A Distributed Algorithm for Power Management in Mobile Ad-Hoc Networks

Bita Amirshahi,
Assistant Professor, Department of Software Engineering, Payamenoor University, Tehran, Iran
Corresponding author: amirshahi@tpnu.ac.ir
Arkady Barsky
Professor, Department of Software Engineering, Moscow state university of communication means, Moscow, Russia
Arkbarsk@miit.ru

Abstract

Resource sharing in distributed network becomes known as a major problem when systems have physical or virtual resources accessible by more than one process at a time. Mutual Exclusion, is proposed as a solution to guarantee the correctness of read/written information from/in the shared resource or critical sections (CS).Mobile ad hoc Network is an unstable network formed dynamically by a connection of wireless mobile nodes without the use of an existing network substructure. Since energy consumption during communication is a major depletion parameter the number of communication must be reduced as much as possible to achieve extended battery life. Since battery technology does not grow as rapidly as CPU or memory does so there is a strong need for the presence of protocols which are as energy efficient as effective. In this paper we present a new fully distributed token–based mutual-exclusion algorithm for clustered Mobile ad-hoc network with message count reduction purpose which eventuates to power optimization. Therefore we use Raymond's algorithm (which is well-known in stepping down message complexity) within the clusters and perpetual Mobility of token (that is so efficient in high load network such as Manet's) between the clusters.

Keywords: Mobile ad-hoc network, mutual exclusion, Power management, Power optimization, critical section

Introduction

Energy consumption is a significant factor in wireless networks with battery-powered nodes and transceivers as a dominant energy consumer .Also mutual exclusion is a fundamental problem in distributed systems. “Mutual exclusion” allows only one process at a time to access the shared resource and the others have to be locked and wait for their turns. Two general approaches for mutual exclusion in distributed computing are centralized and distributed. In a centralized approach one process is considered as a coordinator for CS accession. Any process invoking CS should submit its requirement to above node and waits for permission. Having a single point of failure and bottleneck in coordinator are the challenges of this approach. In a distributed approach there are two major groups of algorithms which are Token based [4] [5] [6] [7] [8] and permission based [12] [13] [14] [15] [16].In token based group a unique token is shared among the nodes. A node is allowed to enter the CS only if it holds the token. Two methods can be considered for such situation, i.e, the perpetual mobility of the token and token-asking method. In the perpetual mobility, the token travels from one process to another to give them the right to enter their critical section exclusively without paying attention to whether that process needs the token or not. Token ring [17] algorithm is one of these algorithms .In token asking methods, a process which is attempting to access CS, if it is not the token-holding process, requests to receive the token and waits for the token arrival. After having finished CS execution, the token-holding process chooses a requesting process and sends it the token. In permission based group a node enters a CS after receiving permission from all of the nodes in its set for the critical section. In this paper we proposed a new energy efficient mutual exclusion algorithm for mobile ad-hoc network. The whole network is divided into clusters. The proposed algorithm is token-based and distributed. Global or centralized controller is absent in this network. As the Raymond's algorithm has low message complexity so we use Raymond's algorithm between the clusters. Regarding the effectiveness perpetual mobility of the token in high load networks we use this method within the clusters. The proposed algorithm significantly reduces the message complexity and hence the energy consumption of the system is optimized.

Assumption

The system is constituted of mobile nodes that are arranged in clusters. Each node is assigned a unique id, each cluster has a cluster head. Nodes communicate by message passing Type of message transmitted in the network is unicast.

We assume that lower layers of the network, such as mac layer and transport layer, ensure reliable delivery of unicast messages to ensure reliability, retransmission mechanism is used in lower layers in case of having lost packets due to noise or interference and also packets are received in the same order as they are sent. Receiving node can receive messages without any error or distortion .The message size is constant and eventually negligible. Nodes are failure free and have a limited amount of time when executing its CS. There is no limitation in the Band width of the network and also no token lost as a result of collision occurrence and also it is assumed that N=C^2 where N is the number of processes and C is the number of clusters therefore the network consists of $\sqrt{N}$ clusters and in each cluster $\sqrt{N}$ processes exist.

Related work

A Dutch Computer Scientist called Edsger Wybe Dijkstra for the first time solved the mutual exclusion problem [18] considering mutual exclusion problem conduced to two approaches which categorized as centralized approach and distributed approach. In the centralized approach a process takes the responsibility as CS coordinator. In the distributed approach there are two kinds of algorithm namely, token based and permission based. In the permission based algorithms if a node needs to use CS it should take permission from all other nodes in its set and there are many algorithms in this class [19] [20] [21] [22]. In the Lamport algorithm a node which invokes CS should send its request and then waits and be locked until receiving acknowledgments from all other nodes. The message complexity of this algorithm is 3(N-1). Ricart Agrawala has made improvements to previous algorithm in order to reduce message complexity to 2 (N-1) messages per each CS entrance [13] by removing release message step of Lamport's algorithm. Agrawala and El Abbadi [21] and Maekawa [20] introduced quorum- based algorithms which reduce message complexity dramatically [20] [21] and their algorithm is in permission-based groups. This algorithm require Olog2 (N) message communication in best case and O (N) in worst case. Mikawa introduced a new mutual exclusion algorithm with $3\sqrt{n}$ message transmission for every CS invoking. This network consists of number of sets with nonempty inter section [23] [24]. There are also a number of token based algorithms [4] [5] [25] [26] [27] [28] [29] [30], in which the node which has obtained the token (virtual object) either by perpetual mobility or token-asking method will have the opportunity to enter CS. One of famous approaches in this group is Suzuki-kasami's [4]. Another one is Raymond's [5] which reduces the message complexity by using minimum spanning tree Raymond's algorithm is the basis for our proposed approach which is explained later. There are also so many other kinds of MX algorithms such as Edmondson [32] that concentrates on QOS or a new MX [31] on path reversal or [1] [2] [3] which focus on opportunistic network , energy consumption or voting based protocols .

The proposed Algorithm

Our new approach guarantees Mutual exclusion in cluster based Mobile ad-hoc networks and is also energy efficient.
This method is fully distributed and token-based so redounds to outstanding fault tolerance and easeful implementation. Token is a unique virtual object and moves perpetually on a logical ring between clusters and also along minimum spanning tree edges within clusters. As regards to efficiency of perpetual mobility of the token in network with high-load attribute, since Manet's usually contains applications with heavy requirements so we use this token trait between clusters. Besides Raymond's algorithm has low energy consumption which is the outcome of its low message complexity and as it is well established that energy consumption is a significant challenge in Manet's so the proposed approach use Raymond's algorithm within the clusters.
The proposed algorithm is defined in 3 steps:

- Outline Design
- Data Structure And Message-type description
- Algorithm Presentation

A. Outline Design:

Each node has a unique ID and communicates with its immediate-neighbor with unicast message and through FIFO ordered channels. We have two kinds of nodes called ordinary and cluster leader (header). In each node some variables maintain as follows:
HOLDER variable that contains the identity of a node that thinks has the token or leads to the node having the token , could be the node itself (in case it is a token-holding node) or the cluster-leader (in case the token is out of the cluster).
USING variable indicates if the current node is executing the critical section or not and contains true or false values.ASKED variable indicates if node has sent a request for the token and also prevents the sending of duplicate requests for token hence makes sure that the request queues of the various nodes do not contain any duplicate elements.REQUEST QUEUE consists of the identities of those immediate neighbors that have requested for token but have not yet been sent the token and contains “self” value if the node makes a request for the token for its own use.
The maximum size of REQUEST QUEUE is the number of immediate neighbors +1(for “self”).CS-Permission is a variable assigned just to leaders and contains ZERO value or the leader identifier to indicate whether a leader can enter its CS or

not. For the following section we assume that leader node (K), is the token holding node and process pi in a different cluster is a requesting node and also $\sqrt{N}$ clusters and $\sqrt{N}$ nodes in each cluster is defined.

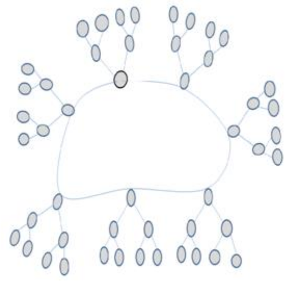

Figure1. an example of a cluster-based manet

B. Data Structure and Message Type Description:

When a non-token holding process as pi invokes CS permission as above it sets ASKED variable true (this makes the REQUEST queue of any node bounded even when operating under heavy load) then queue “self” in its REQUEST-Q and makes a request routine to the token-holding node which is indicated by its HOLDER variable (in case it has not already done so on behalf of itself or some other nodes)which according to our assumption as token is out of recent cluster so the request is sent to leader cluster . In case a cluster leader invokes CS it waits until token arrival then sets the CS-Permission to cluster identity and fulfills differed requests in its REQUEST-Q and then the node itself enters CS. It is noticeable that in this case to avoid starvation; leader's node just does CS requirements which are queued before token arrival. And requests which are generated after that will be done in next arrival of the token in its logical circulation. After the token is achieved either by ordinary or leader node, the node sets the USING variable true and enters its CS. On releasing the CS two possible situations are described as follows:

1) The node is ordinary node so it sets the USING variable FALSE and sends the token to next requesting node in its FIFO queue and if the queue was empty it sends the token to the leader node to be released in its ring to continue its perpetual mobility.

2) The node is a cluster leader so it sets the CS-Permission ZERO and releases the token.

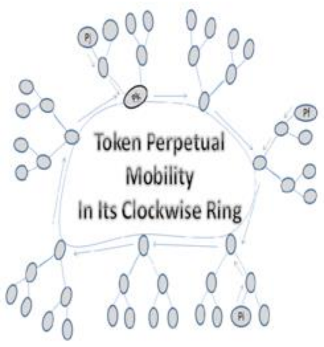

Figure 2. Node Pi invokes to enter its CS (A Sample Senario)

C. Algorithm Presentation:

Initialization: For all processes ASKED: =FALSE USING: =FALSE REQUEST-Q is empty and CS-Permission for all leaders 0.

```
Ordinary node pi invokes for CS
ASKED (i):=TRUE
En queue (REQUEST-Q (i),"self")
 /*inserts its request in the rear of its queue */
Create Request Procedure (i)
Sends Request to its immediate neighbor which leads to token as indicated by its HOLDER variable.
/*unicast message to immediate neighbor in minimum spanning tree*/
WHILE (USING (i) is FALSE)
Leader node invokes for CS
ASKED (i):=TRUE
En queue (REQUEST-Q (i),"self")
 /*inserts its request in the rear of its queue */
WHILE (CS-Permission is ZERO)
```

```
Ordinary node pi releases CS
USING: =FALSE
Remove (REQUEST-Q,"self")
If REQUEST-Q is not empty send
 Token to the head of above queue
Else
Send the token to cluster-leader
Leader node releases CS
USING: =FALSE
CS-Permission: =zero
Release token to its logical ring.
```

Analysis of the proposed Algorithm

A. Proof of Correctness:

The Correctness of new algorithm is tested according to safety and live ness attributes of it.

Safety is assured.

The algorithm ensures that at any moment not more than one node holds the token.

Whenever a node receives a token, it becomes token-holding node, whenever a node sends a token; it becomes a non-token holding node.

Between the instant one node becomes non-token holding and another node becomes token-holding, there is no other token-holding node, since the token is unique in the system thus, there is at most one-token holding node at any point of time in the network.

Live ness is assured

It is well-established that live ness implies freedom of deadlock and starvation. First we study deadlock second switch to starvation.

Deadlock could happen due to any of the following situations.

The token cannot be transferred to a node because no node holds the token, this assumption is incorrect as in the beginning of the algorithm, and PK is the token-Holding node.

The node which is possession of the token is not conscious that there are other nodes requiring the token, this event can never happen in as nodes are assumed failure free and the lower layers of the network such as mac and transport are reliable so there is no packet loss and disordering.

Starvation is due to listed situation as follows:

One cluster keeps the token so nodes are the other clusters encounter starvation. This situation is impossible as the new requests which are generated after token arrival should wait until the next token arrival in its perpetual movement and so the token would not be kept in above cluster.

The token-holding node keeps the token forever

If in a node request-Q becomes empty the token-holding node will pass the token to the cluster-leader to continue its perpetual mobility and becomes non-token holding process.

If request-Q wasn't empty so the token-Holding node will send the token to the next process which is located in the head of queue then becomes non-token holding process.

B. Message complexity evaluation

We estimate message complexity under two distinct situations as follows:

- Message complexity under low-load condition in which only one process invokes its CS, therefore it makes request-procedure directly to its immediate-neighbor which is indicated by the Holder variable leads to token-holding node which according to our assumption (token is out of cluster) is cluster leader.Hence $\log\sqrt{N}$ message exchange are required within cluster and as regards to perpetual mobility of token $\sqrt{N}$-1 message exchange is required between clusters thus the total message exchanged in this situation become ($\log\sqrt{N}+\sqrt{N}-1$) which is approximately $O(\sqrt{N})$ as $\log\sqrt{N} \ll \sqrt{N}$
- Message complexity under heavy-load condition in which all nodes invoke their critical section simultaneously and repeatedly.

As respects to $\sqrt{N}$ node existence in each cluster then the total message exchange for inter cluster is $\sqrt{N}\log\sqrt{N}$ but as it is assumed that before token arrival just requirements of half of the nodes in each cluster reached the leader so the average message exchange is estimated as $\frac{\sqrt{N}}{2}\log\sqrt{N}$. On the other hand, token transferring from one cluster leader to another needs one message exchange hence the total message exchange is $\frac{\sqrt{N}\log\sqrt{N}}{2}$ +1 for each cluster and therefore $(\frac{\sqrt{N}\log\sqrt{N}}{2}+1)\sqrt{N}$ for all the network that result in $\frac{(\frac{\sqrt{N}\log\sqrt{N}}{2}+1)\sqrt{N}}{N}$ for any node individually.

Then we can simplify the above formula to $\frac{\log\sqrt{N}}{2}+\frac{1}{\sqrt{N}}$ that as $\frac{1}{\sqrt{N}} \ll \log\sqrt{N}$ become to $\log\sqrt{N}$.

Discussion and conclusion

In this paper we have presented a new distributed token based algorithm to handle mutual exclusion problem in mobile ad hoc network. As perpetual mobility of token is so efficient in network with heavy applications and as regards to eligible message complexity resulted in token asking schemes so in order to have the advantage of two schemes the proposed protocol uses both attributes of the token. Also the reduction of message count (that is remarkable under heavy load) leads to have lower energy consumption and this is an important factor for wireless systems in general therefore this makes our algorithm more suitable to be used in mantes. The algorithm works on a clustered graph which executes Raymond's algorithm within clusters, while running token perpetual mobility between clusters leaders.By storing the Cs's requests in cluster leader and serving them after token arrival safety and live ness properties are assured. Generally in light demand message complexity is $\sqrt{N}$ which is almost equal to some fore-proposed algorithms [9] [11] but in heavy demand it is $\log\sqrt{N}$ per CS invocation which is dramatically better than other algorithms proposed with power optimization purpose. The authors would rather to focus on energy-optimization in vehicular ad-hoc networks as a future work with dynamic minimum spanning tree within the clusters.

References

[1] tamhane2010token,Token based algorithm for supporting mutual exclusion in opportunistic networks,Tamhane, Sagar A and Kumar, Mohan, Proceedings of the Second International Workshop on Mobile Opportunistic Networking,p126--134,2010,ACM

[2] norouzi2012energy,Energy Consumption Analysis of Routing Protocols in Mobile Ad Hoc Networks,Norouzi, Ali and Zaim, A Halim,Real-Time Systems, Architecture, Scheduling, and Application, Dr. Seyed Morteza Babamir (Ed.), ISBN, pages978--953,2012

[3] kanrar2013new,A new voting-based mutual exclusion algorithm for distributed systems,Kanrar, Sukhendu and Chattopadhyay, Samiran and Chaki, Nabendu,Engineering (NUiCONE), 2013 Nirma University International Conference on,p1--5,2013,IEEE

[4] I. Suzuki and T. Kasami, "A distributed mutual exclusion
algorithm," ACM Transactions on Computer Systems, vol. 3, no.4, pp. 344–349, 1985.

[5] K. Raymond, "A tree-based algorithm for distributed mutual exclusion," ACM Transactions on Computer Systems, vol. 7, no.1, pp. 61–77, 1989

[6] I. Suzuki and T. Kasami. A distributed mutual exclusion algorithm. In ACM TOCS, 1985.

[7] L. Lamport. Time, clocks, and the ordering of events in a distributed system,. In Communications of the ACM 21,, 1978.

[8] G.Ricart And A. K Agrawala. An optimal algorithm for mutual exclusion in computer networks. In Commun.ACM, Jan. 1981.

[9] neamatollahi2012info,Info-based approach in distributed mutual exclusion algorithms,Neamatollahi, Peyman and Taheri, Hoda and Naghibzadeh, Mahmoud,Journal of Parallel and Distributed Computing,vol.72,no.5, pages650--665,2012,Elsevier

[10]sharma2014token,A Token Based Protocol for Mutual Exclusion in Mobile Ad Hoc Networks,Sharma, Bharti and Singh, Awadhesh Kumar},Journal of information processing systems,vol.10,1,p36--54,2014

[11] taheri2011hybrid,A hybrid token-based distributed mutual exclusion algorithm using wraparound two-dimensional array logical topology,Taheri, Hoda and Neamatollahi, Peyman and Naghibzadeh, Mahmoud, Information processing letters,
Vol.111, no.17,p841--847,2011,Elsevier

[12] L. Lamport, "Time, clocks, and the ordering of events in a
distributed system," Communications of the ACM, vol.21,no.7, pp. 558–565, 1978.

[13] G. Ricart and A. K. Agrawala, "An optimal algorithm for mutual exclusion in computer networks," Communications of the ACM,vol. 24, no. 1, pp. 9–17, 1981.

[14] Hsiou Mien Lien amd Shyan-Ming Yuan. A new approach of coiistructiiig informatioii structure for mutual exclusioii in distributed systems. In IEEE, 1994.

[15] M.Maekawa. A √N algorithm for mutual exclusion in decentralised system. In ACM Trans.Comput.Syst., May 1985.

[16] Sandeep Lodha and Ajay Kshemkalyani. A fair distributed mutual exclusion algorithm. In IEEE Transactions on Parallel and Distributed Systems, vol. 11, 2000.

[17] G. Le Lann, Distributed systems towards of a formal approach, in: IFIP Congress, North-Holland, 1977, pp. 155–160.

[18] E.W. Dijkstra, Solution of a problem in concurrent programming control, Communications of the ACM 8 (9) (1965) 569.

[19] M. Singhal, "A heuristically-aided algorithm for mutual exclusion for distributed systems," IEEE Transactions on Computers, vol. 38, no. 5, pp. 70–78, 1989.

[20] M. Maekawa, "A root N algorithm for mutual exclusion in
decentralized systems," ACM Transactions on Computer Systems,vol. 3, no. 2, pp. 145–159, 1985.

[21] D. Agrawal and A. El Abbadi, "An efficient solution to the
Distributed mutual exclusion problem," ACM Transactions on
Computer Systems, vol. 9, no. 1, pp. 1–20, 1991.

[22] S. Lodha and A. Kshemkalyani, "Afair distributedmutual exclusion algorithm," IEEE Transactions on Parallel and Distributed Systems, vol. 11, no. 6, pp. 537–549, 2000.

[23] O. S. F. Carvalho and G. Roucairol, "On mutual exclusion incomputer networks," Communications of the ACM, vol. 26, no.2, pp. 146–147, 1983.

[24] M. Challenger, V. Khalilpour, P. Bayat, andM. R.Meibodi, "A new robust centralized DMX algorithm," in Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks (PDCN '07), pp. 367–374, Innsbruck,Austria, February 2007.

[25] A. S. Tanenbaum, Distributed Operating System, Pearson Education,2007.

[26] G. Hosseinabadi and N. H. Vaidya, "Exploiting opportunistic overhearing to improve performance of mutual exclusion in wireless Ad Hoc networks," in Wired/Wireless Internet Communication, vol. 7277 of LectureNotes in Computer Science, pp. 162–173, 2012.

[27] P. Chaudhuri and T. Edward, "An O(√n) distributed mutual exclusion algorithm using queue migration," Journal of Universal Computer Science, vol. 12, no. 2, pp. 140–159, 2006.

[28] F. Mueller, “Prioritized token-based mutual exclusion for distributed systems,” in Proceedings of the 12th International Parallel Processing Symposium and 9th Symposium on Parallel and Distributed Processing, pp. 791–795, April 1998.
[29] M. L. Neilsen and M. Mizuno, “A dag-based algorithm for
Distributed mutual exclusion,” in Proceedings of the 11th International Conference onDistributed Computing Systems, pp. 354–360, May 1991.
[30] M. Naimi and M. Trehel, “A distributed algorithm for mutual exclusion based on data structures and fault tolerance,” in Proceedings of the 6th International Phoenix IEEE International Conference on Computer Communications, pp. 35–39, Scottsdale,Ariz, USA, 1987.
[31] M. Naimi, M. Trehel, and A. Arnold, “A log (N) distributed mutual exclusion algorithm based on path reversal,” Journal of Parallel and Distributed Computing, vol. 34, no. 1, pp. 1–13, 1996.
[32] J. Edmondson, D. Schmidt, and A. Gokhale, “QoS-enabled
Distributed mutual exclusion in public clouds,” in On the Move to Meaningful Internet Systems: OTM, vol. 7045 of Lecture Notes in Computer Science, pp. 542–559, 2011.