A Hierarchical Leader Election Protocol for Mobile Ad Hoc Networks

Orhan Dagdeviren[1] and Kayhan Erciyes[2]

[1] Izmir Institute of Technology Computer Eng. Dept.
Urla, Izmir TR-35340, Turkey
orhandagdeviren@iyte.edu.tr
[2] Ege University International Computer Institute
Bornova, Izmir, TR-35100, Turkey
kayhan.erciyes@ege.edu.tr

Abstract. Leader Election is an important problem in mobile ad hoc networks and in distributed computing systems. In this study, we propose a hierarchical, cluster based protocol to elect a leader in a mobile ad hoc network. The initial phase of the protocol employs a clustering algorithm to group nodes of the network after which a leader for a cluster(clusterhead) is elected. The second phase is performed by forming a connected ring of these leaders using the *Ring Formation Algorithm*. Finally, *Chang Roberts Leader Election Algorithm* for rings is employed in the final phase to elect the super-leader among the clusterheads. We provide performance results of this protocol for various mobility parameters and analyze its time and message complexities.

Keywords: leader election, Chang Roberts algorithm, mobile ad hoc networks.

1 Introduction

Leader election is a fundamental problem addressed by many researchers in distributed systems. This problem in distributed systems is first introduced by LeLann who also proposes a solution for this problem on a unidirectional ring [1]. Chang and Roberts improve this solution and reduces the average messaging complexity [2]. Also, for bidirectional rings and arbitrary networks, various solutions are proposed [3,4,5,6].

Mobile Ad hoc NETworks (MANETs) are a class of distributed networks which do not have a fixed topology and where the nodes communicate using temporary connections with their neighbors. Leader election in MANETs is a relatively new research area. Malpani et. al. [7] propose two leader election protocols based on TORA. The algorithms select a leader for each connected component. The first algorithm is designed to tolerate single topology change whereas the second algorithm tolerates concurrent topology changes. The nodes only exchange messages with their neighbors, thus making this protocol suitable for MANETs. The authors show the proof of correctness but they do not

M. Bubak et al. (Eds.): ICCS 2008, Part I, LNCS 5101, pp. 509–518, 2008.

give any simulation results. Vasudevan et. al. [8] propose a weakly-self stabilizing and terminating leader election protocol for MANETs. Their algorithm uses the concept of diffusing computations. They show the proof of correctness by using temporal logic but they also do not give any simulation results. Pradeep et. al. [9] propose a leader election algorithm similar to Malpani et. al.'s algorithm. They use the Zone Routing Protocol and show the proof of correctness of their algorithm. Masum et. al. [10] propose a consensus based asynchronous leader election algorithm for MANETs. The authors claim that their algorithm is adaptive to link failures. All of these algorithms elect leader(s) from ordinary nodes. On the other hand, our algorithm elects one super-leader from previously selected leaders. Cokuslu and Erciyes [11] proposed a two level leader election hierarchy for MANETs. Their protocol is based on constructing dominating sets for clustering and electing the super clusterhead from the subset of connected clusterheads. Since the clusterheads must be connected, the super clusterhead selection protocol is restricted by the underlying protocol. Also dominating set construction is an expensive operation under high mobility. Our Mobile_CR protocol can use any clustering and routing protocol under BFA to be stable under high mobility and density.

In this study, we propose a Leader Election Protocol (LEP) that has three layers (phases) for MANETs. At the lowest layer, a clustering algorithm divides the MANET into balanced clusters, using the previously designed Merging Clustering Algorithm (MCA) [12]. The second layer employs the Backbone Formation Algorithm (BFA) which provides a virtual ring architecture of the leaders of the clusters formed by MCA [12]. Finally, using the Mobile Chang Roberts Leader Election Algorithm ($Mobile\_CR$), the super-leader among the leaders elected in the second phase is elected. We show experimentally and theoretically that the protocol is scalable and has favorable performance with respect to time and message complexities. The rest of the paper is organized as follows. Section 2 provides the background and the proposed architecture is outlined in Section 3. Section 4 describes the extended Chang Roberts algorithm on the proposed model called *Mobile_CR*. The implementation results are explained in Section 5 and the discussions and the conclusions are outlined in Section 6.

2 Background

In this section we explain the clustering using MCA and backbone formation using BFA to show the underlying mechanism for LEP.

2.1 Clustering Using Merging Clustering Algorithm

An undirected graph is defined as $G = (V, E)$, where V is a finite nonempty set and $E \subseteq V \times V$. The V is a set of nodes v and the E is a set of edges e. A graph $G_S = (V_S, E_S)$ is a spanning subgraph of $G = (V, E)$ if $V_S = V$. A spanning tree of a graph is an undirected connected acyclic spanning subgraph. Intuitively, a minimum spanning tree (MST) for a graph is a subgraph that has the minimum

number of edges for maintaining connectivity. Merging Clustering Algorithm MCA [12] finds clusters in a MANET by merging the clusters to form higher level clusters as mentioned in Gallagher, Humblet, Spira's algorithm [6]. However, we focus on the clustering operation by discarding minimum spanning tree. This reduces the message complexity as explained in [12]. The second contribution is to use upper and lower bound parameters for clustering operation which results in balanced number of nodes in the cluster formed. The protocol is simulated in $ns2$ and stable results under varying density and mobility are shown.

2.2 Backbone Formation Algorithm

Backbone Formation Algorithm constructs a backbone architecture on a clustered MANET [13]. Different than other algorithms, the backbone is constructed as a directed ring architecture to gain the advantage of this topology and to give better services for other middleware protocols. The second contribution is to connect the clusterheads of a balanced clustering scheme which completes two essential needs of clustering by having balanced clusters and minimized routing delay. Besides, the backbone formation algorithm is fault tolerant as the third contribution. Our main idea is to maintain a directed ring architecture by constructing a minimum spanning tree between clusterheads and classifying clusterheads into *BACKBONE* or *LEAF* nodes, periodically. To maintain these structures, each clusterhead broadcasts a *Leader_Info* message by flooding. In this phase, cluster member nodes act as routers to transmit *Leader_Info* messages. Algorithm has two modes of operation; hop-based backbone formation scheme and position-based backbone formation scheme. In hop-based backbone formation scheme, minimum number of hops between clusterheads are taken into consideration in the minimum spanning tree construction. Minimum hop counts can be obtained during flooding scheme. For highly mobile scenarios, an agreement between clusterheads must be maintained to guarantee the consistent hop information. In position-based backbone formation scheme, positions of clusterheads are used to construct the minimum spanning tree. If each node knows its velocity and the direction of velocity, these information can be appended with a timestamp to the *Leader_Info* message to construct better minimum spanning tree. But in this mode, nodes must be equipped with a position tracker like a GPS receiver. Backbone formation algorithm is implemented on top of MCA using the $ns2$ simulator. The results with varying MANET conditions are shown to be stable [13].

3 The Proposed Architecture

We propose a four layer architecture for MANETs as shown in Fig. 1. Implementations of other higher level functions on top of the lower three layers are possible. The lowest layer is the routing layer in which AODV [14] is used and other routing protocols can also be used. AODV is chosen since it is a widely used routing protocol which also has a stable $ns2$ release. The second layer is where

the clustering takes place at the end of which, balanced clusters are formed. Our clustering layer provides that nodes in vicinity of each other join to the same cluster to reduce routing overhead. The third layer inputs these clusters and form a virtual ring of the leaders of these clusters. Finally, the fourth layer shows the implementation of the Mobile_CR Algorithm on top of these three layers.

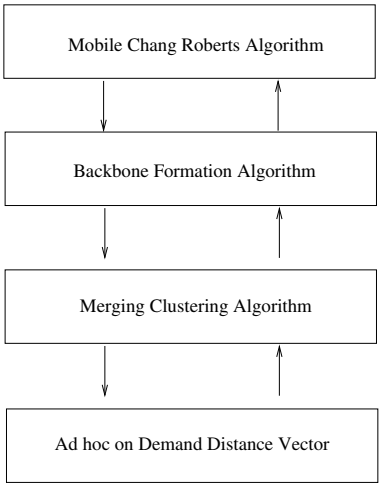

Fig. 1. The Proposed Architecture

4 Mobile Chang-Roberts Algorithm

Chang Roberts Algorithm is an asynchronous leader election algorithm for unidirectional ring networks. Assuming each process can be either *red* meaning a potential candidate for becoming a leader or *black* meaning a resigned state, an informal description of the algorithm is as follows. Any red process can initiate the algorithm, however, if a red process receives a token before initiating the algorithm, it resigns by turning black [15]. Non-initiators remain black, and act as routers. A process that receives a token with a smaller id than itself, removes the token. A higher id token is forwarded to the next node and if it reaches the originator, it has a higher id than all and the originator can then declare itself as the leader.

1. Initially all initiator processes are red.
2. For each initiator i, token $< i >$ is sent to its neighbor;
3. **do** (for every process i)
4. token $< j > \wedge\ j > i \rightarrow$ skip;
5. token $< j > \wedge\ j < i \rightarrow$ send token $< j >$; $color := black$ (i resigns)
6. token $< j > \wedge\ j = i \rightarrow L(i) := i$ (i becomes the leader)
7. **od**

8. (for a non-initiator process)
9. **do** token $< j >$ received $\rightarrow$ *color* := *black*; send $< j >$ **od**

This algorithm ensures that the lowest id of the initiators wins and becomes the leader and its complexity is $O(n^2)$. We provide the following improvement to the classical Mobile_CR Algorithm. Every node keeps a list of the identities of the nodes that it has seen so far by the tokens it has received. It then only passes tokens that has a smaller sender id than the ones in the list, rather than checking the id present in the incoming token with itself only.

4.1 Finite State Machine Diagram of the Mobile_CR Algorithm

The Mobile_CR Algorithm is described using a finite state machine diagram to capture all of the possible asynchronous activities. Firstly, the list of messages used in Mobile_CR Algorithm is specified as follows :

- $LEADER\_DEAD$: It is triggered by an internal event which detects super leader's crash.
- $TOKEN$: Sent or forwarded by a leader to its next leader for election.
- $LEADER$: Sent by the leader which is the winner of the election.

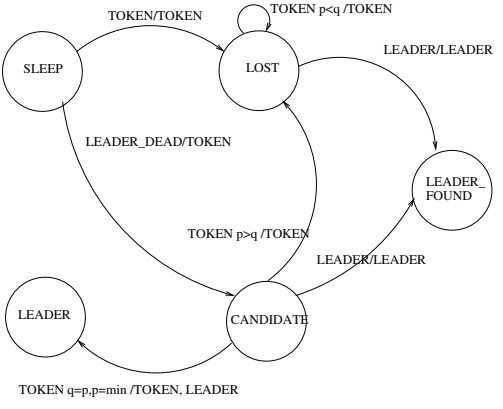

Fig. 2. FSM of the Mobile_CR Leader

The following is the list of node states:

- $SLEEP$: I am in idle state.
- $LOST$: I am not a CANDIDATE as there is someone with higher ID than me.
- $CANDIDATE$: I am CANDIDATE to become a LEADER and I am in election.
- $LEADER\_FOUND$: LEADER is determined and I know who it is.
- $LEADER$: I am the LEADER.

4.2 An Example Operation

Fig. 3 shows an example scenario of the Mobile_CR Algorithm for a network with 40 nodes located on top of a 600m^2 surface area. The x,y coordinate of each node is given next to it. Node 37's coverage area is shown by the dotted circle. As shown by the bold lines, the network is partitioned into 5 clusters with MCA. After partitioning the network, the backbone is constructed with BFA as a directed ring architecture which is depicted with the arcs and its arrows in Fig 3. Initially all leader nodes are at the $SLEEP$ state. Within a small amount of

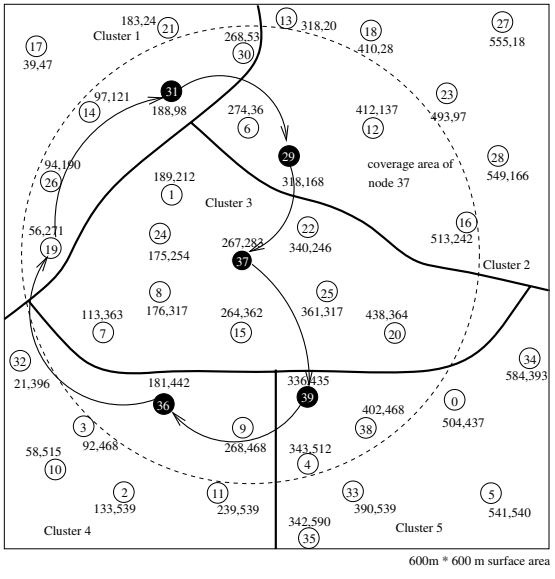

Fig. 3. An Example Operation for Mobile_CR

time, node 31 becomes an initiator and changes its state to the $CANDIDATE$ by sending a $TOKEN$ message to its next leader on the ring, node 29. Node 29, which is in $SLEEP$ state receives the $TOKEN$ message and changes its state to the $LOST$. Node 29 sends an acknowledgement message to node 31. Each protocol message is acknowledged to maintain reliable transmission. Node 29 forwards $TOKEN$ message to the Node 37. At the same time, node 39 becomes an initiator and also makes a state transition from $SLEEP$ to $CANDIDATE$ state by sending a token message to its next leader on the ring, node 36. Node 37 forwards the $TOKEN$ of node 31 to the node 39. Node 39 loses the election, since id of received token is smaller. However node 39's token will be forwarded to the node 31. Node 31 blocks the $TOKEN$ message of the node 39. In the end, the $TOKEN$ message of node 31 circulates the ring and node 31 becomes the $LEADER$ of leaders. Node 31 sends a $LEADER$ message to the next leader on the ring which then circulates the ring to indicate the new leader of leaders.

4.3 Analysis

The proposed protocol consists of three layers as shown in Fig. 1.

1. Merging Clustering Algorithm MCA
2. Backbone Ring Formation Algorithm BFA
3. Mobile Chang Roberts Algorithm $Mobile\_CR$

Theorem 1. *The message and time complexity of the protocol is* $O(kn)$ *where* k *is the number of clusters.*

Proof. The message complexity for the protocol is the sum of the message complexities of the above three algorithms plus the messages required for termination detection of the first two algorithms. Assuming termination detection requires negligible number of messages, the message complexity of the Leader Election Protocol(LEP) is :

$$O(LEP) = O(MCA) + O(BFA) + O(Mobile\_CR) \tag{1}$$

$$O(LEP) = O(n) + O(kn) + O(k^2) \tag{2}$$

$$O(LEP) = O(kn) \tag{3}$$

By using the same method time complexity can be found as:

$$O(LEP) = O(n) + O(kn) + O(n) \tag{4}$$

$$O(LEP) = O(kn) \tag{5}$$

5 Results

We implemented the protocol stack with the $ns2$ simulator. Different size of flat surfaces are chosen for each simulation to create medium , dense and highly dense connected networks. Medium, small and very small surfaces vary between 140m × 700m to 700m × 700m, 130m × 650m to 650m × 700m, 120m × 600m to 600m × 600m respectively. Average degree of the network is approximately $N/4$ for the medium connected, $N/3.5$ for the dense connected and $N/3$ for the highly dense connected networks where N denotes the total number of nodes in the network. Although each packet is acknowledged by the destination to maintain reliable transmission and retransmitted if dropped, sparse networks are not studied because of the lack of connectivity. N varies from 10 to 50 in our experiments. Random movements are generated for each simulation and random waypoint model is chosen as the mobility pattern. Low, medium and high mobility scenarios are generated and respective node speeds are limited

between 1.0m/s to 5.0m/s, 5.0m/s to 10.0m/s, 10.0m/s to 20.0m/s. We use the codes of MCA and BFA previously simulated by us under Mobile_CR to obtain end-to-end measurements. Each measurement is taken as the average value of 20 measurements with the same mobility and density pattern but different randomly generated node locations and speeds. Our previous study shows that MCA and BFA is stable under different density and mobility conditions. Fig. 4 and Fig. 5 show that election time increases linearly with total number of nodes and Mobile_CR is stable under density and mobility changes. The runtimes decrease as mobility increases as shown in Fig. 5 since the number of clusterheads forming the ring are smaller for high mobility scenarios resulting in less network traffic.

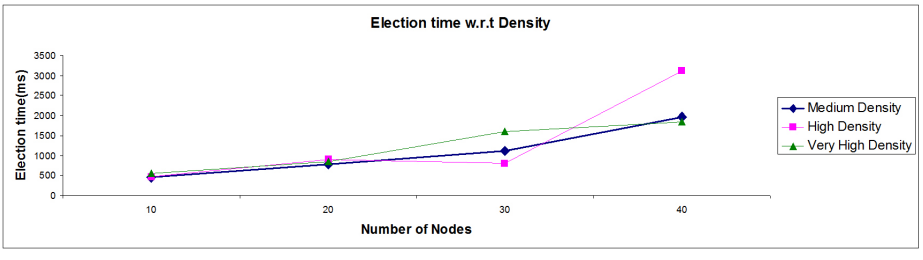

Fig. 4. Election Time against Density for Mobile_CR

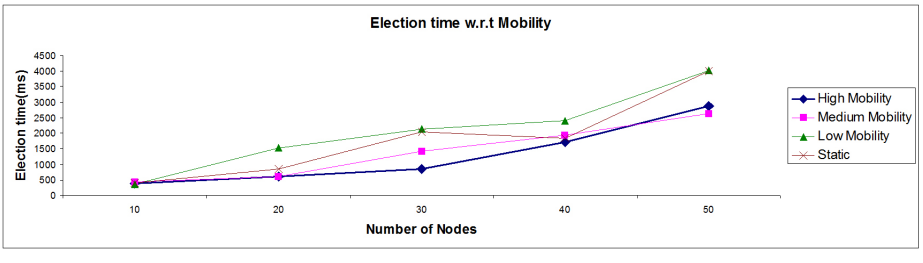

Fig. 5. Election Time against Mobility for Mobile_CR

Number of nodes on the ring formed by BFA is an important parameter for election time in Mobile_CR. Upper and lower bound cluster parameters are defined in MCA to adjust cluster sizes. We use this parameter to divide the network into a number of clusters. Network is divided into 3, 4, 5, and 7 clusters and the effect of the number of clusterheads on the ring is measured. Fig. 6 shows that the election time slightly changes with the total number of clusters in the network. One can expect a linear increase of the election time with the total number of clusters, but the number of actively routing nodes are selected by AODV and not only clusterheads are used for routing.

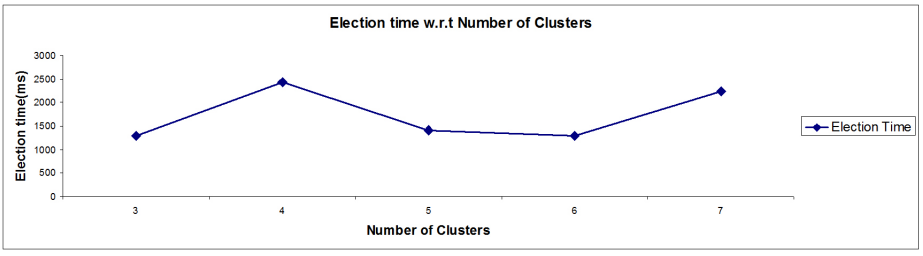

Fig. 6. Election Time against Number of Clusters for Mobile_CR

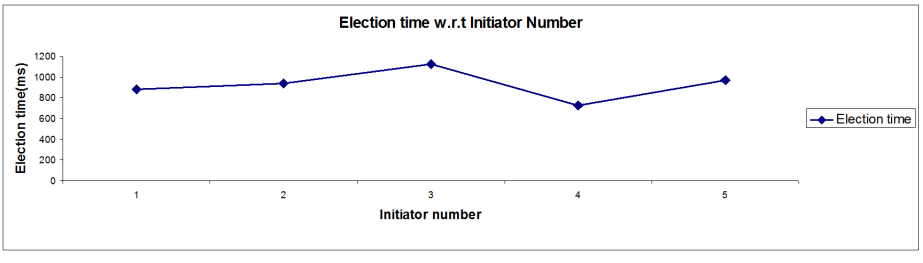

Fig. 7. Election Time against Initiator number for Mobile_CR

Lastly, we investigate the behaviour of the election time when nodes start concurrently. In our algorithm, each node stores a list of the received *id* of initiators and blocks the tokens of the candidates having greater *id* than leader and 1 to 5 initiators are selected for simulations. The results in Fig. 7 shows that the algorithm is stable against varying number of concurrent initiators.

6 Conclusions

We provided a three layer architecture for the dynamic leader election problem in MANETs where the clustering phase provided the leaders for each cluster in the first phase. The backbone formation algorithm provided a ring network among the local leaders in the second phase and a super-leader among the local leaders is elected using the Mobile_CR algorithm in the final phase. We showed experimentally and theoretically that this approach is scalable and has an overall favorable performance. We think this approach may find various implementation environments in MANETs where sub activities within the MANET are handled by group/cluster of nodes each having a leader for local decisions but overall control is achieved by a single super-leader node. The protocol can be invoked periodically which ensures correct handling of failing leaders.

References

1. LeLann, G.: Distributed Systems: Towards a Formal Approach. IEEE Information Processing 77, 155–169 (1977)
2. Chang, E.J., Roberts, R.: An Improved Algorithm for Decentralized Extrema Finding in Circular Arrangements of Processes. ACM Com., 281–283 (1979)
3. Franklin, W.R.: On an Improved Algorithm for Decentralized Extrema Finding in Circular Configurations of Processors. ACM Com., 281–283 (1982)
4. Peterson, G.L.: An O(nlogn) Unidirectional Algorithm for the Circular Extrema Problem. ACM Trans. Prog. Lang. 4, 758–763 (1982)
5. Dolev, D., Klawe, M., Rodeh, M.: An O(nlogn) Unidirectional Distributed Algorithm for Extrema-Finding in a Circle. J. Algorithms 3, 245–260 (1982)
6. Gallagher, R.G., Humblet, P.A., Spira, P.M.: A Distributed Algorithm for Minimum-Weight Spanning Trees. ACM Trans. Prog. Lang., 66–77 (1983)
7. Malpani, N., Welch, J., Vaidya, N.: Leader Election Algorithms for Mobile Ad Hoc Networks. In: Proc. Int. Works. on Disc. Alg. and Meth., pp. 96–103 (2000)
8. Vasudevan, S., Immerman, N., Kurose, J., Towsley, D.: A Leader Election Algorithm for Mobile Ad Hoc Network. UMass Comp. Sci. Tech. Rep. (2003)
9. Pradeep, P., Kumar, V., Yang, G.-C., Ghosh, R.K., Mohanty, H.: An Efficient Leader Election Algorithm for Mobile Ad Hoc Networks. In: Ghosh, R.K., Mohanty, H. (eds.) ICDCIT 2004. LNCS, vol. 3347, pp. 32–41. Springer, Heidelberg (2004)
10. Masum, S.M., Ali, A.A., Bhuiyan, M.T.I.: Asynchronous Leader Election in Mobile Ad Hoc Networks. In: AINA 2006, pp. 827–831 (2006)
11. Cokuslu, D., Erciyes, K.: A Hierarchical Connected Dominating Set Based Clustering Algorithm for Mobile Ad Hoc Networks. In: IEEE MASCOTS 2007, pp. 60–66 (2007)
12. Dagdeviren, O., Erciyes, K., Cokuslu, D.: Merging Clustering Algorithm for Mobile Ad Hoc Networks. In: Gavrilova, M.L., Gervasi, O., Kumar, V., Tan, C.J.K., Taniar, D., Laganá, A., Mun, Y., Choo, H. (eds.) ICCSA 2006. LNCS, vol. 3981, pp. 681–690. Springer, Heidelberg (2006)
13. Dagdeviren, O., Erciyes, K.: A Distributed Backbone Formation Algorithm for Mobile Ad Hoc Networks. In: Guo, M., Yang, L.T., Di Martino, B., Zima, H.P., Dongarra, J., Tang, F. (eds.) ISPA 2006. LNCS, vol. 4330, pp. 219–230. Springer, Heidelberg (2006)
14. Perkins C. E., Belding-Royer E. M., Das S.: Ad Hoc On Demand Distance Vector (AODV) Routing. RFC 3561 (2003)
15. Ghosh, S.: Distributed Systems, An Algorithmic Approach, ch. 11, pp. 175–176. Chapman and Hall/CRC (2007)