Cooperative Leader Election Algorithm for Master/Slave Mobile Ad Hoc Networks

Redouane Ali, Suksant Sae Lor, Rabah Taleb Benouaer, Miguel Rio
University College London
{r.ali, s.lor, taleb, m.rio}@ee.ucl.ac.uk

***Abstract*—This paper proposes a novel and efficient cooperative leader election algorithm for master/slave mobile ad hoc networks. The algorithm relies on collecting and re-distributing information amongst local nodes in order to find the leader. It is based on the assumption that if this process is repeated sufficiently then the algorithm will converge towards a unique leader. It is shown that the proposed mechanism outperforms existing algorithms in terms of time complexity and response to node mobility. The algorithm was simulated for Bluetooth ad hoc networks, which, by default, rely on a master/slave architecture, however, the cooperative approach could be adapted to any network that exhibits the master/slave configuration such as clustered ad hoc networks or ZigBee-based sensor networks.**

I. Introduction

Ad hoc networking has emerged as an elegant approach to mobile computing. In these networks, mobile devices act both as hosts and relays, and do not rely on fixed underlying infrastructures. Typical usage scenarios of ad hoc networks include battlefields, building sites, personal area networks and sensor networks. It is common for mobile ad hoc networks to be organised in clusters or pico-networks for ease of routing and management [1],[2], in which, one of the nodes in the pico-network acts as the cluster head (master) with the remaining ones as cluster nodes (slaves). The master plays the role of coordinator in the pico-network; it could for instance, schedule transmissions, allocate subnet addresses or carry out clock synchronisation. Moreover, in low power ad hoc networks time-division medium access might be more appropriate as these are simple to implement while ensuring collision-free use of the shared radio medium [3]. A particular arrangement of master/slave networks is one wherein slave nodes can only communicate to each other via master nodes [1] hence creating a bipartite transmission network. This specific topology sets some constraints to the way algorithms operate but protocol design can take advantage of this particularity to devise algorithms that are specifically suited to work over this type of networks. Leader election is one example that can use the master/slave constraint to its advantage without the need for extra functionality.

The leader election problem is a well studied area in wired, distributed networks [4] but its application in wireless ad hoc networks is still in its early days. A network leader would, for instance, be used as a data sink, assign tasks to various nodes in the network or carry out security management functions. Depending on the function of a leader, it must be chosen according to criteria that maximise or facilitate its role. For example, in the case of task assignment function, centrality of a node might be of importance, whereas in the case of data analysis, processing power would be a selection criterion. Regardless of the role of the leader or the network in use, the leader election problem must satisfy two fundamental conditions:

1) There exists only one leader in a connected component.
2) Every node in the connected component has knowledge of the identity of the leader.

The remainder of the paper is organised as follows: section II discusses related work done on leader election for ad hoc networks; section III sets out assumptions and requirements for the system, section IV provides details of the cooperative leader election algorithm; section V presents simulation results and findings for different network sizes and settings; and section VI gives concluding remarks and outlines future work.

II. Related Work

A major concern of leader election in ad hoc networks is the dynamic and unpredictable nature of the environment. Several algorithms have attempted to tackle this issue, each with its benefits and drawbacks. Three of the predominant approaches are tree-based [5],[6], routing-based [7],[8] and probabilistic [9],[10] algorithms.

Tree based algorithms rely on building and maintaining a spanning tree rooted at the initiator that would eventually inform the other nodes of the identity of the leader. While this process ensures global knowledge of the leader identity, it has a relatively slow convergence time and incurs rather high cost in terms of memory and computational overheads. Moreover, tree-based algorithms do not cope well with high node mobility and require the network to be static for some period of time for the protocol to converge, which is not practical in ad hoc networks due to the unpredictability of node movements. The work presented in [5] proposes an algorithm based on diffusing computations, in which an initiator of a computation expands and shrinks a tree until all nodes in the network are reached. As a result, the leader will be known to every node in the network.

The work in [6] proposes a similar approach to that of [5] but improves on efficiency by reducing the number of simultaneous computations to pre-elected candidates within the network. The algorithm in [6] also suffers from large convergence times and poor response to high mobility. In

978-1-4244-5661-1/09/$26.00 ©2009 IEEE

[7], a routing-based leader election algorithm is proposed. It builds on the principles of the Temporally Ordered Routing Algorithm (TORA) wherein nodes adjust a locally maintained variable, called the *height*, to point to the leader, in a decrementing manner over a Directed Acyclic Graph (DAG). This approach responds better to mobility than do tree-based algorithms; however it comes at an extra cost, requiring the nodes to build and maintain routing information which results in large overhead. The work in [8] improves on the idea of [7] by devising a protocol that is self-stabilising with relatively fast convergence but also suffers from routing overhead and extra complexity in maintaining the DAG.

Probabilistic election algorithms assume that the underlying networks are unreliable where processes can become arbitrarily faulty, hence resulting in impossibility of consensus. To tackle this issue, [9] proposes a probabilistic leader election method which elects a unique leader with a given probability of success considering arbitrary changes in the states of processes. In [10], this problem is fixed using a fault detection algorithm, Ω, which distinguishes correct from faulty processes so that the latter are not considered at particular execution rounds. It is shown in [10] that fault detection methods are expensive and do not necessarily improve the performance of the election by much. It is therefore more desirable to design an algorithm that is simple enough yet with a high probability of success rather than implementing expensive and power hungry systems.

III. Assumptions and Requirements

This paper discusses only the leader election algorithm, *i.e.* any other simultaneous data transmissions are not considered. It is assumed that a network is a connected component where there exists a path between any node pair. A disjoint component in the network would run its own election algorithm although components might merge or disconnect. All connected nodes are assumed to be synchronised to a unique and shared clock. Nodes are mobile and are identified by a unique identifier; they have infinite queue size and are linked via perfect wireless links, *i.e.* it is assumed that packets are not corrupted or lost during transmission. Nodes are also assumed to have knowledge of the identity of their next hop neighbours as well as their own role in the network. The leader is elected based on some leadership capacity, which is pre-computed prior to the election process and has a numerical value stored at each node; when two nodes share the same value, ids are used to break ties. Any node can declare itself a candidate.

The following terms are used throughout this paper:

- **Piconet:** a set of nodes with at least 2 nodes and at most one master.
- **Master:** a node in the network that manages a piconet and can connect to up to 7 slaves.
- **Slave:** a node that can only communicate with a master node if it had exchanged information in the previous time-slot and cannot initiate a communication.
- **PMP (Participant in Multiple Piconets):** a slave node that bridges between two or more masters.

IV. Leader Election Algorithm

The leader election algorithm falls within the category of gossip algorithms, in that a gossip (the identity of the leader) will be disseminated to the entire network through local exchange of information. Each master gathers information from its currently connected neighbours, compares this information with its own, and distributes the best found value through broadcast. A fundamental question to be answered, therefore, is: "how many times must this process be repeated in order to elect a unique leader?". The answer lies with knowledge of connectivity characteristics of the underlying topology, namely the estimated number of PMP nodes in the network and their average degree. The estimator proposed in [11] is based on a random tour approach, which, primarily, has the function of estimating network sizes from information gathered along the walk. This is given by:

$$\hat{N} = \hat{M} \cdot (\overline{d_m} + 1) - \hat{P} \cdot (\overline{d_p} - 1) \tag{1}$$

where $\hat{N}$ is the estimated network size, $\hat{M}$ is the estimated number of masters, $\hat{P}$ represents the estimated number of PMP nodes, and $\overline{d_m}$ and $\overline{d_p}$ are estimated average degrees of the master and PMP nodes respectively. Using (1), the information required by the election algorithm ($\hat{P}$ and $\overline{d_p}$) can be derived.

Because Bluetooth relies explicitly on a master/slave architecture, the performance of the cooperative leader election algorithm was evaluated on such networks. In a Bluetooth multi-hop ad hoc network (also called a scatternet), a time-based scheduler ensures that a slave PMP maintains only one active connection with one of its neighbouring masters at any given time [12]. As a result, there are multiple independent network states, which, in a static setting, would cycle back to the initial state.

Bluetooth relies on a Time-Division Duplex (TDD) polling scheme where a slave node can only communicate with its master if it has received a message or was polled in the previous time-slot [13]. There are various scheduling mechanisms that could be used for this purpose. However, in order to keep operations simple and to ensure fairness, a Round-Robin scheduler is employed. In this scheduler, a master polls each slave in turn in a cyclic manner to exchange information with it in a dedicated time-slot. In some applications, Round-Robin is not the most efficient scheme as nodes might be polled without having anything to send [14]. However, it is perfectly adequate for the proposed leader election algorithm as it generates low traffic and requires continuous updates from all slaves [12]. Nonetheless, in scatternets, the scheduling becomes more complex and the mechanism must ensure that no transmission loops occur [15],[16]. The inter-piconet scheduling employed in this work is based on a leasing mechanism, in which, a master node reserves a time-slot for communication with a particular slave in the piconet. Consequently, this slave becomes unavailable to any other master during that time-slot and is only released upon completion of the update.

The proposed leader election algorithm makes use of this transmission mechanism in order to spread the identity of

TABLE I
NOTATIONS

M	master node.
S_M	set of next hop slave nodes to master M.
$S_c(M)$	slaves currently participating in M's piconet.
itr	number of iterations remaining.
S	slave node.
$L_{id}(M)$	id of the leader currently held by master M.
$L_{val}(M)$	value of the leader currently held by master M.
$L_{id}(S)$	id of the leader currently held by salve S.
$L_{val}(S)$	value of the leader currently held by slave S.
$POLL(M, S_M)$	polls slaves from set S_M for communication with master M during reserved time-slots.

$S_c(M) \leftarrow POLL(M, S_M)$
while $itr > 0$ **do**
  for all $S \in S_c(M)$ **do**
    if $L_{val}(S) > L_{val}(M)$ **and** $L_{id}(S) > L_{id}(M)$ **then**
      $L_{id}(M) \leftarrow L_{id}(S)$
      $L_{val}(M) \leftarrow L_{val}(S)$
    end if
  end for
  for all $S \in S_c(M)$ **do**
    $L_{id}(S) \leftarrow L_{id}(M)$
    $L_{val}(S) \leftarrow L_{val}(M)$
  end for
  $itr \leftarrow itr - 1$
end while

Fig. 1. Process run by each master at each iteration.

the unique leader in the connected component. The algorithm works as follows:

Every node in the network keeps a tuple, (L_{id}, L_{val}), that indicates the id and value of the leader known so far and initiates itself as a candidate. Initially, every node sets the leader id and value to its own. At each iteration, when a slave within a piconet is polled, it transmits the id and value of the leader it holds to its master; the master then compares the information it receives from the slave with the local one. If it learns of a better-valued leader, it updates its leader information (L_{id} and L_{val}) or discard it otherwise. After the completion of the Round-Robin tour, the master node broadcasts the information of the best leader in the piconet.

A PMP that was previously connected to a particular master would join another piconet at the subsequent iteration. If this PMP holds a better leader, it will update the master and corresponding slaves participating in the same piconet with the leader information, otherwise, it will update its own leader information. After going through this process several times, the unique identity of the leader will eventually be known to every node in the network. The process run by each master is illustrated in Figure 1.

Proposition 4.1: The number of rounds, itr, needed to elect a unique leader has an upper bound determined by:

$$itr \leq itr_{max} = 2 \times \left(\sum_{i \in P(G)} d(i) - P \right) + 1 \qquad (2)$$

where $P(G)$ is the set of PMP nodes in network G, P is the number of PMP nodes in G and $d(i)$ is the degree of PMP i.

itr_{max} denotes the maximum number of iterations that a master goes collecting and re-distributing leader information to satisfy the fundamental condition of the leader election problem.

Proof: Consider a star network with a single PMP node, p_1, and up to l masters $\{m_1, m_2, ...m_l\}$. The maximum number of iterations it takes for the PMP node to communicate with a particular master, considering a Round-Robin schedule, is d_{p_1}, where d_{p_1} is the degree of the PMP node. And the maximum number of rounds it takes for two master nodes to exchange information is $d_{p_1} + d_{p_1} - 1 = 2 \times d_{p_1} - 1$.

If we now have another star network, with single PMP node p_2 of degree d_{p_2} linked to the first network via a bridge master node then the maximum number of rounds it takes to convey information from one end to another is:

$$(2 \times d_{p_1} - 1) + \{(2 \times d_{p_2} - 1) - 1\} \qquad (3)$$

If n subnetworks are cascaded in a similar arrangement then the upper bound is given by:

$$\begin{aligned} itr_{max} = &(2 \times d_{p_1} - 1) + \{(2 \times d_{p_2} - 1) - 1\} + \cdots \\ &+ \{(2 \times d_{p_n} - 1) - 1\} \end{aligned} \qquad (4)$$

(4) becomes

$$itr_{max} = 2 \times \left(\sum_{i \in P(G)} d(i) - P \right) + 1 \qquad (5)$$

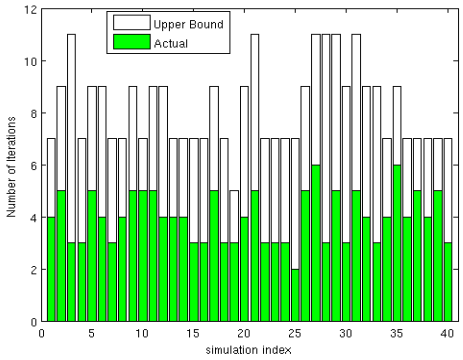
■

Equation (5) defines an upper bound for the number of rounds a master needs to exchange information with its neighbours, which can be approximated to:

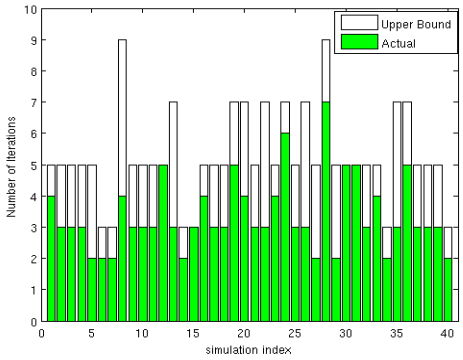
$$2 \times \hat{P}(\overline{d_p} - 1) + 1 \qquad (6)$$

where $\hat{P}$ is the number of estimated PMP nodes and $\overline{d_p}$ is the average degree of the estimated PMPs.

This upper bound is reached in two particular cases: when nodes are arranged in a star topology, as described above, and in a chain arrangement, wherein the gossip needs to traverse the entire network, from one end to the other. This is due to the fact that, under the worst case scenario, an information needs to go through every single node in the network for it to arrive at a particular destination. In realistic networks, nonetheless, this limit is rarely reached, because of the connectivity properties of nodes. Figure 2 shows a comparison between the actual number of rounds taken to elect a leader and the corresponding upper bound limit as expressed by equation (5), for two master/slave networks of 50 and 8 nodes respectively.



(a) 50 nodes



(b) 8 nodes

Fig. 2. Comparison of itr and itr_{max}.

It can be seen from Figure 2 that the number of iterations it takes to elect a leader, in realistic networks, rarely reaches the upper limit. This is certainly true for medium and large size networks. However, for smaller networks, in which nodes are more likely to be arranged in a star or chain topology, the upper limit has higher chances to be attained, although these cases still remain relatively infrequent (as shown in Figure 2(b)). This proves that chain or star topologies (or their combination) present worst case arrangements in terms of the number of rounds.

Dealing With Node Mobility

Unlike other leader election schemes, the cooperative leader election algorithm does not require significantly long election times when nodes move arbitrarily. Note that because of the strict connectivity rules, nodes might change their roles as a result of movement. If the moving node holds the information on the best leader, it propagates this information to other nodes in the new area. Similarly, if a node does not possess the knowledge of the unique leader and moves within proximity of nodes that do have that knowledge, it will be updated. Therefore, as the number of moving nodes and the velocity increase, we do not expect to see a degradation of performance of the algorithm in terms of the election time.

Under cooperative leader election, node arrivals are handled slightly differently. When a node joins a network, it would have its leader information set to its own id and value. There are two distinct cases to consider:

1) The network already has a leader: if the joining node has a lower value than the leader of the network, it simply adopts that leader as its own. However, if its own value is greater than that of the current leader, it informs its neighbours, which subsequently inform their own in a flooding process so that the leader information is propagated to the entire network. Updating through flooding, in this case, is a more sensible approach, as initiating a new leader election process would be wasteful since there is only one candidate.

2) The network has yet to establish its leader: when a node, n, joins the network during the election process, it learns the current iteration counter from its neighbours and updates it according to its connection information. If it assumes the role master node then it increments the counter by 1 or if it joins as a PMP node, it adds $2 \times (d_n - 1)$ to the counter (where d_n is its degree), and updates its neighbouring masters with the new counter. The node takes no action if it is a pure slave.

V. Performance Evaluation

The criteria used to evaluate the performance of the leader election algorithm are the time it takes to find the leader, its scalability and response to dynamic changes in a network topology. The algorithm is simulated on various randomly generated networks and under different assumptions of node mobility conditions.

The algorithm is first simulated in a static environment in which nodes do not move and no new node joins or leaves the network. Figure 3 shows the election time of the cooperative leader algorithm as a function of network size and compares its performance with tree-based election schemes.

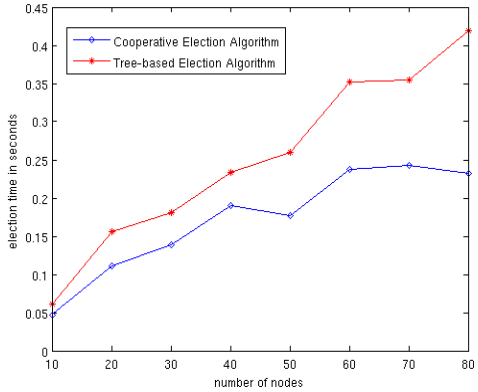
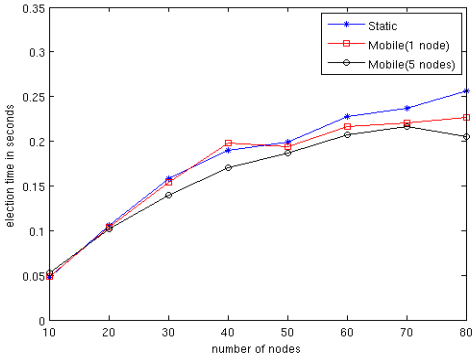


Fig. 3. Leader election time - Static environment

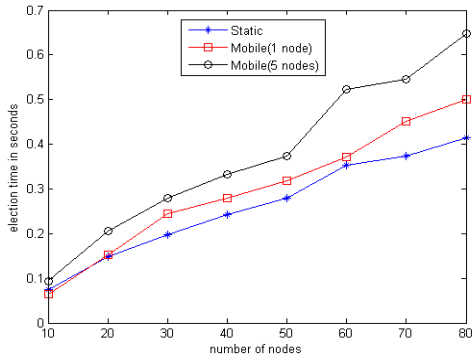
Figure 3 indicates that a key difference between tree-based algorithms and the proposed cooperative approach is that the former increases linearly with the size of the network (as one would expect from a tree algorithm) whereas the latter tends towards saturation. This fact can also be verified by Figure 2, which indicates that an increase in network size does not follow a linear increase in the number of rounds.

These results are interesting as they prove that the cooperative election algorithm scales well with network size.

The cooperative leader election method is also simulated in mobile settings, varying the velocity of nodes and the number of moving nodes. In order to evaluate the impact that frequent and unpredictable changes in network topology have on the performance of the algorithm, the simulation is performed for the exact time it takes to elect a leader when nodes move freely. Figure 4(a) illustrates comparative performance results of the cooperative leader election algorithm as a function of network size under different assumptions of node mobility conditions: static setting, mobile setting with a single node moving at a time and mobile setting with 5 simultaneous node movements in the network. The same scenarios are simulated for the tree-based election algorithm and the results are given in Figure 4(b) for comparison. Note that time delays related to nodes' displacement and connection set up delays are not taken into account.



(a) Cooperative Leader Election



(b) Tree-based Leader Election

Fig. 4. Leader election time - Mobile environment

As expected, node mobility does not deteriorate the performance of the cooperative leader election, in the sense that an increase in mobility does not yield longer election time, while the impact that mobility has on tree-based election is noticeable. This suggests that the cooperative approach is more suitable for dynamic environments.

VI. Conclusions

This paper presented a novel leader election algorithm for master/slave mobile ad hoc networks and showed that it scales well with network size and node mobility. The algorithm exploits knowledge of the underlying network, which is obtained through a random tour, in order to estimate the number of rounds a master needs to repeat the election procedure for a leader to be elected. When compared to Tree-based algorithms, it was shown that the cooperative approach performs better in terms of convergence time as well as the response to node mobility, which makes it more attractive for ad hoc settings that exhibit spontaneous changes in topology. Furthermore, the cooperative election approach may be applied to other types of ad hoc networks without any major modifications. This will be investigated in the future.

References

[1] J. Ryu, S. Song, and D. Cho, "New clustering schemes for energy conservation in two-tiered mobile ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 51, no. 6, pp. 1661–1668, 2002.

[2] M. Gerla and J. Tzu-Chieh Tsai, "Multicluster, mobile, multimedia radio network," *Wireless networks*, vol. 1, no. 3, pp. 255–265, 1995.

[3] R. Negi and A. Rajeswaran, "Capacity of power constrained ad hod networks," in *IEEE INFOCOM*, 2004, pp. 443–453.

[4] M. Shirali, A. Toroghi, and M. Vojdani, "Leader Election Algorithms: History and Novel Schemes," in *Convergence and Hybrid Information Technology, ICCIT'08. Third International Conference on*, vol. 1, 2008.

[5] S. Vasudevan, J. Kurose, and D. Towsley, "Design and analysis of a leader election algorithm for mobile ad hoc networks," in *Network Protocols, 2004. ICNP 2004. Proceedings of the 12th IEEE International Conference on*, 2004, pp. 350–360.

[6] S. Lee, R. Muhammad, and C. Kim, "A Leader Election Algorithm Within Candidates on Ad Hoc Mobile Networks," *LECTURE NOTES IN COMPUTER SCIENCE*, vol. 4523, p. 728, 2007.

[7] N. Malpani, J. Welch, and N. Vaidya, "Leader election algorithms for mobile ad hoc networks," in *Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications*. ACM New York, NY, USA, 2000, pp. 96–103.

[8] A. Derhab and N. Badache, "A Self-Stabilizing Leader Election Algorithm in Highly Dynamic Ad Hoc Mobile Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 7, pp. 926–939, 2008.

[9] I. Gupta, R. Van Renesse, and K. Birman, "A probabilistically correct leader election protocol for large groups," in *Distributed computing: 14th International Conference, DISC 2000, Toledo, Spain, October 4-6, 2000: proceedings*. Springer Verlag, 2000, p. 89.

[10] M. Aguilera, C. Delporte-Gallet, H. Fauconnier, and S. Toueg, "On implementing omega with weak reliability and synchrony assumptions," in *Proceedings of the twenty-second annual symposium on Principles of distributed computing*. ACM New York, NY, USA, 2003, pp. 306–314.

[11] R. Ali, S. S. Lor, and M. Rio, "Two algorithms for network size estimation for master/slave ad hoc networks," 2009, available: http://arxiv.org/pdf/0908.2231.

[12] C. Yu, K. Yu, and S. Lin, "Efficient Scheduling Algorithms for Bluetooth Scatternets," *Wireless Personal Communications*, vol. 48, no. 2, pp. 291–309, 2009.

[13] J. Misic and V. Misic, "Modeling Bluetooth piconet performance," *IEEE Communications Letters*, vol. 7, no. 1, pp. 18–20, 2003.

[14] D. Yang, G. Nair, B. Sivaramakrishnan, H. Jayakumar, and A. Sen, "Round robin with look ahead: a new scheduling algorithm for Bluetooth," in *Proceedings of the International Conference on Parallel Processing Workshops (ICPPW)*, 2002, pp. 45–50.

[15] S. Baatz, M. Frank, C. Kuhl, P. Martini, and C. Scholz, "Bluetooth scatternets: An enhanced adaptive scheduling scheme," in *IEEE INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, vol. 2, 2002.

[16] R. Kapoor, A. Zanella, and M. Gerla, "A fair and traffic dependent scheduling algorithm for Bluetooth scatternets," *Mobile Networks and Applications*, vol. 9, no. 1, pp. 9–20, 2004.