tema de la exclusión mutua se vuelve de sumo interés. Este problema debe de ser resuelto para asegurar el acceso de forma correcta a los recursos compartidos, con la finalidad de mantener la integridad y consistencia en los datos.

Se han elaborado trabajos que analizan algoritmos de exclusión mutua en sistemas distribuidos. El trabajo de Velazquez [2] hace una revisión extensa, sin embargo, los algoritmos que analiza son propuestas que ya se dieron hace varios años. Saxena et al. en [3] hace otra revisión pero solo se enfoca en algoritmos basados en permisos. En este trabajo se realiza una revisión más actual con algoritmos basados en tokens y se lista alguna aplicación de estos algoritmos. A forma de discusión se brinda una comparativa de las propuestas donde se discuten sus principales características y aplicaciones.

En los sistemas distribuidos, los procesos utilizan recursos compartidos. Cada proceso dentro de un sistema tiene un segmento de código llamado región crítica, en esta región es donde se accede al recurso compartido. Cada proceso debe de pedir permiso para poder entrar en la región crítica y debe de liberarla después de haberla ocupado en su ejecución, para permitir, a otro proceso, entrar a la región crítica. Un algoritmo de exclusión mutua se define como el mecanismo para poder asegurar que solo un proceso esté en la región crítica, debe de asegurar que no existan *deadlocks* (interbloqueos entre procesos), inanición de procesos y que se brinde cierta justicia en el orden en el que las peticiones son procesadas [4,5].

Hay dos grandes enfoques que pueden ser ocupados para implementar un mecanismo de exclusión mutua en un sistema distribuido. Un enfoque es el centralizado, en este uno de los nodos funciona como coordinador central. Los procesos preguntan únicamente al coordinador si pueden entrar en la región crítica o no. El coordinador es totalmente responsable de tener toda la información del sistema y de dar permiso para hacer uso de los recursos compartidos. Este enfoque tiene un punto de falla crítico y es que tiene un punto centralizado, el coordinador, por lo que si este falla lo más probable es que el sistema no funcione de forma correcta.

El otro enfoque es el distribuido, la toma de decisión es distribuida alrededor del sistema entero y la solución al problema de exclusión mutua es mucho más complicada, por la dificultad que representa tener un conocimiento total del sistema, esto debido a la falta de una memoria compartida, un reloj físico común y por lo impredecible en los retrasos de los mensajes y la tolerancia a los fallos. En este enfoque hay dos grupos, los basados en permisos y los basados en tokens [6,7].

A pesar de la complejidad que pueda representar brindar una solución distribuida, esta solución es más completa y robusta que una centralizada, por esta razón en el presente trabajo no se estudia el enfoque centralizado.

El documento está organizado de la siguiente forma. En la sección 2 se presentan los algoritmos de exclusión mutua basados en permisos y en tokens que han tenido un impacto considerable en propuestas posteriores, y también se presentan algunas de las propuestas más recientes de algoritmos de exclusión mutua basados en tokens. En la sección 3 se hace una discusión con los resultados de cada propuesta. Finalmente en la sección 4 se brindan conclusiones sobre el trabajo.

2 ALGORITMOS DE EXCLUSIÓN MUTUA

Dentro del diseño de los algoritmos distribuidos hay que considerar varios aspectos. Por ejemplo, algunos de estos serían determinar si todo nodo tiene un identificador único, si los procesos compiten por un

recurso compartido o más, o si todos los nodos están conectados.

Otro elemento a tener en cuenta son los aspectos pertinentes a la red. Algunas de las características a considerar son, corroborar si la entrega de los mensajes se puede garantizar, si el orden en el que se envían los mensajes se conserva, el retraso que puede existir en el envío de mensajes y finalmente si la topología es conocida.

Hay dos grandes grupos en los que los algoritmos distribuidos de exclusión mutua pueden clasificarse por el principio básico de su diseño. Estos grupos son los algoritmos basados en los permisos y los algoritmos basados en tokens [6,7].

2.1 ALGORITMOS BASADOS EN TOKENS

De acuerdo a Tanenbaum [1] estos algoritmos logran la exclusión mutua pasando un mensaje especial entre los procesos llamado token. Solo hay un token disponible y solo aquel proceso que lo tenga podrá entrar en la región crítica.

Hay mecanismos para mover el token alrededor de todos los procesos. Si los procesos están organizados de forma lógica en una estructura de anillo, el token puede viajar alrededor de todo el anillo, de proceso en proceso para darles el derecho a entrar a la región crítica. Si un proceso recibe el token y está interesado en entrar a su región critica, procede a su ejecución. Después de que el proceso salga de la región crítica el token es liberado para circular otra vez. Por otro lado si el proceso no está interesado en su región critica solo lo pasa al siguiente nodo en el anillo. Este mecanismo tiene un alto costo en cargas de trabajo ligeras pues el mensaje de token circulará aun cuando ningún proceso quiera entrar en su región crítica [6,7].

Otro método para mover el token en el sistema es preguntando cuando un proceso quiera entrar a su región critica. Un proceso solicitante manda un mensaje de petición al dueño actual del token y espera a que este le llegue. Una vez que el poseedor del token termine de ejecutar su región crítica escoge a un proceso a quien mandárselo. Si ningún proceso requiere entrar en la región critica, el poseedor del token no tiene la obligación de mandarlo. La principal preocupación de este mecanismo es como localizar al poseedor del token de forma que se minimicen los mensajes generados por un proceso que lo solicite [6,7].

2.2 ALGORITMOS BASADOS EN PERMISOS

En el grupo basado en permisos el derecho a entrar a la región crítica es formalizado al recibir permiso de un conjunto de nodos en el sistema. Un proceso que desea entrar en su región critica pregunta a los otros si le dan permiso para poder acceder y espera hasta que este permiso llegue. Un proceso solo entra en su región crítica después de haber recibido el permiso de todos los nodos del conjunto [6,7].

Uno de los problemas en estos algoritmos es encontrar un número mínimo de nodos que tengan que dar permiso, para que un proceso solicitante pueda entrar en su región crítica. Esto se puede traducir en definir cuántos permisos debe recolectar un proceso para poder ejecutar su región crítica.

2.3 ALGORITMOS DE EXCLUSIÓN MUTUA A TRAVÉS DEL TIEMPO

A continuación se discuten algunos de los primeros algoritmos de exclusión mutua en sistemas distribuidos. Dentro de los algoritmos que se han propuesto ocupando el modelo distribuido, hay trabajos que por su importancia y aporte han servido como referencia tanto en el grupo basado en permisos como en el grupo basados en tokens. Se estudian los siguientes algoritmos debido a que han sido ocupados en varios trabajos posteriores [8,9,10,11,12,13,14,15,16] demostrando tener un alto grado de influencia, sin embargo, hay que remarcar que no son los únicos trabajos de impacto, dicho esto podemos encontrar en el grupo de los basados en tokens a los siguientes:

Algoritmo Suzuki-Kazami. En el algoritmo desarrollado por Suzuki y Kazami [17], cada nodo tiene un arreglo de tamaño N para almacenar el número de secuencia de la invocación más reciente del token de entre todos los nodos del sistema. Cuando un nodo quiere entrar en su región crítica y no tiene el token, incrementa su registro de contador y manda un mensaje de petición a todos los demás nodos, espera a recibir el token y cuando por fin lo recibe puede entrar en su región crítica. El mensaje del token contiene una cola de los nodos que han realizado su petición y el arreglo que contiene el número de secuencia más reciente al que se le ha dado el token. Después de salir de la región crítica, el nodo actualiza el arreglo con el número de secuencia más reciente con el valor del registro de su contador, y encolará todas las peticiones que no han sido procesadas en orden ascendente, si existe un proceso en la cola, le envía el token al que esté en la cabeza y sino mantiene el token.

Algoritmo de Raymond. En el algoritmo de Raymond [18] se usa una estructura lógica estática de árbol. Los nodos se organizan en una estructura de árbol sin raíz y solo se comunican con sus vecinos. La idea del algoritmo es la siguiente, si el nodo A tiene el token y tiene dos vecinos, B y C, estos saben que A tiene el token de forma directa, si a su vez B y C tienen vecinos directos, estos sabrán de forma indirecta quien tiene el token. Cada nodo tendrá una cola de peticiones. Si un nodo hace una petición a uno de sus vecinos y este no sabe de forma directa quien tiene el token, este vecino hará otra petición de vecino en vecino hasta que tenga la información. Cuando el nodo que tiene el token deja de ocuparlo se lo manda al primer nodo que tenga en su cola de peticiones y si tenía más peticiones encoladas se las manda al nuevo nodo que posee el token.

Algoritmo Naimi-Trehel. En este algoritmo [19] se utiliza una estructura lógica dinámica dentro de la red de comunicación. Los procesos que hacen peticiones se van organizando dinámicamente en una estructura de árbol con raíz, así cuando una petición de un nodo viaja por todo el camino para llegar a la raíz, este se convierte en el padre de todos los nodos excepto de él mismo, de esta forma este nodo se convierte en la nueva raíz. No es necesario tener una cola de peticiones pendientes pues cada nodo tiene dos variables, LAST Y NEXT. LAST indica el último nodo del que fue recibida una petición y el nodo vecino a quien este nodo enviará un mensaje de petición la próxima vez que quiera entrar en su región critica. NEXT indicará el nodo a quien el token se le dará una vez que el nodo haya dejado su región crítica. Así cuando un nodo *i* quiere entrar en su región crítica, éste mandará una petición a LASTi y esperará a que el token le llegue. La cola de peticiones pendientes puede ser deducida siguiendo el camino del estado NEXT en cada nodo. La cabeza de esta cola dinámica será el nodo a quien se privilegiará.

En cuanto a algoritmos relevantes basados en permisos podemos encontrar los siguientes:

Algoritmo de Lamport. Lamport en [20] ocupa un mecanismo basado en relojes lógicos para el orden total de las peticiones en el sistema. Cada nodo mantiene una cola de peticiones que contiene las peticiones ordenadas por marcas de tiempo, y también tiene un conjunto de peticiones de los nodos que necesita permiso para entrar en su región crítica. El algoritmo requiere que los mensajes se entreguen en orden FIFO entre cada par de nodos. Así cuando un nodo *i* quiere entrar en su región critica, manda un mensaje de petición a todos los nodos que necesita que le den permiso y coloca su petición en la cola de peticiones. Cuando el nodo *j* recibe la petición del nodo *i* regresa un mensaje de respuesta. El nodo *i* entrará en su región critica cuando reciba un mensaje de respuesta con una marca mayor de tiempo de todos los nodos que esperaba respuesta, y cuando la petición del nodo *i* sea la primera en la cola de peticiones. Cuando el nodo *i* termine la ejecución de su región critica quita su petición de la cola de peticiones y manda un mensaje a todos los nodos que están en su conjunto de peticiones.

Algoritmo de Ricart-Agrawala. En este algoritmo [21] se optimiza la propuesta de Lamport [20]. Cuando un nodo *i* quiere entrar en su región critica, manda una petición con marca de tiempo a todos los nodos en su conjunto de peticiones. Cuando un nodo *j* recibe la petición del nodo *i* manda un mensaje de respuesta al nodo *i* si el nodo *j* no está ejecutando o solicitando la región critica. En otro caso la solicitud se aplaza. El nodo *i* entra a la región crítica cuando recibe respuesta de todos los nodos de quienes esperaba recibir permiso. Cuando el nodo *i* sale de la región critica manda un mensaje a todas las peticiones aplazadas. Las solicitudes se ordenan utilizando números de secuencia en el sistema.

Existen algoritmos más recientes de exclusión mutua basados en tokens. La revisión de estos algoritmos es el aporte de este trabajo. Estos algoritmos se escogieron para su estudio pues ocupan técnicas distintas en su diseño. Dada la diferencia de enfoques se podrá realizar de mejor forma una comparación en la siguiente sección. Dicho esto, las propuestas que se listan son las siguientes:

Algoritmo Khah, Mazrae y Koroupi. Método basado en una topología de malla para sistemas que no interfieren en el registro de trabajos y con tráfico mínimo. Khah et al. [22] propone un algoritmo donde todos los nodos están conectados en una estructura lógica en forma de malla. Todos los nodos se representan como un grafo y para poder recorrerlos se ocupa el algoritmo de búsqueda por profundidad, al hacer este recorrido se va moviendo el token al siguiente nodo y con ello se evita la inanición. Cuando un nodo requiere entrar a su región critica manda una petición y hasta que recibe la autorización se bloquea. El algoritmo maneja tolerancia de pérdida de nodos, para ello se maneja un tiempo en las peticiones que continuamente se está censando, cuando se rebasa un tiempo de control previamente definido, si esa petición pertenecía a un nodo que se perdió, se borra. La tolerancia a la pérdida de un nodo se maneja de forma sencilla gracias a la topología de malla, esto debido a que todos los nodos están conectados, de tal forma que todo nodo tiene mínimo dos vecinos y a lo mucho cuatro, es decir no hay nodos aislados, por lo que si un nodo se pierde se puede redirigir el token hacia el vecino del nodo que se perdió

sin problema. En la figura 1 se muestra un diagrama general del sistema de malla redirigiendo el token ante la pérdida de un nodo.

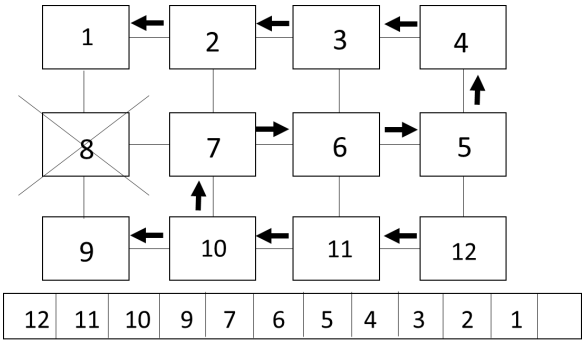

Fig. 1. Topología de malla con redireccionamiento en pérdida de un nodo. Imagen tomada de [22].

Algoritmo de Kanrar y Chaki. Kanrar et al. en [23] da un algoritmo basado en token con prioridades dinámicas para evitar la inanición de procesos y un mecanismo para generar un acceso justo a la región critica por cualquier proceso. Cuando un proceso quiere acceder a la región critica manda una petición para obtener el token, cuando este lo obtiene hace uso del recurso compartido, una vez que lo usa se libera el token para cualquier otro proceso que quiera entrar en la región critica. Si existen múltiples procesos compitiendo por el recurso se toma el que tenga la prioridad más alta. Al momento que el token se le brinda al proceso con mayor prioridad, se incrementa en uno la prioridad de los demás procesos que estaban compitiendo, esto con el propósito de que no puedan ser desplazados en otro intento de acceder al recurso compartido. De igual forma una vez que un proceso tiene la región critica, la prioridad del proceso se reinicia a su valor original. Para este algoritmo se tiene una estructura de grafo conectado dirigido con ciclos o sin ciclos. Haciendo uso de esta estructura la propuesta tiene tolerancia a fallas pues si un nodo se cae, un proceso de redirección se inicia tomando en cuenta los vecinos del nodo al que se intentaba mandar el token.

Algoritmo de Razzaque y Hong. Razzaque et al. [24] propone un algoritmo de exclusión mutua ocupando múltiples tokens basado en una estructura lógica de anillo. Cada proceso que compite por entrar en la región critica genera un token único que manda en una petición a todo lo largo del anillo, solo cuando su propio token regresa es cuando el proceso puede entrar en la región critica. Dentro de las condiciones que se asumen en el ambiente del sistema, se encuentra que los nodos pueden competir por múltiples recursos al mismo tiempo, pueden ocurrir fallas en los procesos o en la comunicación a través de la red, existe retraso en los mensajes pero es finito, así como la característica de que los mensajes son procesados en el orden que se envían. La propuesta tiene tolerancia de fallas pues si un nodo no recibe su propio token después de un cierto tiempo, ya sea porque el token se perdió o porque otro nodo murió, el nodo retransmite el token con la prioridad inicial con una bandera que indica que la petición se reenvió, una vez localizado el token se analiza si fue retransmitido o no, si así fue se descarta por ser un reenvío sino se agrega a la cola de peticiones.

Algoritmo de Neamatollahi, Taheri y Naghibzadeh. El algoritmo propuesto por Neamatollhi et al. en [25] ocupa una estructura de tipo torus bidimensional, esta propuesta tiene mejores resultados en condiciones de carga pesada pues su mejora viene en decrementar el número de mensajes en el sistema. En el caso de un sistema con carga de trabajo ligera se limita el número de mensajes a intercambiar. El token se mueve circularmente a lo largo de las columnas de la estructura torus. De forma simultánea se procesan peticiones nuevas generadas a lo largo de las filas, una vez que se genera una petición se alerta a los nodos de esa fila. La figura 2 muestra un diagrama de la estructura ocupada y un ejemplo de cómo el token circula sobre las columnas y cómo en el momento que se genera una petición en un nodo, este alerta a sus vecinos que se encuentran sobre la misma fila. Cuando un nodo es la intersección entre la columna donde se encuentra el token y la fila donde hay una petición para entrar a la región critica, este nodo rotará el token de forma horizontal hasta llegar al nodo que hizo la petición. Se comprueba la cola de peticiones y de existir una petición se le concede el token al proceso que hizo la petición, el token regresa al nodo donde se hizo la intersección de columna-fila y de ahí vuelve a realizar su movimiento vertical a lo largo de las columnas.

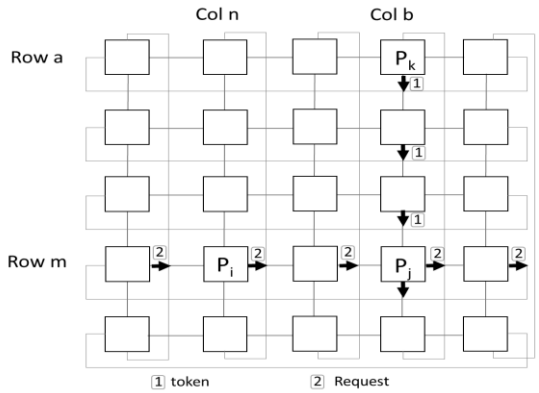

Fig. 2. Estructura de tipo torus bidimensional ejemplificando el funcionamiento del sistema. Imagen tomada de [25].

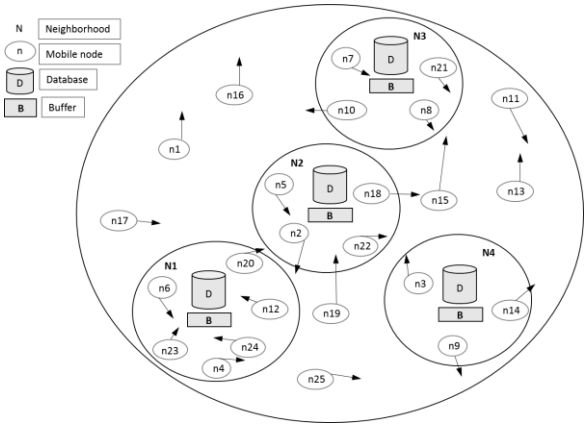

Fig. 3. Grupos locales que tienen el mismo recurso compartido. Imagen tomada de [26].

Cabe mencionar que en el algoritmo propuesto se asume que los procesos funcionan de forma correcta por lo que no hay tolerancia a fallos en caso de que el token se pierda o un nodo se caiga.

Algoritmo de Khanna, Singh y Swaroop. Este algoritmo considera las redes ad hoc para móviles. En este tipo de redes la compartición de recursos viene de forma inherente. Khanna et al. en [26] se centra en los *k* grupos locales compuestos por nodos cercanos que comparten el mismo recurso compartido. La figura 3 muestra cuatro grupos locales compartiendo el mismo recurso, en este caso una base de datos. El algoritmo aparte de un token también ocupa el concepto de líder. Después de que los nodos que pertenecen al grupo que comparten un recurso se han inicializado, se realiza un proceso para la elección de un líder, quien se elija generará un token y transmitirá su información. Dentro de la información del token se sabe el número de recursos libres en un momento dado. Cuando los demás nodos reciben esta información pueden mandar sus peticiones para entrar en la región crítica. Si el nodo que es líder quiere entrar en la región crítica y no tiene peticiones previas, este entra sin ningún retraso, si ya existía una petición le manda el token al nodo solicitante y este queda como líder. La propuesta tiene tolerancia a fallos ante la pérdida de un nodo a quien se le enviaba el token, esto se hace a través de un mensaje que manda el nodo que deja el grupo diciendo que lo abandonará, de esta forma el nodo que tiene el token lo mantendrá si no hay más peticiones o se lo enviará al próximo que haya hecho una petición.

3 RESULTADOS

En este trabajo se presentan varias propuestas para manejar las condiciones de competencia en los sistemas distribuidos, como resultado del mismo, se desarrolla una breve discusión sobre todas las propuestas en forma de resumen comparativo, este busca delimitar ventajas y desventajas entre todos los algoritmos. Todas las técnicas son algoritmos de exclusión mutua basados en tokens. Es claro que en muchas ocasiones los grafos fueron muy útiles para el intercambio de mensajes así como para tener información de los vecinos de un nodo, y así poder tener tolerancia a la pérdida de un nodo como se demostró en algunas de las investigaciones. Pero otras estructuras y topologías fueron propuestas para la solución del problema de la exclusión mutua, desde un torus bidimensional hasta un anillo lógico.

Otro aspecto a considerar fue si el algoritmo se ocupa en sistemas con carga de trabajo pesada o ligera, el ejemplo de esto fue el algoritmo de Neamatollahi et al [25]. Por otro lado Kanrar et al. [23] desarrolló un algoritmo que enfatiza en servir las peticiones en el orden que estas llegaron, y así evitar la inanición de procesos con un esquema de prioridades dinámicas, sin embargo, el número de mensajes que se intercambian en este trabajo puede ser un factor a mejorar.

El tipo de red que se ocupará también delimita una posible solución como se presentó en el trabajo de Khanna et al. [26] donde se considera el caso de las redes ad hoc para móviles ocupando un algoritmo con token con la característica de que solo lo moverá un líder predefinido.

Algunas aplicaciones de estos algoritmos se encuentran en Lejeune et al. [27], esta propuesta es una mejora del algoritmo de Kanrar, ya que Lejeune identifica que pueden existir violaciones de prioridades. Para lograr

esta mejora sin repercutir en el rendimiento de la propuesta original, se aplican un par de heurísticas. Así, en lugar de incrementar las prioridades cada vez que un nodo libera la región critica, ocupa una función asociada a una heurística con el propósito de que los procesos con prioridades bajas no se atiendan demasiado rápido, con ello se brinda un criterio más justo en la repartición del token. Por otro lado en Bindahman [28] se propone un algoritmo hibrido para mejorar las operaciones de bloqueo en recursos compartidos, esto lo logra al combinar el modelo basado en token y el multicast. Este algoritmo decrementa el consumo del ancho de banda y el tiempo de respuesta desde que se hace una petición hasta que se brinda el acceso. Bindahman ocupa como referencia para su propuesta el algoritmo de Razzaque [24]. La tabla 1 resume las topologías, la carga de trabajo del sistema así como algunas observaciones y aplicaciones de las propuestas presentadas.

Tabla 1. Comparación entre los algoritmos.

Algoritmo	Estructura	Observaciones	Aplicaciones
Khah et. al.	Malla lógica	Tolerante a fallas, se usa el algoritmo de búsqueda en profundidad para recorrer el grafo asociado.	En sistemas donde se manejan múltiples regiones criticas al mismo tiempo.
Kanrar et. al.	Grafo conectado o dirigido	Prioridades dinámicas, política primero en llegar primero en ser atendido, tolerante a fallas.	Lejeune [27] ocupando heurísticas mejoró el algoritmo en cuanto a la violación de prioridades.
Neamatollahi et al.	Torus bidimensional	Disminuye el número de mensajes en la comunicación, no es óptimo en sistemas de poco trabajo, no es tolerante a la pérdida del token.	En sistemas de carga pesada.
Razzaque et al.	Anillo lógico	Múltiples tokens, múltiples regiones críticas, no tiene una topología en especial asociada, tolerante a fallas, política primero en llegar primero en ser atendido.	Bindahman [28] ocupó este trabajo como referencia para su algoritmo hibrido.
Khanna et al.	Grupos locales de nodos vecinos	Esquema de k nodos en un mismo grupo compartiendo un mismo recurso compartido, existe un líder que es quien mueve el token, tolerante a fallas.	En sistemas que ocupan redes ad hoc móviles.

4 CONCLUSIONES

En este trabajo se presentaron algunos avances que se han desarrollado en los algoritmos de exclusión mutua de sistemas distribuidos, en particular, se pudo observar que los trabajos recientes se centran en algoritmos distribuidos basados en tokens. Sus principales características fueron descritas y se realizó una comparación breve entre ellos.

En dicha comparación se muestra que la propuesta de Neamatollahi es idónea en situaciones de carga de trabajo pesada, sin embargo, no termina de ser robusta debido a que no tiene tolerancia a fallos. Las demás propuestas sí son tolerantes a fallos. Khah y Kanrar hacen uso de una estructura de grafo y con ello ocupan algoritmos de búsqueda sobre estas estructuras que les permite ser más eficientes, aunque en ambas propuestas solo se considera una región crítica. En situaciones donde se deban de gestionar múltiples

regiones críticas la propuesta de Razzaque sería la que habría que implementar. Finalmente Khanna presenta una propuesta para ser ocupada en redes ad hoc para móviles, donde además se hizo la mezcla del concepto del token con el concepto de un líder haciendo su algoritmo interesante y eficiente.

Otro aspecto importante es el de considerar el número de mensajes que habrá que intercambiar en la comunicación entre nodos, para saber quién hizo una petición por el token y quien lo tiene, aquí se tiene un problema de optimización, situación donde se han buscado hacer muchas contribuciones en los trabajos recientes. Investigaciones venideras tendrán que hacer énfasis en estos puntos integrando las técnicas en tendencia como las heurísticas, aprendizaje automático, programación lineal entre otras con el propósito de lograr resultados cada vez más cercanos al óptimo y confiables bajo toda circunstancia.

REFERENCIAS

1. Tanenbaum, A. S.; Van Steen, M., *Distributed systems: principles and paradigms*. Prentice-Hall. 2007.
2. 2. Velazquez, M. G., A survey of distributed mutual exclusion algorithms. Colorado State University. 1993.
3. Saxena, P. C.; Rai, J., A survey of permission-based distributed mutual exclusion algorithms. *C omputer standards & interfaces*. 2003, 25(2), 159-181.
4. Maekawa, M.; Oldehoeft, A.E.; Oldehoeft, R.R., *Operating Systems, Advanced Concepts*. Benjamin-Cummings. 1987.
5. Silberschatz, A; Peterson, J. L., *Operating System Concepts*. Addison-Wesley, Alternate Edition. 1998.
6. Raynal, M., A simple taxonomy for distributed mutual exclusion algorithms. *Operating Systems Review*. 1991, 25(2), 47-49.
7. Singhal, M., A heuristically-aided algorithm for mutual exclusion in distributed systems. *IEEE Transactions on Computers*. 1989, 38(5), 651-662.
8. Agrawal, D.; El Abbadi, A., An efficient and fault-tolerant solution for distributed mutual exclusion. *ACM Transactions on Computer Systems.* 1991, 9(1), 1-20.
9. Chandy, K. M.; Lamport, L., Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems*. 1985, 3(1), 63-75.
10. Foster, I., *Designing and building parallel programs*. Addison Wesley Publishing Company, 1995.
11. Maekawa, M., An algorithm for mutual exclusion in decentralized systems. *ACM Transactions on Computer Systems*. 1985, 3(2), 145-159.
12. Manivannan, D.; Singhal, M., Decentralized token generation scheme for token-based mutual exclusion algorithms. *Computer Systems Science and Engineering* 1996, 11(1), 45-54.
13. Mellor-Crummey, J. M.; Scott, M. L., Algorithms for scalable synchronization on shared-memory multiprocessors. ACM Transactions on Computer Systems. 1991, 9(1), 21-65.
14. Neilsen, M. L.; Mizuno, M., A dag-based algorithm for distributed mutual exclusion. *In Distributed Computing Systems, 11th International Conference on*, IEEE. 1991, 354-360.
15. Singhal, M., A taxonomy of distributed mutual exclusion. *Journal of Parallel and Distributed Computing*. 1993, 18(1), 94-101.
16. Sunderam, V. S., PVM: A framework for parallel distributed computing. *Concurrency: practice and experience*. 1990, 2(4), 315-339.
17. Suzuki, I; Kasami T., A distributed mutual exclusion algorithm. *ACM Transactions on Computer Systems*. 1985, 3(4), 344-349.
18. Raymond, K., A tree-based algorithm for distributed Mutual Exclusion. *ACM Transactions on Computer Systems*. 1989, 7(1), 61-77.
19. Naimi, M; Trehel, M., How to detect a failure and regenerate the token in the log(n) distributed algorithm for mutual exclusion. *Proc. Of the Second Workshop on Distributed Algorithms*, *Lecture Notes in CS*. 1987, 155-166.
20. Lamport, L., Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*. 1978, 21(7), 558-565.
21. Ricart, G; Agrawala, A., An optimal algorithm for mutual exclusion in computer networks. *Communications of the ACM*. 1981, 24(1), 9-17.
22. Khah, Hussein N.; Mazrae, A. A.; Koroupi, F., Presenting a new algorithm for management mutual exclusion in distributed systems by

connecting mesh. *International Journal of Advanced Research in Computer Science*. 2013, 4(10).

23. Kanrar, S., Chattopadhyay, S., & Chaki, N. A New Link Failure Resilient Priority Based Fair Mutual Exclusion Algorithm for Distributed Systems. *Journal of network and systems management*. 2013, 21(1), 1-24.

24. Razzaque, M. A.; Hong, C. S., Multi-token distributed mutual exclusion algorithm. *Advanced Information Networking and Applications*, AIN, *22nd International Conference on*, IEEE. 2008, 963-970.

25. Neamatollahi, P.; Sedaghat, Y.; Naghibzadeh, M., A simple token-based algorithm for the mutual exclusion problem in distributed systems. *The Journal of Supercomputing*. 2017, 1-18.

26. Khanna, A.; Singh, A. K.; Swaroop, A., A Token-Based Solution to Group Local Mutual Exclusion Problem In Mobile Ad Hoc Networks. *Arabian Journal for Science and Engineering*. 2016, 41(12), 5181-5194.

27. Lejeune, J.; Arantes, L.; Sopena, J.; Sens, P., A fair starvation-free prioritized mutual exclusion algorithm for distributed systems. *Journal of Parallel and Distributed Computing*. 2015, 83, 13-29.

28. Bindahman, S.; Yong, C. H., Performance enhancement for Distributed Lock Manager using Hybrid Multicast and Ring algorithm. *In Digital Information and Communication Technology and it's Applications.* 2012 *Second International Conference on*. IEEE, 2012, 388-393.

Acerca de los autores

M.C. Sánchez Ruiz, estudió la Licenciatura en Ingeniería en Ciencias de la Computación en la Benemérita Universidad Autónoma de Puebla (BUAP), y la Maestría en Ciencias de la Computación en la línea de investigación de Sistemas Distribuidos también en la BUAP. En su tesis de maestría trabajó en el área de Reconocimiento de Patrones y Métodos Estocásticos. Sus temas de interés son Aprendizaje Automático, Visión artificial y Reconocimiento de Patrones mediante métodos estadísticos, con especial atención en ámbitos médicos y análisis de personalidad. Actualmente, se encuentra realizando los trámites necesarios para ingresar al programa de doctorado en Ciencias Computacionales en el Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE).

Dra. Hilda Castillo Zacatelco, estudió la Licenciatura en Computación en la Benemérita Universidad Autónoma de Puebla (BUAP), posteriormente estudió la Maestría en Ciencias de la Computación en la BUAP, y el doctorado en Ciencias Computacionales en el Centro de Investigación y Desarrollo Tecnológico (CENIDET) de Cuernavaca. Es Profesor-Investigador en la Facultad de Ciencias de la Computación, BUAP; es miembro de la academia de Programación y de la de Software de base, e imparte cursos de estas áreas; forma parte del cuerpo académico Cómputo Distribuido. Actualmente sus temas de interés son los Objetos de Aprendizaje, los Sistemas Distribuidos, el problema de Bin Packing y el Diseño de Algoritmos.

Dra. Claudia Zepeda Cortés, estudió la Licenciatura en Computación en la Benemérita Universidad Autónoma de Puebla, posteriormente estudió la maestría en Ciencias con Especialidad en Sistemas Computacionales en la Universidad de las Américas Puebla, y el doctorado en Ciencias Computacionales en conjunto entre el INSA de Lyon-Francia y la Universidad de las Américas Puebla. Es Profesor-Investigador en la Facultad de Ciencias de la Computación, BUAP; es miembro de la academia de Matemáticas y de la de Teoría de la Computación, e imparte cursos de estas áreas; forma parte del cuerpo académico Cómputo Distribuido. Actualmente sus temas de interés son los Objetos de Aprendizaje y las aplicaciones en Programación Lógica.

Dr. Rafael de la Rosa Flores, estudió la Licenciatura en Computación en la Benemérita Universidad Autónoma de Puebla (BUAP), posteriormente estudió la Maestría en Ciencias de la Computación en la BUAP, y el doctorado en Ciencias Computacionales en el Centro de Investigación y Desarrollo Tecnológico (CENIDET) de Cuernavaca. Es Profesor-Investigador en la Facultad de Ciencias de la Computación, BUAP; es miembro de la academia de Bases de datos e Ingeniería de Software y de la de Software de base, e imparte cursos de estas áreas; forma parte del cuerpo académico Cómputo Distribuido. Actualmente sus temas de interés son los Objetos de Aprendizaje, los Sistemas Distribuidos, el problema de Bin Packing y los Dispositivos Móviles.

M.C. Ana Patricia Cervantes Márquez, estudió la Licenciatura en Computación en la Benemérita Universidad Autónoma de Puebla (BUAP), posteriormente estudió la maestría en Ciencias con Especialidad en Sistemas Computacionales en la Universidad de las Américas Puebla. Actualmente es Profesor-Investigador en la Facultad de Ciencias de la Computación, BUAP, imparte cursos del área de programación básica y software de base, y está realizando su doctorado en la Universidad Popular Autónoma del Estado de Puebla en el área de Ingeniería de Software. Es colaboradora del cuerpo académico de Cómputo Distribuido. Sus temas de interés son los Objetos de Aprendizaje, la Calidad en Uso y el Diseño de Algoritmos.

C. Misael Limón Martínez es estudiante de la Ingeniería en Ciencias de la Computación en la Facultad de Ciencias de la Computación de la Benemérita Universidad Autónoma de Puebla (BUAP). Ha participado en varios programas de Haciendo Ciencia en apoyo a proyectos de investigación de la BUAP. Cursa actualmente el sexto cuatrimestre de la carrera. Sus temas de interés son los Objetos de Aprendizaje y los Sistemas Operativos.

El Dr. José Luis Carballido Carranza, estudió su doctorado en Matemáticas en la Facultad de Ciencias Físico Matemáticas, con la tesis titulada “Fundamentos Matemáticos de la Semántica P-stable en Programación Lógica”.