Scientific Research Publishing

Journal of Transportation Technologies, 2021, 11, 404-411
https://www.scirp.org/journal/jtts
ISSN Online: 2160-0481
ISSN Print: 2160-0473

A Dynamic Leader Election Algorithm for Decentralized Networks

Vijay K. Madisetti, Siddhanta Panda

Georgia Institute of Technology, Atlanta, USA
Email: vkm@gatech.edu, panda.siddhanta@gatech.edu

How to cite this paper: Madisetti, V.K. and Panda, S. (2021) A Dynamic Leader Election Algorithm for Decentralized Networks. *Journal of Transportation Technologies*, **11**, 404-411.
https://doi.org/10.4236/jtts.2021.113026

Received: June 28, 2021
Accepted: July 16, 2021
Published: July 19, 2021

Abstract

Leader election algorithms play an important role in orchestrating different processes on distributed systems, including next-generation transportation systems. This leader election phase is usually triggered after the leader has failed and has a high overhead in performance and state recovery. Further, these algorithms are not generally applicable to cloud-based native microservices-based applications where the resources available to the group and resources participating in a group continuously change and the current leader may exit the system with prior knowledge of the exit. Our proposed algorithm, the dynamic leader selection algorithm, provides several benefits through selection (not, election) of a set of future leaders which are then alerted prior to the failure of the current leadership and handed over the leadership. A specific illustration of this algorithm is provided with reference to a peer-to-peer distribution of autonomous cars in a 5G architecture for transportation networks. The proposed algorithm increases the efficiencies of applications that use the leader election algorithm and finds broad applicability in microservices-based applications.

Keywords

Bully Algorithm, Peer-to-Peer Services, Dynamic Leader Algorithm, 5G, Cloud-Native Functions and Microservices

1. Introduction

In a computer cluster comprising multiple computers or nodes, the bully algorithm is primarily used to elect a coordinator that can serve as an orchestrator among the processes to select the node with the highest priority hosted on the cluster [1]. A priority is assigned by virtue of a load balancing algorithm very similar to those used in operating systems [2].

The traditional bully algorithm as shown in Figure 1 is triggered when and after a leader/coordinator fails and is unresponsive. One of the nodes in the cluster then initiates a broadcast asking for an election once it realizes the leader has potentially crashed. This, in turn, triggers broadcasts from every other node in the cluster. In the end, a new leader is elected.

If there are N nodes in the cluster, in the worst-case scenario, the total number of messages exchanged is $N(N+1)/2$ or $O(n^2)$. This algorithm has proved to be useful in applications such as cloud computing and within datacenters.

In certain scenarios, such as ad-hoc networks of microservices, these algorithms may not be efficient or suitable for reasons that will be outlined below. For instance, peer-to-peer distribution of a "data center on wheels" comprising autonomous cars that are continually in motion would imply that cars (and their processing units) will be continuously entering and leaving the radio ranges of clusters [3]. The traditional algorithms are not suitable because nodes in such a network (including the current leader, elected leader, or the node calling the election) may leave the system at any point.

2. Related Work

There are several different leader election algorithms widely used in dependable distributed systems. Cloud service providers, such as Amazon Web Services (AWS) [4], Windows Azure [5], and Google Cloud [6] use leader election algorithms as a means for assigning nodes and microservices for different purposes. AWS, Azure and Google Cloud use algorithms such as Paxos, RAFT and software such as Apache Zookeeper that monitor leader "heartbeat" to detect failure to call for leader elections [4] [5] [6]. In addition to the bully algorithm, the Ring Algorithm (Chang and Roberts Algorithm) is also commonly used for leader elections [7]. Recently local leader election protocols for decentralized vehicular

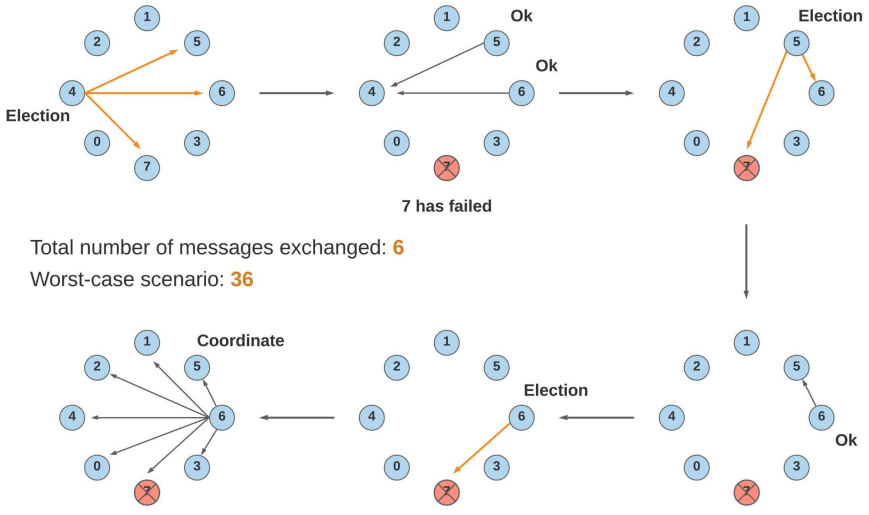

Figure 1. Traditional bully algorithm is employed by node 4 to elect a new leader only after detection of the crash of the current leader 7. After an election takes place, a new leader 6 takes over.

traffic regulation involving a Vehicle to Vehicle (V2V) communication in a 5G architecture [8]. Additional modified bully algorithms have been developed for use in wireless networks [5] and cloud-based systems [9] and other contexts [10]. All these algorithms are triggered only after the failure of the appointed leader.

In many cases it is possible to predict in advance when the leader is about to fail. This motivates a dynamic leader algorithm where the leader can be "elected" through pre-selection quickly, prior to predicted exit (or failure) of the current leader. The number of messages in the worst-case is reduced by an order of magnitude and simulations show improvement of average number of messages exchanged over previous approaches.

3. Proposed Approach

Our proposed *dynamic leader algorithm*, while broader than discussed here, can be studied through the following exemplary embodiments:

1) Variation 1: Once an exit is predicted from the cluster, a current leader probes all the nodes and decides (according to a cost function) which unit to relinquish its state to (and thus pre-selecting a new leader) prior to exiting the cluster (See **Figure 2**).

2) Variation 2: Once elected to leadership, a leader designates another node to share its leadership state and maintains a "leadership list" to aid in handover prior to its predicted exit from the cluster (See **Figure 3**).

3) Variation 3: Once elected to leadership in the cluster, a current leader designates a set of nodes that will be sharing its state at all times to aid in handover during and prior to its predicted exit from the cluster (See **Figure 4**).

In all these three variations of the dynamic leader algorithm, the elected leader decides which node(s) shares its state. Instead of waiting for the leader to go

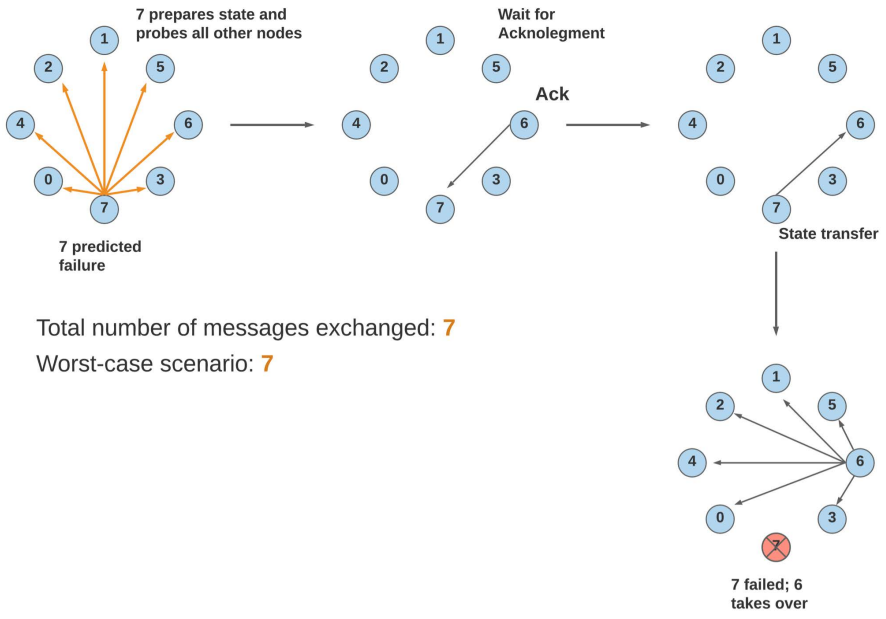

Figure 2. Operation of variation 1.

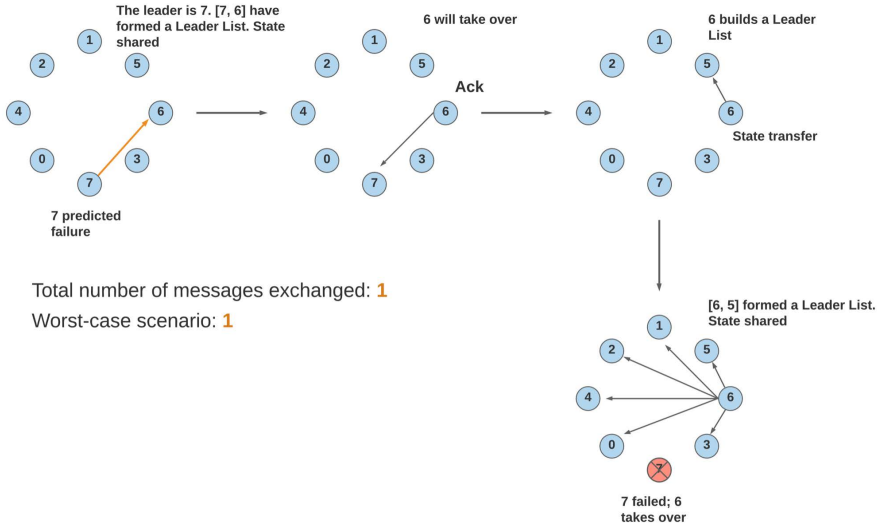

Figure 3. Operation of Variation 2 with a leadership list of two.

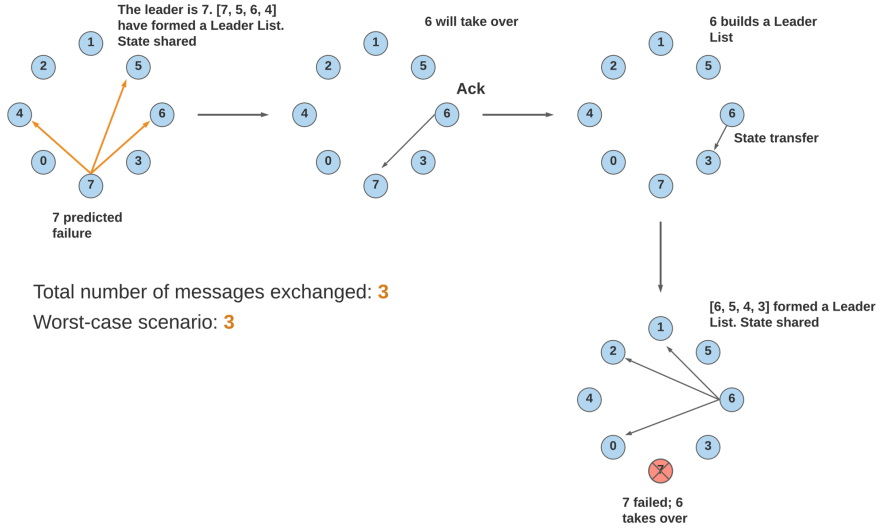

Figure 4. Variation 3 with a leadership list of four.

offline and trigger the traditional bully algorithm to elect a new leader, our proposed algorithm presents a potential way of electing and handing over (prior to failure) to a new leader with a reduced number of messages, which lessens the latency of the entire process compared to that of the traditional bully algorithm.

The three variations of the algorithm differ based on when the leader decides which node(s) shall be provided the state on a predictive exit. The first variation involves deciding whom to hand over the state when it predicts its exit from the cluster. The second variation involves making this decision during the leader election. On being elected before coordination, the leader decides which node will get the state on predicted exit. Finally, the third variation proposes the generation of a list of prospective leaders, all of whom share the state of the leader. This list should be generated once the leader has been elected. As a failover, the traditional leader algorithm may be applied should a leader fail before it can ap-

ply the dynamic leader selection algorithm.

4. Performance Evaluation for a 5G Transportation Network

We evaluate the performance of the proposed algorithms in a use case of clusters of autonomous cars in a 5G cellular network setting to compare the performance of the three dynamic leader algorithms against the static bully algorithm.

Within the 3rd Generation Partnership Project 3GPP 5G standards, the V2X (Vehicle to Everything) includes communications between vehicles (V2V), vehicles and the infrastructure (V2I), and vehicles and the network (V2N) [11]. V2X involves the use of Cellular-V2X (CV2X) and IEEE V2X technologies to facilitate efficient communications [11]. The 5G V2X infrastructure involves cars, Road Side Units (RSU) and the cloud. The IEEE standard, IEEE 802.11p is commonly used for wireless communications between cars and between cars and the Road Side Units (RSU) [11]. We are considering the model as shown in Figure 5 which demonstrates the formation of distributed networks by the autonomous vehicles which communicate with the RSUs through the elected leader.

A cluster of cars shares information between each other, and only contacts the data center when there is no "hit" to information within the cluster. The following are the assumptions of the 5G V2X architecture [11]:

- Cars, containing processing units, form a cluster and continually contact and communicate the Road Side Units through a leader/coordinator and also with each other.
- Cars may contact every other car in the cluster through 5G transceivers.

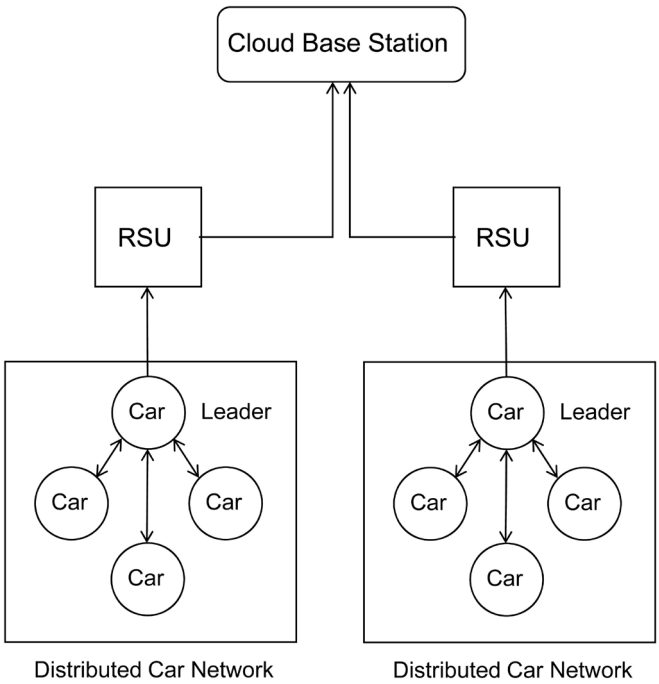

Figure 5. The V2X environment comprising a decentralized network of autonomous cars in communication with RSUs and the cloud.

- All cars produce different types of requests for information that are handled through hits/misses by other cars or by the RSU cluster. If there is a miss then the RSU queries the base station in the cloud for the required information.
- Requests and responses may require a series of computations that other cars/RSU may carry out.
- Which requests are handled on which processing unit is decided by the leaders based on balancing load and other cost functions (e.g., network load, interference, and security).

In this paper, we will interpret the number of messages exchanged as an example of a cost function. In other words, we say a system performed poorer than another system if and only if the number of messages exchanged was greater than that of the latter.

The following are the possible scenarios for a cluster consisting of a set of nodes, amongst which is a selected set of nodes belonging to a "leader list".

- A leader could exit with/without setting up the Leader List in a predictive manner.
- A leader inadvertently fails with/without setting up the Leader List.
- Any node in the Leader List exits.
- Anyone in the no-Leader List could exit.
- Anyone could enter the cluster.
- Elements in the Leader List can inadvertently fail during the simulation.

The traditional bully algorithm is called as a baseline in all the six cases. For the dynamic leader election case with between two and ten leaders in the Leader List, handovers occur as designed when the leader fails with the Leader List fully set up. When anyone processing unit (or car) enters the cluster, there is some overhead cost in determining if the tasks need to be taken up by the Leader List. Any inadvertent failure of any element in the Leader List involves transferring an element from the non-Leader List to the Leader List. Every attempt is made to keep the Leader List complete with the required number of elements.

If all leaders in the Leader List fail without carrying out the steps needed, the traditional bully algorithm would be used to determine the leader and then subsequently the Leader List. This static leader algorithm is triggered when one of the cars notices that the entire Leader List is unresponsive. Predictive exit involves a car in the Leader List taking over operations from the exiting leader and beginning orchestrations as soon as the leader exits.

Without a Leader List, the dynamic leader algorithm, on a predictive exit, probes all the elements in the cluster linearly to select which node to relinquish the state and elect the leader.

Figure 6 describes the number of messages exchanged for the three variations compared to prior art static baseline. An initial fixed number of cars in the cluster were subjected to event triggers from different scenarios outlined above. It is worthwhile to note that the best case performance of all the three proposed variations is O(1) when the first car the leader requests for handover acknowledges,

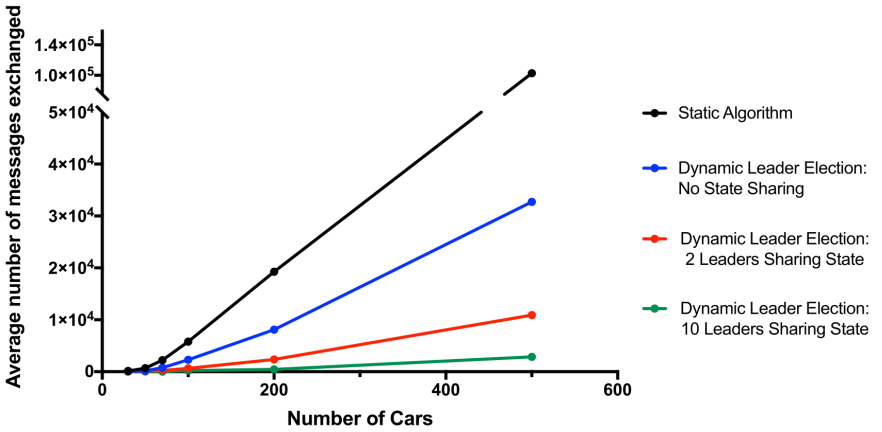

Figure 6. The proposed dynamic algorithm shows a significant improvement in the average number of messages exchanged between autonomous cars over the traditional static leader election algorithms.

while the best case performance for the baseline static bully algorithm is O(N) where N is the number of cars in the cluster.

5. Conclusion and Future Work

This paper has proposed new and efficient leader selection/election algorithms in decentralized computing and communicating clusters that dynamically change over time as in transportation networks. Current and future work is focused on implementing these algorithms to realize network slices within the 5G/6G networking environments for autonomous vehicles.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Garcia-Molina, H. (1982) Elections in a Distributed Computing System. *IEEE Transactions on Computers*, **C-31**, 48-59. https://doi.org/10.1109/TC.1982.1675885

[2] Kiyarazm, O., Moeinzadeh, M.-H. and Sharifian-R, S. (2011) A New Method for Scheduling Load Balancing in Multi-processor Systems Based on PSO. 2011 *Second International Conference on Intelligent Systems, Modelling and Simulation*, Phnom Penh, Cambodi, 25-27 January 2011, 71-76. https://doi.org/10.1109/ISMS.2011.73

[3] Berger, C., Nguyen B. and Benderius, O. (2017) Containerized Development and Microservices for Self-Driving Vehicles: Experiences & Best Practices. 2017 *IEEE International Conference on Software Architecture Workshops* (*ICSAW*), Gothenburg, Sweden, 5-7 April 2017, 7-12. https://doi.org/10.1109/ICSAW.2017.56

[4] Amazon Web Services (n.d.) Leader Election in Distributed Systems. https://aws.amazon.com/cn/builders-library/leader-election-in-distributed-systems/

[5] Windows Azure Services (n.d.) Leader Election Pattern. https://docs.microsoft.com/en-us/azure/architecture/patterns/leader-election

[6] Google Cloud Services (n.d.) Implementing Leader Election on Google Cloud Storage.

https://cloud.google.com/blog/topics/developers-practitioners/implementing-leader-election-google-cloud-storage

[7] Chang, E. and Roberts, R. (1979) An Improved Algorithm for Decentralized Extrema-Finding in Circular Configurations of Processes. *Communications of the ACM*, **22**, 281-283. https://doi.org/10.1145/359104.359108

[8] Elchamaa, R., Guériau, M., Dafflon, B., Chamoun R.K. and Ouzrout, Y. (2017) A Local Leader Election Protocol Applied to Decentralized Traffic Regulation. 2017 *IEEE 29th International Conference on Tools with Artificial Intelligence* (*ICTAI*), Boston, MA, 6-8 November 2017, 1013-1020. https://doi.org/10.1109/ICTAI.2017.00156

[9] Cahng, H. and Lo, C. (2012) A Consensus-Based Leader Election Algorithm for Wireless Ad Hoc Networks. 2012 *International Symposium on Computer, Consumer and Control*, Taiwan, 4-6 June 2012, 232-235. https://doi.org/10.1109/IS3C.2012.66

[10] Biswas, A. and Dutta, A. (2016) A Timer Based Leader Election Algorithm. 2016 *Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress* (*UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld*), Toulouse, France, 18-21 July 2016, 432-439. https://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0079

[11] Abdel Hakeem, S.A., Hady, A.A. and Kim, H. (2020) 5G-V2X: Standardization, Architecture, Use Cases, Network-Slicing, and Edge-Computing. *Wireless Networks*, **26**, 6015-6041. https://doi.org/10.1007/s11276-020-02419-8