2012 International Conference on Smart Grid Systems (ICSGS 2012)
IPCSIT vol.45 (2012) © (2012) IACSIT Press, Singapore

An Efficient Leader Election Algorithm of Performance-Related Characteristics for Dynamic Networks

Che-Cheng Chang [a] and Jichiang Tsai [b +]

Department of Electrical Engineering National Chung-Hsing University Taichung, Taiwan

Abstract. The core of a smart grid is characterized by two-way flow of power in electrical networks as well as information in communication networks. Hence, such a new technical issue can benefit from some existing techniques related to wireless communication and fault tolerant distributed computing. Among the foregoing applications, leader election is a very critical problem. Solving the leader election problem in static networks is easier than in dynamic networks because dynamic behavior of processes must be considered in the latter. In particular, a simple way to solving this problem in dynamic networks is attaching a synchronous clock to each process. But doing so violates the assumption of asynchrony. Moreover, a leader had better be a process with the best performance-related characteristic among all nodes within a connected component. In this paper, we present an efficient leader election algorithm with regard to performance-related characteristics for dynamic networks, without any synchronous clock assumption.

Keywords: leader election, dynamic networks, asynchronous, broadcast, smart grid systems.

1. Introduction

Due to the growing demand for electric power and concerns about the environment impacts, currently power distribution networks are increasingly developing toward smart grids [1], the core of which is characterized by two-way flow of power in electrical networks and information in communication networks [2]. Particularly, the characteristics of the power line channel vary geographically. Hence, it is not safe to assume that a technology that works well in one area will work as well in another. Moreover, advanced wireless systems can offer the benefits of inexpensive products, rapid deployment, and low cost installation that wired technologies cannot provide [3]. Obviously, such wireless technical issues for smart grids can benefit from some existing techniques of wireless communication and fault tolerant distributed computing.

Leader election is a fundamental building block for many applications, such as group communication services, key distribution and management, routing coordination, sensor coordination, general control, and so on. Obviously, solving the leader election problem in static networks is easier than in dynamic networks because we have to consider the dynamic behavior of processes in the latter networks. Therefore, electing a new leader will constantly occur in such a network. This means that the traditional solution for static networks is not suitable for dynamic networks.

In the literature, a simple way to solving the leader election problem in dynamic networks is to attach a synchronous clock to each process in the system. For example, in [4], the authors used the concept of synchronous clock to indicate which node has the highest priority to be the current leader. Recently, in [5], the authors improved the algorithm proposed in [4] in a manner that the resultant convergence time of the new algorithm is less than the original algorithm. However, attaching a synchronous clock obviously violates the assumption of asynchronous systems.

+ Corresponding author
E-mail address: [a] d9864001@mail.nchu.edu.tw, [b] jichiangt@nchu.edu.tw

On the other hand, in [6], the authors considered that the elected leader should be the most-valued process from among all the processes within the connected component it resides, where the value of a process is a performance-related characteristic. Hence, a most-valued process means that this process has the best performance-related characteristic among all processes within a connected component, e.g. the longest remaining battery life, shortest average distance to other nodes, etc. Obviously, electing a leader with the best performance-related characteristic is more desirable than simply electing a random leader. For example, electing a leader with the longest remaining battery life can postpone the time for reelecting a new leader.

In this paper, we present a new leader election algorithm with regard to performance-related characteristics for dynamic networks. Likewise, we do not assume any synchronous clock. Moreover, our algorithm can elect a new leader more quickly than the algorithm proposed in [6].

2. Preliminaries

2.1. System Model

A finite set $\prod$ of processes $\{p_1, p_2, \ldots, p_n\}$ with $n \geq 1$ is considered in the text. Unlike the usual model of traditional fixed networks, the processes in $\prod$ are not necessarily aware of each other initially. This assumption reveals the self-organization nature of the considered unstructured peer-to-peer network.

Processes in $\prod$ communicate with other processes by exchanging messages through reliable channel. Hence, if the processes of two ends are correct, a message sent is eventually delivered to its destination exactly once. Moreover, process p_i can send a message to another process p_j only if p_j is within the transmission range of p_i, and vice versa. We also assume that processes in $\prod$ have the same transmission range. This means that the communication is symmetric. Furthermore, there is no bound neither on the transmission delay of messages nor on relative speeds of processes. Thus we face an asynchronous system.

On the other hand, a crashed process is considered as a leaving process since any process cannot communicate with a crashed process anymore, just like the process did leave. Similarly, we also consider a recovered process as a new joining one because any process can communicate with the recovered process from now on, just like it did join. Moreover, the distance from node p_i to node p_j is defined as the minimum number of hops among all paths from p_i to p_j. Obviously, if there is no path from p_i to p_j, the distance between p_i to p_j is infinite. Note that we use the terms "process" and "node" interchangeably in this paper.

In order to facilitate explaining the ideas proposed in this paper, we describe a few technical terms first.

Definition 1: In a network, a process is called a neighbor of p_i if and only if it is within the transmission range of p_i.

In the text, we denote the set of neighbors of p_i as $neighbors_i$. Particularly, a process can employ the techniques of Probe and Reply proposed in [6] to find which neighbor has left its transmission range or which node has entered its transmission range. In other words, we can assume that a process is able to know its neighbors all the time, even the topology is dynamic. Obviously, a neighbour of p_i is a node with one hop away from p_i.

Definition 2: In a network, a process is called a participant of p_i if and only if it and p_i are in the same connected component of the network topology.

Namely, a process can route a message to any of its participants since there exists at least one path from the process to any participant in the network topology. In the text, the set of participants of p_i is denoted as $participants_i$. Moreover, $neighbors_i$ is a subset of $participants_i$. For example, in Figure 1(a), p_1, p_3 and p_7 are the neighbors of p_9; while p_1, p_3, p_4 and p_7 are the participants of p_9. Besides, p_2 is not within the transmission range of any participant of p_9, so p_2 and p_9 cannot communicate mutually. Evidently, the two nodes p_2 and p_9, which belong to two disconnected parts, will elect distinct nodes as the leader.

2.2. The Specification of Leader Election

In the classical leader election problem for static networks, every process p_i in the network eventually elects the same leader [7]. Later, the authors in [4]-[6] adapted the classical definition for dynamic networks. They just assumed that every connected component will eventually have a unique leader, but not that eventually there is a unique leader in the system. Such a new definition is more proper for dynamic networks

because dynamic networks may contain some disconnected parts. Moreover, in [6], the elected leader must be the most-valued node because electing a leader with the best performance-related characteristic is more desirable than simply electing a leader randomly. Accordingly, the specification of the leader election problem for dynamic networks is given below:

Definition 3: The specification of the leader election problem for dynamic networks is:

1) For every connected component, there is a unique eventual leader.

2) The eventual leader of a connect component is the highest priority node among all the nodes within the component, where the priority of a node is based on the considered performance-related characteristic, using the node ID to break ties among nodes with the same value.

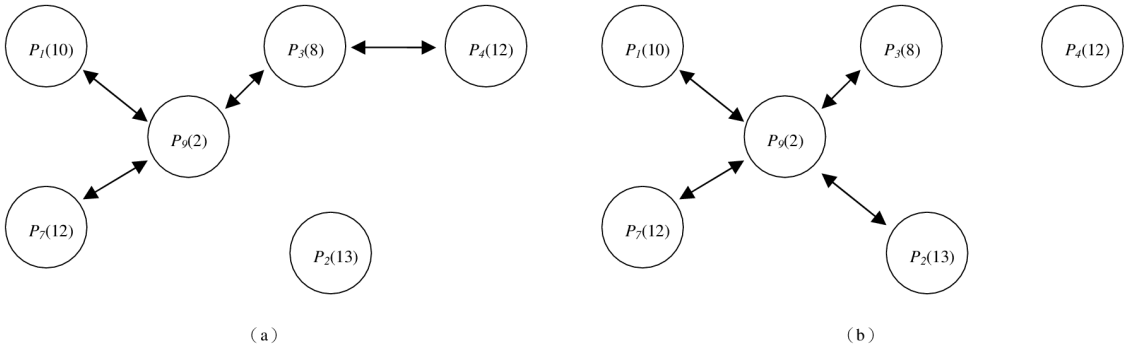

Fig. 1: (a) A topology contains two connected two components; (b) The topology after some nodes in (a) moves

3. The Leader Election Concept and Algorithm

In this section, we start to present our algorithm for solving the leader election problem. Foremost, the core property of our algorithm is that in a connected component, if every process broadcasts its neighbor set and any received neighbor set from another process, every process can finally know the existence of all other processes in the same component. Therefore, we know that a process can know all its participants by all processes broadcasting their neighbor sets as well as any received neighbor sets. Finally, all processes will choose the same process as their leader, i.e. the one with the best performance-related characteristic among all processes in the component.

Obviously, the values of the considered performance-related characteristic of any two nodes may not be distinct. For example, in Figure 1(a), the value of the considered performance-related characteristic of each process is shown in the bracket in the corresponding node. We can see that p_4 and p_7 in Figure 1(a) have the same value. If such a scenario happens, we use the Node ID of these nodes to break the tie. Particularly, the node with the smallest ID will be considered. For instance, since p_4 and p_7 in Figure 1(a) have the same performance-related characteristic value, we will choose p_4 as the highest priority node in this component.

In [6], the authors used several distinct cases to explain the behavior of dynamic networks. However, these cases cannot cover all possible behavior of dynamic networks due to rapid node mobility. By using the concept shown above, our algorithm can solve such an issue. Furthermore, the algorithm in [6] adopted the concept of "growing and shrinking" to discover all processes in the same component. Then any process uses the collective information to elect the leader and broadcasts its decision to all nodes in the same component. Obviously, such a concept requires that messages traverse a component three times for an election, i.e. growing, shrinking and broadcasting the decision about the leader. Yet, our concept only needs that any message from each node traverses a component one time for electing the leader. Particularly, upon receiving messages from all other processes in the same component, a process can immediately find the process with the best performance-related characteristic among all its participants.

In light of the idea introduced in the above, we further present the leader-election algorithm for dynamic networks. In particular, unlike static networks, the knowledge of a node in dynamic networks may be decreasing. Hence, instead of adopting the assumption of synchronous clocks, we use the following concept of routing history to deal with such a critical situation.

Definition 4: The routing history of a message contains all processes by which the message is traversed.

Trivially, while a process p_i receives a message, p_i can learn all the processes belonging to the path traversed by the message via the routing history of the message. More specifically, the routing history can prevent a node from sending the backflow message below:

Definition 5: A backflow message is a message that one process p_i transmits to another process p_j, and p_j transmits it back to p_i later.

Obviously, the information carried on a backflow message may be obsolete and then makes the problem more sophisticated and more difficult. For instance, consider the network topology shown in Figure 1(a). Since there are two disconnected components in it, two leaders will be elected in each component, i.e. p_4 and p_2. Now assume that some nodes move, and then the foregoing topology becomes the topology depicted in Figure 1(b). Particularly, because p_4 moves out of the transmission range of p_3, p_4 and p_3 cannot communicate with each other from now on. Simultaneously, p_2 enters the transmission range of p_9 such that p_2 and p_9 can communicate mutually from now on. By detecting p_4 leaving, p_3 can tell p_9 the fact. Without the routing history piggybacked on messages, after p_7 transmits a backflow message to p_9 that had been originally created by p_3 before p_4 left, p_9 will consider that p_4 is in the component again if the backflow message arrives after p_3 has told p_9 that p_4 left. If the aforementioned scenario happens again and again, the participant set of p_9 will not be stable forever. With the routing history, the foregoing scenario can be avoided because a node can never receive a message originally sent from itself by checking the routing history carried on received messages. Consider the above example again. All backflow messages broadcast by p_7 back to p_9 were sent by p_9 earlier. Therefore, p_9 is contained in the routing histories of these messages. By checking the routing history, p_9 can discard those backflow messages.

Now we begin to describe our leader election algorithm for dynamic networks presented in Figure 2. In the beginning, every process p_i takes its identification and the value of the performance-related characteristic as the input (lines 1-2). Subsequently, it initiates the algorithm by executing the init phase. First, p_i adds those nodes within its transmission rage and itself to $neighbors_i$ (lines 7-8). Similarly, it sets $participants_i$ to $neighbors_i$ to indicate that its initial knowledge about process only includes its neighbors (line 9). Then it elects the node with the highest priority in its current $participants_i$ set as the current leader (line 10). Finally, p_i broadcasts $neighbors_i$ for informing the existence of its neighbors to other nodes in the same component (line 11). On the other hand, upon receiving a message carrying $neighbors_j$, if p_i is not in the routing history of the message, it means that p_i can update its knowledge through $neighbors_j$ (line 12). Such a condition can prevent p_i from receiving a backflow message as well as broadcasting forever, even after they already know all nodes in the same component. Then p_i adds itself to the routing history of the message carrying $neighbors_j$ for noting that it has received such a message and accordingly has updated its $participants_i$ set (lines 13-14). Also, p_i elects the node with the highest priority in its current $participants_i$ set as the current leader (line 15). Then p_i broadcasts $neighbors_j$ to other processes in the same component (line 16).

There are two more affairs required to be conducted in dynamic networks. First, upon detecting neighbors changing, p_i updates both its $neighbors_i$ and $participants_i$ sets (lines 17-18). Hence, it must reelect the node with the highest priority in its current $participants_i$ set as the new leader (line 19). Subsequently, p_i broadcasts the new $neighbors_i$ set for informing the existence of nodes in $neighbors_i$ to other nodes in the same component (line 20). Particularly, p_i also broadcasts the $reelection_i$ message to ask other nodes in the same component to broadcast their neighbor sets and elect their leaders again (line 21). On the other hand, upon receiving the $reelection_j$ message, if p_i is not in the routing history of $reelection_j$, p_i adds itself to the routing history of $reelection_j$ for noting that it has received such a message (lines 22-23). Then it also broadcasts its $neighbors_i$ set again and routes the $reelection_j$ message to its neighbors (lines 24-25).

Obviously, the routing history can also help a process tolerate network partition. Since the participants of a process p_i in dynamic networks may be decreasing instead of increasing, the update operation may remove some processes from its $participants_i$ set while p_i detects network partition or receives a message about network partition. Now consider the situation where the topology shown in Figure 1(a) is changed to the one in Figure 1(b). We can see that p_4 knows that a neighbor, i.e. p_3, has left its transmission range or crashed. Then p_4 must discard all information obtained via p_3. As a result, there is only one node, i.e. p_4, in both the two sets $neighbors_4$ and $participants_4$. This means that p_4 will elect itself as the current leader.

4. Conclusions

In this paper, we have explored the problem of leader election in dynamic networks, where not only the topology will change but also processes may crash. Moreover, we use the routing history of messages to deal with the dynamic behavior of networks instead of adopting the synchronous clock assumption. Doing so makes the leader election problem easier to solve in practice. Particularly, the elected leader is the one with the best performance-related characteristic among all nodes within the same connected component. Therefore, our result can be applied to smart grid systems to improve the efficiency of energy use.

Algorithm: The leader election algorithm for dynamic networks

input:
(01) id_i: identification of p_i;
(02) v_i: performance-related characteristic value of p_i;
variables:
(03) $neighbors_i$: set of processes and routing history of $neighbors_i$;
(04) $participants_i$: set of processes and routing history of $participants_i$;
(05) $leader_i$: current leader of p_i;
(06) $reelection_i$: a message to ask to make reelection and routing history of $reelection_i$;
init:
(07) $neighbors_i \leftarrow$ detecting node(s) within transmission range;
(08) $neighbors_i \leftarrow neighbors_i \cup \{ p_i \}$;
(09) $participants_i \leftarrow neighbors_i$;
(10) $leader_i \leftarrow$ the highest-priority-node in $participants_i$;
(11) **broadcast** $neighbors_i$;
upon receipt of $neighbors_j$**:**
(12) **if** $p_i \notin$ the routing history of $neighbors_j$ **then**
(13) **add** p_i to the routing history of $neighbors_j$;
(14) **update** $participants_i$;
(15) $leader_i \leftarrow$ the highest-priority-node in $participants_i$;
(16) **broadcast** $neighbors_j$;
upon detecting neighbors changed:
(17) **update** $neighbors_i$;
(18) **update** $participants_i$;
(19) $leader_i \leftarrow$ the highest-priority-node in $participants_i$;
(20) **broadcast** $neighbors_i$;
(21) **broadcast** $reelection_i$;
upon receipt of $reelection_j$**:**
(22) **if** $p_i \notin$ the routing history of $reelection_j$ **then**
(23) **add** p_i to the routing history of $reelection_j$;
(24) **broadcast** $neighbors_i$;
(25) **broadcast** $reelection_j$;

Fig. 2: The algorithm for dealing with the leader election problem in dynamic networks.

5. Acknowledgements

This work was supported by the National Science Council, Taiwan, ROC, under Grant NSC 100–2221–E–005–038.

6. References

[1] A. Maneerung, S. Sittichivapak, and K. Hongesombut, “Application of Power Line Communication with OFDM to Smart Grid Systems,” Proc. 8th Int’l Conf. Fuzzy Systems and Knowledge Discovery (FSKD), pp. 2239–2244, 2011.

[2] P. P. Parikh, M. G. Kanabar, T. S. Sidhu, “Opportunities and Challenges of Wireless Communication Technologies for Smart Grid Applications,” IEEE Power and Energy Society General Meeting, pp. 1–7, 2010.

[3] F. Cleveland, “Use of Wireless Data Communications in Power System Operations,” IEEE Power Systems Conference and Exposition, pp. 631–640, 2006.

[4] N. Malpani, J. Welch and N. Vaidya, “Leader Election Algorithms for Mobile Ad Hoc Networks,” Proc. 4th Int’l Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, Boston, MA, August 2000.

[5] A. Derhab and N. Badache, “A Self-Stabilizing Leader Election Algorithm in Highly Dynamic Ad Hoc Mobile Networks,” IEEE Trans. Parallel and Distributed Systems, vol. 19, no. 7, JULY 2008.

[6] S. Vasudevan, J. Kurose, D. Towsley, “Design and Analysis of a Leader Election Algorithm for Mobile Ad Hoc Networks,” IEEE Proc. 12th Int’l Conf. Network Protocols, pp. 350-360, Oct. 2004.

[7] N. Lynch, Distributed Algorithms, Morgan Kaufmann, San Francisco, 1996.