이동 에드 혹 네트워크를 위한 제어노드 선택 알고리즘 및 성능 평가

Pradeep Parvathipuram† · 양 기 철†† · 오 수 열††

요 약

이동 에드 혹 네트워크(Mobile Ad hoc Network)에서는 노드들이 pear-to-pear 레벨에서 무선 채널을 통하여 대화 한다. 노드들은 지리적으로 이동이 자유로우며 무선 채널들은 영역 내에서 약하게 결합된다. 그리고 노드들이 자유롭게 이동할 수 있기 때문에 고정된 네트워크의 형태가 존재하지 않는다. 따라서 노드들간의 데이터 교환이나 노드간의 통신을 관리하기 위해서 리더 노드(Leader Node)가 필요하다. 본 논문에서는 이웃 노드간에만 대화가 허용되는 이동 에드 혹 네트워크를 위한 효율적인 리더 노드 선택 알고리즘을 소개한다. 그리고 시뮬레이션을 통한 알고리즘의 성능 평가 결과를 보인다. 본 논문에서 소개한 알고리즘은 노드들의 이동에 영향을 받지 않고 최소한의 무선 자원만을 사용하는 효율적이고 실용적인 알고리즘이다.

A Leader Election Algorithm and Performance Evaluation for Mobile Ad hoc Networks

Pradeep Parvathipuram† · Gi-Chul Yang†† · Sooyul Oh††

ABSTRACT

Nodes communicate through wireless channels under peer-to-peer level in ad-hoc mobile networks. The nodes are free to move around in a geographical area and are loosely bounded by the transmission range of the wireless channels. Also, a node is completely free to move around, there is no fixed final topology. Hence, to manage the inter-node communication and data exchange among them a leader node is required. In this paper we introduce an efficient leader election algorithm for mobile ad hoc networks where inter-node communication is allowed only among the neighboring nodes. Furthermore we present the result of performance evaluation through simulation. The algorithm is efficient and practical since it uses least amount of wireless resources and does not affect the movement of the nodes.

키워드 : 이동 에드 혹 네트워크(Mobile Ad hoc Networks), 제어노드 선택(Leader Election)

1. Introduction

The leader election problem originally appeared in token ring network for managing the use of tokens [1]. The problem resurfaced again in ad hoc or dynamic networks, however, with added complexity. Since then a number of papers have discussed the nature of algorithms [2-5] and to our knowledge only few papers [2, 3] have presented such algorithms. One of the main reasons for such lack of work in this area is the complexity involved in finding an efficient solution for a highly dynamic network. Unlike conventional distributed systems, designing a leader election algorithm for mobile ad hoc networks is challenging mainly because (a) the nodes are highly mobile, (b) the mean time failure of these nodes are relatively high compared to static wired network nodes, (c) the transmission range and bandwidth of wireless channels are limited, (d) *neighbor* configuration may change randomly, and (e) there is no fixed network topology. For these reasons, an efficient leader election algorithm must do everything, which an algorithm for static networks does, and in addition it must handle the movement of the nodes. Furthermore, the algorithm must guarantee that (a) there should be a leader at the end of the execution of the algorithm and (b) there cannot be more than one leader in a single connected component.

※ This work was supported by Mokpo National University through International Cooperative Research Program.
† 비 회 원 : Sprint Corporations, System Developer
†† 정 회 원 : 목포대학교 정보공학부 교수
논문접수 : 2004년 5월 3일, 심사완료 : 2004년 7월 27일

Our aim is to develop an efficient leader election (identification) scheme for this highly dynamic system that (a) incurs minimum messaging cost and time to elect a leader (b) is correct, that is, at any time there cannot be two or more conflicting leaders in the network, and (c) it does not hinder or restrict the geographical movement of nodes. In our approach we do not impose any specific structure for the mobile ad hoc network.

2. The Algorithm

The algorithm basically selects the largest identity node as the leader using minimum wireless messages. A mobile ad hoc network can be considered as a static network with frequent link or node failures, which can be thought of as a mobile node of an ad hoc network going out of reach. We use the *diameter* concept to cover all the nodes in the network. A *diameter* is defined as the longest *distance* between any two nodes in the network where the *distance* is defined as the shortest path between the nodes. In (Figure 2), the diameter of the network is 6, which is the distance between the nodes L and I. The distance metric is measured in number of hops. We assume the network gets stabilized after a single change occurs during leader election process. We further assume that there are only a finite number of changes in the network. The steps of the algorithm can be stated briefly as follows and later we provide the pseudo code for the algorithm.

Consider a network of N nodes. Our algorithm may take more than diameter rounds to terminate since the topological changes are considered during the leader election. If, however, the topological changes are not considered, then it takes diameter rounds to elect the leader.

2.1 Leader election

For each round, each node propagates its unique identifier to its neighbors and a maximum identifier is elected as a leader. This maximum identifier is propagated in the subsequent rounds. All the rounds need to be synchronized. idlist (i) identifies identifier list for $node_i$, which consists of all the neighbors for $node_i$. Lid(i) = max(idlist(i))

2.2 Termination

At (rounds >= diameter), for each $node_i$

If all identifiers in idlist (i) are the same, then the $node_i$ stops sending the maximum identifier further and elects the maximum identifier in the idlist (i) as the leader. The termination may not be at the end of the diameter rounds, the algorithm gets terminated if for each $node_i$ the elements in idlist (for each node) are the same.

Algorithm :
For each node i in the network, we have the following.
idlist - Identifier list, lid(i) - leader id of node i.
For each round,
Begin
Each node say $node_i$ transmits its unique identifier in the first round and Lid(i) in the subsequent rounds to their neighbors and all these ids will be stored in idlist.
Lid(i) = max (idlist(i));
End
A unique leader is elected in diameter rounds, if there are no topological changes in the network. The algorithm is modified to incorporate topological changes in between the rounds and below is the description of how the algorithm is modified.

- ***Case 1*** : *If a node has no outgoing links then lid(i) = i;*
- ***Case 2*** : *If a node leaves between the rounds, then the neighbors would know this. Suppose $node_i$ leaves the network after round r and let its neighbors be j and k.*

Begin
∀ neighbors of i (i.e. j, k).
Delete (ilist, idlist(j & k)) *// delete ilist from idlist*
Where ilist contains the group of identifiers that $node_i$ has sent to its neighbors before round r along with i.
The ilist information is also deleted from all the neighbors of j and k if the ilist identifiers have been propagated in the previous rounds. This process continues until all the nodes in the network are covered.
While (round > = diameter), // Termination condition
Begin
∀ nodes in the network
say for $node_i$,
compare all the identifiers present in idlist(i)
If all the identifiers in idlist(i) are equal, $node_i$ stops propagating its maximum identifier and elects the maximum identifier as the leader.
All nodes in the network follow this process and a unique leader is elected in a single connected component.
End *//end of while loop*
End *//end of case 2*

- ***Case 3*** : *If a new node i joins the network in between the rounds say round r then the neighbors will update its idlist.*

Begin
∀ Neighbors of i say nodej is the neighbor for $node_i$.
add (i, idlist(j));
The normal algorithm continues (the ids are propagated), nodes keep exchanging the information till diameter rounds.

while (round > = diameter),

```
Begin
    ∀ nodes in the network. Say nodej, and nodej receives
    an identifier i at diameter round.
    If i is greater than the maximum identifier nodej has
    propagated in the previous round (diameter 1).
            Propagate nodei to all the neighbors of j.
            Also propagate the nodei information to all the
            neighbors of neighbors of i until the whole net-
            work is covered, if the above condition sa-
            tisfies.
    Else Do not propagate the information.
    ∀ nodes in the network    // Termination Condition
    say for nodei,
    compare all the identifiers present in idlist(i)
    If all the identifiers in idlist(i) are equal, nodei stops prop-
    agating its maximum identifier and elects the maximum
    identifier as the leader.
    All nodes in the network follow this process and a unique
    leader is elected in a single connected component.
End         //end of the while loop
End         //end of case 3
```

So the time taken for the algorithm to elect a leader will be O (diam + Δt) where Δt is the time taken for all the nodes to converge and Δt depends on the topology changes in the network (Δt might be the time taken for few more rounds). The algorithm terminates when all nodes have exactly one identifier as a leader.

The message complexity for the leader election algorithm depends on the network topology and the number of messages propagated as the topology changes. Even for multiple concurrent changes in the network, our algorithm ensures only one leader but the time Δt may increase. Since all the cases are considered in the algorithm, even if multiple topological changes occur the algorithm can still elect a unique leader as different cases are called for different changes in the network at different times.

3. Handling Concurrent Multiple Topology Changes

When multiple topology changes occur, the algorithm should still be in a position to elect a single leader. For example, when the new leader information is to be propagated in the network, a topology change, like a link failure may occur and the required information may not be able to propagate through out the network. The algorithm should elect a single leader even in such scenarios. We consider different cases.

- **Case 1** : When two components merge together, the leader with the largest *id* is elected as new leader and that information is propagated in the component with the defeated leader *id*. During the propagation, if another new component comes into the network or a partition occurs in the network then the timestamps are compared and the id with the highest timestamp is propagated in the whole component.

```
Let C1 and C2 be with leaders lid1 and lid2.
    if (lid1 > lid2) then propagate (lid1, C2, t)
                                    // propagated at time t
    else propagate (lid2, C1, t)
    During this propagation if a new node comes into the network
    or a network partition occurs, then the new leader id in-
    formation will be propagated with a different time stamp. Now
    the time stamps are compared and the one with the most recent
    time stamp is propagated through out the network and is
    elected as the leader.
```

- **Case 2** : If a partition occurs in the network, the node that detects the partition will be elected as a leader and this information is propagated through out the component. The timestamp is also included in the propagated message so that if a new component comes in, the *lids* can be compared based on the timestamps. Algorithm for the propagate function is given below.

```
Propagate (lid, comp, t)
    int oid ;   //oid is the old leader id of the nodes in the
                component.
    nid = lid ; // nid is the new leader id that needs to be propa-
                gated
    if (oid < > nid) then
            broadcast nid to all the children (all the nodes that are
            at distance of one hop) of that node. Let this set of nodes
            be k
    else    do not broadcast
    For each k repeat the if loop. During the propagation, if some
    other process like partition or failure of a link happens. Then
    let the time stamps of the two processes be t1, t2.
For a node
    if (t1 > t2) then accept the id at t1, broadcast it and delete the
    id at t2
    else accept the id at t2, broadcast it and delete the id at t1
```

The timestamp in case of partition is the time that has been assigned when the node first detects the partition. When two components merge together, the node that decides to propagate the new *id* sets the timestamp and this timestamp should be the same through out the propagation.

The above algorithm can handle concurrent changes. Since we are assuming only finite number of topological changes to occur, the algorithm would definitely converge and only one leader gets elected in a single connected component.

4. Simulation Model and Performance Evaluation

We studied the performance of our algorithm with NS2 simulator [15]. We use a simple network with the mobile nodes and use the propagation models to check if the packets are received successfully. *Free space* model is used for short distances and *Tworayground* model is used for long distances. These models predict the received signal power of each packet. At the physical layer receiving threshold is defined. When a packet is received and if its receiving signal power is less than the receiving threshold, then the packet is dropped by the MAC layer. IEEE 802.11 MAC is implemented in NS which handles collision detection, fragmentation and acknowledgements. 802.11 is a CSMA protocol which checks for the availability of channel before sending data on to that channel. If the channel is not available, it waits for a random amount of time and then transmits the data.

Channel implementation is based on the shared media model. Every mobile node has different interfaces for connecting to the channel. Each channel has a particular frequency with a particular modulation scheme. A packet is received if the transmission range falls within the radio propagation model.

The shared media parameters need to be set to make it work like the 914MHz Lucent WaveLAN DSSS radio interface. <Table 1> lists the input parameters for an ad hoc network. The parameters specified in <Table 1> are used to establish communication between the mobile nodes, which include the capture threshold (cpthresh), carrier sense threshold (csthresh), receive power threshold (rxthresh), bandwidth (rb), transmission power (pt) and frequency (freq). The propagation model and Pt determines the received signal power of each packet. The packet cannot be correctly received if received power is below *rxthresh*. We measure the time for leader election, which is the output parameter in our simulation.

<Table 1> Simulation parameters and their values

Simulation System Parameters	Values
Propagation model	Tworayground
MAC	802_11
Ad hoc routing protocol	ZRP
Topography dimensions	670 × 670
Capture threshold (CPThresh_)(db)	10.0
Carrier sense threshold (CSThresh_)(db)	1.559e-11
Receive power threshold (RXThresh_)(db)	3.652e-10
Bandwidth (Rb_)	2 * 1e6
Transmission power(Pt_)	0.2818
Frequency(freq_)	914e + 6

- **Node movement and traffic** : Every node can be moved from one place to other by specifying the destination with (x, y). The *setdest* command is used to move the mobile nodes. Also the traffic can be sent between the mobile nodes i.e. both tcp and udp traffic can be sent.
- **Scalability issue** : We have evaluated the performance of the algorithm by scaling the network size i.e. by increasing the number of nodes. We considered networks with different topologies and diameters. We have started with 2 nodes and then increased the size to 10 nodes. The topological changes are made while the algorithm is executing in between rounds. (Figure 3) shows that the leader has been elected in each topology according to the algorithm specified. The different topologies considered are a 3-node network with diameter 2, 4-node network with diameter 2, 5-node network with diameter 3, 6-node network with diameter 4, 7-node network with diameter 4, 8-node network with diameter 5, 9 and 10-node network with diameters 5. The network topologies are taken in such a way that as the network size increases, the nodes come together and the diameter is reduced. We have implemented all the cases described in the algorithm. The time for election does vary as the time entirely depends on the topology and the type of topology that has occurred.

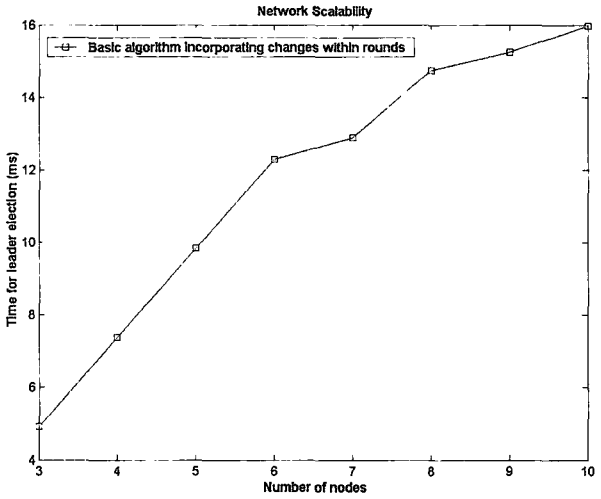
(Figure 1) shows that the leader gets elected in every network under consideration and as the network size increases, the leader election time is reduced because as the network size increases, the mobile nodes come physically close and the network diameter reduces. Since the time taken according to the algorithm is $O(diam + \Delta t)$, time is directly proportional to the diameter. We notice that for 7-node network, the graph deviates from the linear fashion because its diameter is 4, which is same as that for 6-node network. The same situation can be seen with 9 and 10-node networks. We have considered a single topology change after which the network gets stabilized and have incorporated the changes at different rounds in different topologies.

Consider the scenario where the leader has been already elected. We discuss three cases :

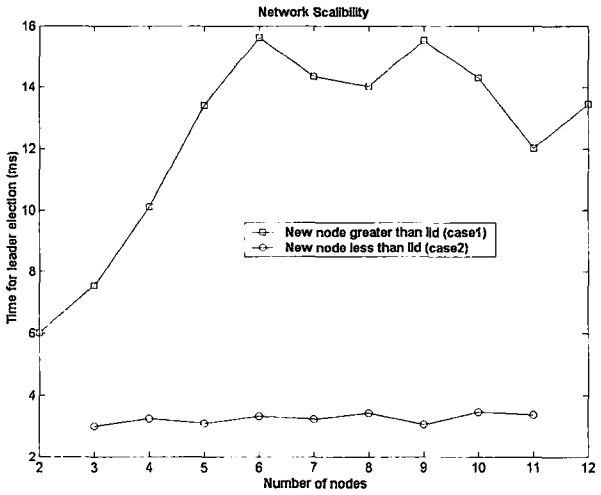
- **Case 1** : A new node comes into the network with an id greater than the current leader of the network. In such a scenario, the new id should be elected as a leader and this information needs to be propagated through out the

network.

- **Case 2** : If the new node id is smaller than the leader in the network, then nothing has to be done.
- **Case 3** : If a node leaves the network, two cases arise : (a) the node that leaves is a leader and (b) the moved node is not a leader. If the node that has moved is not a leader, then no change has to be done.



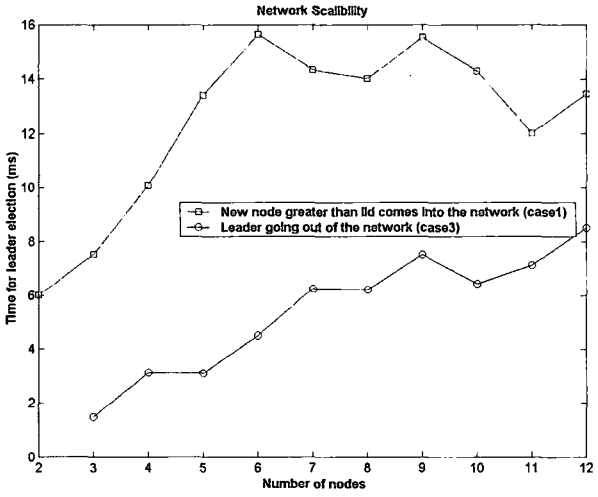
(Figure 1) Time for the algorithm



(Figure 2) Time comparison between case 1 and case 2

(Figure 2) shows the time comparison between case 1 and case 2. Case1 has few spikes, which are due to the topology of the network. As the network size increases, the nodes tend to get closer to each other so the maximum distance that the leader needs to be propagated is greatly reduced, which reduces time. For case 2, only single message has to be sent from the neighbor with which it meets first. So time taken for the whole network is very small. In both cases, only one leader is elected after the propagation is done.

(Figure 3) shows the time comparison between case 1 and case 3. Case 1 takes more time than case 3, and the spikes seen in the graphs correspond to the increase in network size. The reason for case 1 taking more time is that when a network of 3 nodes is considered, it becomes a network of 4 nodes in case 1 and network of 2 nodes in case 3. In case 1, the spikes are due to the increase in network size because of which the time to elect a leader gets reduced. In case 3, the network increase in the size but when compared with the previous network size the time taken will be reduced in some cases where the nodes come more close to each other. Finally from the graphs it can be seen that in each case a unique leader has been identified and elected in all cases.



(Figure 3) Time comparison between case 1 and case 3

5. Conclusions

In this paper an efficient leader election algorithm for mobile ad hoc networks is introduced along with the result of performance evaluation. The algorithm is easy to implement as the messages are exchanged only between the neighbors. The algorithm has used the concept of time stamping to distinguish the messages so that the most recent information is taken into consideration. All cases are considered in the algorithms and show that only one leader is elected at a particular time.

Also, we developed a simulation model for performance evaluation of the algorithm. The simulation result shows that the leader gets elected in every network under consid-

eration and as the network size increases, the leader election time is reduced because as the network size increases then the mobile nodes come physically close and the network diameter reduces.

References

[1] B. Liang and Z. J. Haas, "Virtual Backbone Generation and Maintenance in Ad Hoc Network Mobility Management," *IEEE INFOCOM'2000*, Tel Aviv, Israel, March 26-30, 2000.

[2] Kostas P Hatzis, George P. Pentaris, Paul G. Spirakis, Vasilis B. Tampakas and Richard B. Tan Fundamental Control Algorithms in Mobile Networks Proc. 11th Annual ACM Symp. on Parallel Algorithms And Architectures.

[3] N. Malpani, J. L. Welch. N. H. Vaidya, Leader Election Algorithms for Mobile Ad Hoc Networks, Fourth International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, Boston, August, 2000.

[4] Elizabeth M. Royer and Charles E. Perkins, "Multicast Operation of the Ad-hoc On-Demand Distance Vector Routing Protocol," *Proceedings of MobiCom '99*, Seattle, WA, pp.207-218, August, 1999.

[5] Chunhsiang Cheng and Srikanta P. R. Kumar, A loopfree spanning tree protocol in Dynamic topology, Proc. 27th Annual Allerton Conference on Communication, Control and Computing, pp.594-595, Sep., 1989.

[6] Vincent D. Park and M. Scott Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks, Proc. IEEE INFOCOM, April 7-11, 1997.

[7] A. Acharya, B. R. Badrinath, T. Imielinski, Impact of mobility on distributed computations, Operating Systems Review, April, 1993.

[8] A. Acharya, B. R. Badrinath, T. Imielinski, Structuring distributed algorithms for mobile hosts, In Proc. Of 14th International Conference on distributed computing systems, June, 1994.

[9] Z. J. Haas and S. Tabrizi, "On Some Challenges and Design Choices in Ad-Hoc Communications," *IEEE MILCOM'98*, Bedford, MA, October 18-21, 1998.

[10] A. Tsirigos and Z. J. Haas, "Multipath Routing in Mobile Ad Hoc Networks or How to Route in the Presence of Topological Changes," IEEE MILCOM'2001, Tysons Corner, VA, October 28-31, 2001.

[11] J. Haas and B. Liang, "Ad-Hoc Mobility Management with Randomized Database Groups," *IEEE ICC'99*, Vancouver, BC, Canada, June 6-10, 1999.

[12] Z. J. Haas and M. R. Pearlman, "The Performance of Query Control Schemes for the Zone Routing Protocol," *ACM DialM 1999*, Seattle, WA, August 20, 1999.

[13] Z. J. Haas and M. R. Pearlman, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks," Internet Draft, draft-ietf-manet-zone-zrp-02.txt, June, 1999

[14] G. LeLann, "Distributed Systems : Towards a formal approach," In Proc. Information Processing '77, B. Gilchrist (ed.), North-Holland, 1977.

[15] Kevin Fall, Kannan Varadhan. NS user manual, "The VINT project," A Collaboration between researches at UC Berkeley, LBL, USC/ISI, and Xerox PARC.

[16] S. Vasudevan, B. DeCleene, N. Immerman, J. Kurose and D. Towsley, ***Leader Election Algorithms for Wireless Ad Hoc Networks***, In Proceedings of IEEE DARPA Information Survivability Conference and Exposition (DISCEX), 2003.

[17] S. Vasudevan, J. Kurose and D. Towsley, ***Design and Analysis of a Leader Election Algorithm for Mobile, Ad Hoc Networks***. Accepted to appear in IEEE International Conference on Network Protocols (ICNP) 2004.

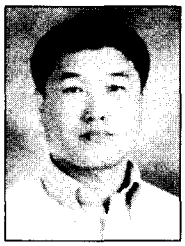
Pradeep Parvathipuram

e-mail : pkp39c@umkc.edu

2003년 University Of Missouri-Kansas City(MS)

2003년~현재 Sprint Corporation, USA, system developer

관심분야 : 컴퓨터 네트워크, 분산 시스템



양 기 철

e-mail : gcyang@mokpo.ac.kr

1982년 전남대학교 계산통계학과(학사)

1986년 University of Iowa 전산학(석사)

1993년 University of Missouri 전산학 (박사)

2001년 영국 Heriot-Watt University 객원교수

1993년~현재 목포대학교 정보공학부 교수

관심분야 : 인공지능, 자연어처리, 시맨틱 웹, 분산시스템 등

오 수 열

e-mail : syoh@mokpo.ac.kr

1981년 전남대학교 재료공학과(학사)

1986년 조선대학교 대학원 전산학과 (공학석사)

2004년 전남대학교 대학원 전산통계학과 (박사수료)

1988년~현재 목포대학교 정보공학부 컴퓨터공학전공 교수

관심분야 : 객체지향 시스템, 소프트웨어 품질 관리, 웹서비스 등