Non-Pre-emptive Leader Node Election Algorithms for Maintaining Connected Network in Energy Conscious MANET

Arnab Mallik[*] and Abhijit Das[**]
[*]Infosys. Ltd. Bhubaneswar, India
arnabdmallik@gmail.com
[**]RCC Institute of Information Technology Kolkata, India
ayideep@yahoo.co.in

Abstract: In this paper, we shall propose three algorithms for leader node election for an adaptive network of movable nodes in a mobile ad-hoc network (MANET) with an effort to minimize overhead costs during election process. A MANET or mobile ad-hoc network is a collection of wireless devices/nodes which needs to communicate with each other in the absence of any well-defined infrastructure and these devices are constantly moving autonomously. A source node will communicate with a destination node with the help of other intermediate nodes, thus it is of utmost importance to prevent the individual nodes from breaking away from the network. Our simulation model follows hybrid routing.
A leader node will be elected from among the participating nodes, based on current network topology, movement trends and other parameters, which will be responsible for maintaining the network membership by directing the velocity of movement of the other nodes. It is the responsibility of the Leader Node to prevent the network from breaking apart and allow routing to continue. Once a leader fails some predefined criteria, it ceases to remain leader and it will behave like any non-leader node, and there will be a leadership handover. The change of leader is done without any re-election so as to preserve node battery life.

Keywords: MANET; Topology; Hybrid Routing; Priority Queue; Network Movement; Simulation; Leader Node Election; Node Weight;

Introduction

Mobile ad-hoc network (MANET) [3, 5] is suitable for use where communications infrastructure is either unavailable or deployment is infeasible like in military networks [6]. Each wireless device (from here on referred to as node) in MANET act as router as message transmission from the source node to the destination node requires routing over intermediate nodes. These nodes which are taking part in message transmission (routing) often suffer from link failure due to their constant autonomous movement. Thus, it is very important to control the mobility of nodes when designing any routing protocol.
Routing schemes in MANET can broadly be classified into three categories – Proactive Routing, Reactive Routing and Hybrid Routing [4, 7, 8, 9].
In Proactive Routing, a list of destination nodes and the paths to reach the destination are maintained. This protocol has to maintain a lot of data about the nodes. Also, this protocol is not suitable when network topology is constantly changing as stored data about the destination and path also need to be updated [13].
In Reactive Routing, a path between source and destination node is determined by flooding the network with discovery packets. The disadvantage of this protocol is that flooding of discovery packets may lead to network clogging.
The Hybrid Routing protocol combines the advantages of Proactive and Reactive Routing Protocols. Advantage of Hybrid Routing depends on the number of nodes activated.
Also, reaction to traffic demand depends on gradient of traffic volume.
None of these routing algorithms can work unless and until the network is stable and well connected during its movement, thus it is critical for the communicating nodes to remain connected for routing of messages to take place [10, 11, 12].
An Adaptive Movement Algorithm and Leader Node Election Algorithm was proposed to maintain network connectivity during node movement. A leader node is elected from among the participating nodes based on Node Weight. Each node has some parameters like Node Importance, Node Hop Count, Node Hop Strength and Current Battery Life" [14, 15].
All these parameters determine the Node Weight of a particular node. Higher the Node Weight, greater the chance of becoming the leader, provided other criteria (like battery life remaining) are also fulfilled.
Intuitively, we can think of Node Weight as a quantitative measure of how well a node is connected and located in the network and for how long it can operate.

This paper proposes Three Leader Election Algorithms which refines the leader election process by trying to minimize time loss and battery drain during Leader Election. We also compare and contrast the three algorithms proposed.
The scientific contributions of this paper are:

- The leader election algorithms are more resource and energy conserving which is achieved by trying to minimize the overhead incurred during Election Process.
- All the nodes of network remain connected. Thus proving the Leader Node selected does its job.

Assumptions and Background Concepts

All the algorithms are based on some assumptions and background concepts [1, 2, 22, 23].

- All the nodes are assumed to be moving along the Y-axis in the positive direction. There is no movement along the X-axis. This is done to keep the calculations simple.
- Message sent from source node to a destination node at one hop distance is always received if nodes are within the communication range i.e.- network is perceived to be a lossless network [16].
- The various node information such as location, velocity, battery are all assumed to be correct throughout the network runtime.
- The network is connected at the beginning. In other words, there is at least one path between every pair of node. When any new node is added to the network, that node must also be connected. Since the topology management algorithm followed is adaptive in nature, the network remains connected throughout. This can be achieved using Self Stabilizing Spanning Tree Algorithm. [18, 19, 20, 21].
- Every node is given a unique natural number for identification. Node numbering are done like Node 1, Node 2, Node 3, … Node n.
- The nodes communicate with each other in full-duplex mode.
- Every node has the same maximum velocity, maximum communication distance and battery capacity. Also, the logic required for calculating hop count, hop strength, node weight, are known to all the nodes present.
- All the nodes are aware of the direction of movement. The leader node may instruct any node to slow down to keep the network connected. The leader node will never instruct any node to change its movement direction.
- It is assumed that none of the nodes will suffer from any circuit failure and become unstable while the network is at work.

The maximum communication range is assumed to be 100kms. The maximum neighbourhood distance (the distance between nodes to be considered as neighbours) is assumed to be 50kms. The maximum node velocity is chosen as 60km/hr. All the nodes are assumed to have 100% battery life initially.

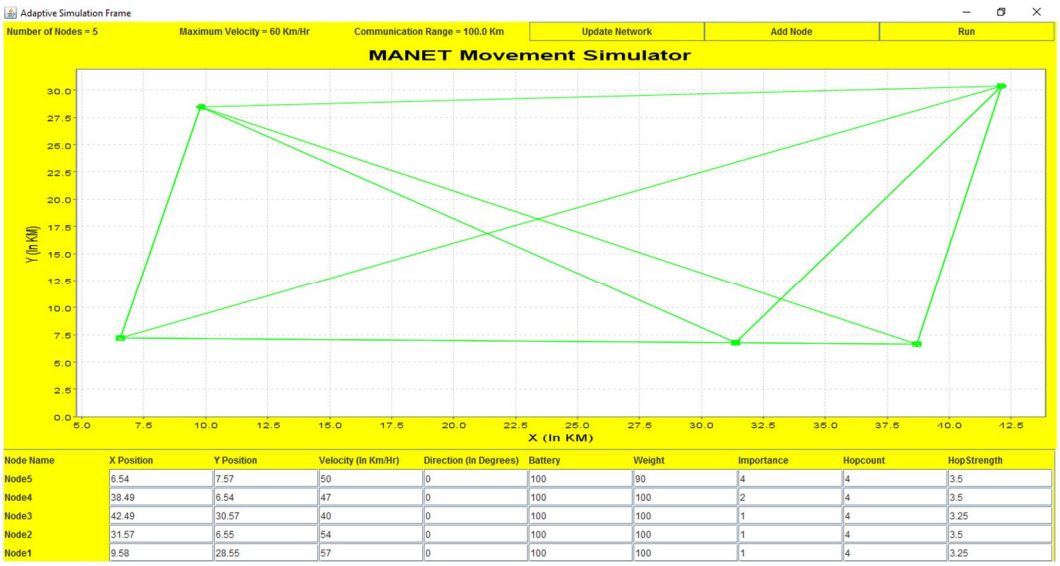


Figure 1. Simulation is run with 5 nodes

Figure 1, shows the simulation in action. Nodes are placed at random locations (X-Position and Y-Position). Each node is assigned various network parameters w.r.t. the location they are placed in and the location and distance of their neighbours. Figure 2 shows the network parameters in greater details and the roles are elaborated.

- Importance is a rank given to each node in the network. Rank is a natural number having value that range from 1 to n, where n is the total number of nodes, and can have identical values when nodes are having similar responsibilities.
- Battery strength of any node is the remaining battery life at any time instance. In "Reference [17], Battery strength has been measured as:
  $BS = E \div (P_S - T_A \times P_S + T_A \times P_A)$
  Where, BS=Battery Strength, T_A=Time in Active State, P_A=Power consumed in Active State, E=Battery Energy, P_S=Power consumed in Sleep State.
- Distance D is 50% of maximum Neighbourhood Distance i.e. - 50kms. Four fixed values- 0.25, 0.5, 0.75 and 1 are used to calculate Hop Strength.
  If, 0.75D < (node distance) <= 1.00D, then OneHopStrength = 0.25
  If, 0.50D < (node distance) <= 0.75D, then OneHopStrength = 0.50
  If, 0.25D < (node distance) <= 0.50D, then OneHopStrength = 0.75
  If, 0.00D < (node distance) <= 0.25D, then OneHopStrength = 1.00
  Strength of a multi hop route = ∑minimum {OneHopStrength for all nodes in any path}
- The summation of hop strengths of a node is known as Neighbour Score (NS). Neighbourhood strength is computed with the help of total neighbour score.
  1-hop NS = [1 * (count 1-hop neighbours of strength 1)
  + 0.75 * (count 1-hop neighbours of strength 0.75)
  + 0.50 * (count 1-hop neighbours of strength 0.50)
  + 0.25 * (count 1-hop neighbours of strength 0.25)].
  2-hop NS = [1 * (count 2-hop neighbours of strength 1)
  + 0.75 * (count 2-hop neighbours of strength 0.75)
  + 0.50 * (count 2-hop neighbours of strength 0.50)
  + 0.25 * (count 2-hop neighbours of strength 0.25)]
  …
  …
  …

  N-hop NS = [1 * (count n-hop neighbours of strength 1)
  + 0.75 * (count n-hop neighbours of strength 0.75)
  + 0.50 * (count n-hop neighbours of strength 0.50)
  + 0.25 * (count n-hop neighbours of strength 0.25)]
  For calculating n-hop neighbour score, the minimum value from different n-hop is chosen.

  Neighbourhood Strength = $\sum_{i=1}^{i=n}$ i-hop neighbour score

- Let the weight percentage for hop count (HC), importance (I), remaining battery strength (RBS) and hop strength (HS) be α, β, γ and δ respectively. Hence node weight is calculated as:
  $\alpha * [HC / MHC] + \beta * [(n - I + 1) / n] + \gamma * [RBS / 100] + \delta * [HS / MHS]$
  Where MHC and MHS are maximum hop count and maximum hop strength of a node respectively".

Node Name	X Position	Y Position	Velocity (In Km/Hr)	Direction (In Degrees)	Battery	Weight	Importance	Hopcount	HopStrength
Node5	6.54	7.57	50	0	100	90	4	4	3.5
Node4	38.49	6.54	47	0	100	100	2	4	3.5
Node3	42.49	30.57	40	0	100	100	1	4	3.25
Node2	31.57	6.55	54	0	100	100	1	4	3.5
Node1	9.58	28.55	57	0	100	100	1	4	3.25

Figure 2. Network properties of the 5 nodes are shown

Proposed Work

Three algorithms have been proposed in this paper. First the descriptions, then the pseudo codes of the algorithms are described. For all the algorithms proposed, the following values hold true.
The node battery life starts from 100%.
Leader Handover Battery Level = 50%
Node Deletion Battery Level = 10%

Static Weight Leader Election Algorithm

Algorithm
The Node Weights are calculated at the beginning. All the Weights remain constant and are unaffected by changing network parameters. Nodes are elected as Leader solely based on their Weights. New nodes are also eligible for election as soon as they become part of the network.

1.1. Input the number of nodes.
1.2. Weights of all the nodes are calculated using parameters like Node Importance, Current Battery Life, Node Hop Count, Node Hop Strength. The Node Weights once calculated will be constant throughout the simulation run irrespective of change in Battery, Hop Count etc.
1.3. Two Priority Queues (queue1 & queue2) are maintained. In both the queues, nodes are de-queued according to their node weight i.e.- node with highest node weight is removed from queues first.
1.4. All the nodes in the network are inserted into queue1 when simulation is run. New Nodes added to the network are also inserted into queuel1.
1.5. The node which is removed from queue1 (node with highest priority) will be the Leader Node. A Leader will continue to remain leader node unless and until it's battery life falls below 50%. Then new leader node is selected from queue1 based on Node Weight. Once battery life of any node falls below 50%, it is inserted into queue2.
1.6. A time will come when all the nodes in network will have battery life below 50%. That time none of the nodes will be eligible to become Leader Node. That time Leader Node will be elected from nodes present in queue2 i.e.- Node with highest weight and battery life between 10% and 50%.
1.7. Now if a new Node is added to the network, it will be inserted into queue1 and is also eligible for Leadership Election.
1.8. if queue2 isNotEmpty, then Leader Node will be the node with highest Weight considering both queue1 and queue2.
1.9. Any node having battery life below 10% will be removed from the network.

Priority Queues queue1 and queue2: Priority given to node weight.

Pseudo code

```
1.  for tempNode in 1 → n
2.      calculate(nodeWeight)
3.      insert(queue1(tempNode))
4.  end for
5.  if(#nodes = = n)
6.      while(queue1 isNotEmpty)
7.          leaderNode ← delete(queue1)
8.          while(leaderNode.BATTERY_LIFE >= LEADER_HANDOVER_BATTERY_LEVEL)
9.              runSimulation()
10.         end while
11.         insert(queue2(leaderNode))
12.     end while
13.     while(queue2 isNotEmpty)
14.         leaderNode ← delete(queue2)
15.         while(leaderNode.BATTERY_LIFE >=  NODE_DELETION_BATTERY_LEVEL)
16.             runSimulation()

17.         end while
18.         delete(leaderNode)
19.     end while
20. else
21.     calculate(newNodeWeight)
```

```
22.    insert(queue1(newNode))
23.    goto 5
24.  end if
```

Dynamic Weight Leader Election Algorithm

Algorithm

The Weights are calculated at every instance of time. All the Weights are dynamic and are affected in real time by the changing network parameters. Nodes are elected as Leader solely based on their Weights. New nodes are also eligible for election as soon as they become part of the network.

2.1. Input the number of nodes.
2.2. Weights of all the nodes are calculated using parameters like Node Importance Current Battery Life Node Hop Count, Node Hop Strength. The Node Weights are dynamic in nature i.e. - the node weights are constantly changing based on changing values of in Battery, Hop Count etc.
2.3. Two Priority Queues (queue1 & queue2) are maintained In both the queues nodes are de-queued according to their node weight i.e.- node with highest node weight is removed from queues first.
2.4. All the nodes in the network are inserted into queue1 when simulation is run. New Nodes added to the network are also inserted into queue1.
2.5. The node which is removed from queue1 (node with highest priority) will be the Leader Node. A Leader will continue to remain leader node unless and until it's battery life falls below 50%. Then new leader node is selected from queue1 based on Node Weight. Once battery life of any node falls below 50%, it is inserted into queue2.
2.6. A time will come when all the nodes in network will have battery life below 50%. That time none of the nodes will be eligible to become Leader Node. That time Leader Node will be elected from nodes present in queue2 i.e. - Node with highest weight and battery life between 10% and 50%.
2.7. Now if a new Node is added to the network, it will be inserted into queue1 and is also eligible for Leadership Election.
2.8. If queue2 isNotEmpty, then Leader Node will be the node with highest Weight considering both queue1 and queue2.
2.9. Any node having battery life below 10% will be removed from the network.

Priority Queues queue1 and queue2: Priority given to node weight.

Pseudo code

```
1.  for tempNode in 1 → n
2.      calculate(nodeWeight)
3.      insert(queue1(tempNode))
4.  end for
5.  if(#nodes = = n)
6.      while(queue1 isNotEmpty)
7.          leaderNode ← delete(queue1)
8.          while(leaderNode.BATTERY_LIFE >= LEADER_HANDOVER_BATTERY_LEVEL)
9.              for each beacon-signal:
10.                 runSimulation()
11.                 calculate(nodeWeight)
12.             end for
13.         end while
14.         insert(queue2(leaderNode))
15.     end while
16.     while(queue2 isNotEmpty)
17.         leaderNode ← delete(queue2)

18.         while(leaderNode.BATTERY_LIFE >=  NODE_DELETION_BATTERY_LEVEL)
19.             runSimulation()
20.         end while
21.         delete(leaderNode)
22.     end while
23. else
24.     calculate(newNodeWeight)
```

```
25.     insert(queue1(newNode))
26.     goto 5
27.  end if
```

Senior Static Weight Leader Election Algorithm

Algorithm

The Node Weights are calculated at the beginning. All the weights remain constant and are unaffected by changing network parameters. Nodes are elected as Leader based on their seniority. Older Nodes become Leader Node first. If two nodes have same seniority, then Node Weight is used to break the deadlock, i.e.- the Node with higher Weight becomes Leader. New Nodes are not eligible for election as soon as they become part of the network because they are the most junior nodes.

3.1. Input the number of nodes.
3.2. Weights of all the nodes are calculated using parameters like Node Importance, Current Battery Life, Node Hop Count, Node Hop Strength. The Node Weights once calculated will be constant throughout the simulation run irrespective of change in Battery, Hop Count etc.
3.3. Two Priority Queues (queue1 & queue2) are maintained. In both the queues, nodes are de-queued per their arrival time (node seniority) i.e.- node which are arrived first are de-queued first. If more than one node has arrived at the same time, then node with higher Node Weight is de-queued first.
3.4. All the nodes in the network are inserted into queue1 when simulation is run. New Nodes added to the network are also inserted into queue1.
3.5. The node which is removed from queue1 (node with highest priority) will be the Leader Node. A Leader will continue to remain leader node unless and until it's battery life falls below 50%. Then new leader node is selected from queue1 based on Node Seniority (Node Weight if arrival time is same). Once battery life of any node falls below 50%, it is inserted into queue2.
3.6. A time will come when all the nodes in network will have battery life below 50%. That time none of the nodes will be eligible to become Leader Node. That time Leader Node will be elected from nodes present in queue2 i.e.- Node with highest Node Seniority (Node Weight if arrival time is same) and battery life between 10% and 50%.
3.7. Now if a new Node is added to the network, it will be inserted into queue1 and is also eligible for Leadership Election as per Node Seniority i.e.- It will be placed at last in queue1.
3.8. Any node having battery life below 10% will be removed from the network.

Priority Queues queue1 and queue2: Priority given to node arrival time. If arrival times are equal, node weight is used to break deadlock.

Pseudo code

```
1.  for tempNode in 1 → n
2.     calculate(nodeWeight)
3.     insert(queue1(tempNode))
4.  end for
5.  if(#nodes = = n)
6.     while(queue1 isNotEmpty)
7.        leaderNode ← delete(queue1)
8.        while(leaderNode.BATTERY_LIFE >= LEADER_HANDOVER_BATTERY_LEVEL)
9.           runSimulation()

10.       end while
11.       insert(queue2(leaderNode))
12.    end while
13.    while(queue2 isNotEmpty)
14.       leaderNode ← delete(queue2)
15.       while(leaderNode.BATTERY_LIFE >=  NODE_DELETION_BATTERY_LEVEL)
16.          runSimulation()
17.       end while
18.       delete(leaderNode)
19.    end while
```

```
20. else
21.     calculate(newNodeWeight)
22.     insert(queue1(newNode))
23.     goto 5
24. end if
```

Simulation Results

Start simulation with 3 nodes. Then add a new node before 1st leader handover time.
Therefore, total no. of nodes = 4.
7 time intervals are considered for this comparison.
The simulation, when run asks for the number of nodes to be entered, as shown in Figure 3. A random network is generated as seen in Figure 4. However, all nodes are present within the maximum communication range,

Figure 3. Simulation is run with 3 nodes

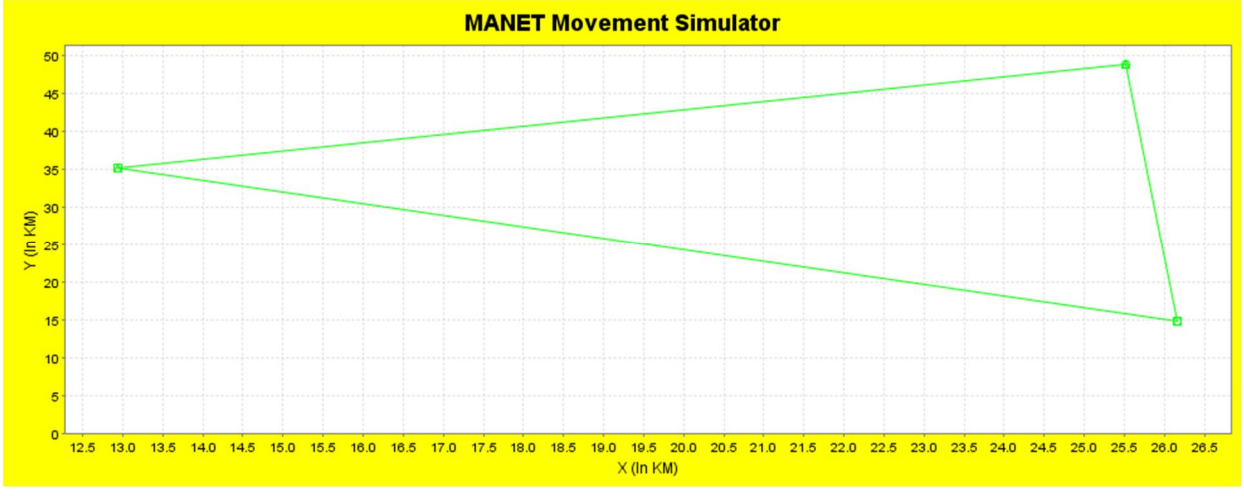
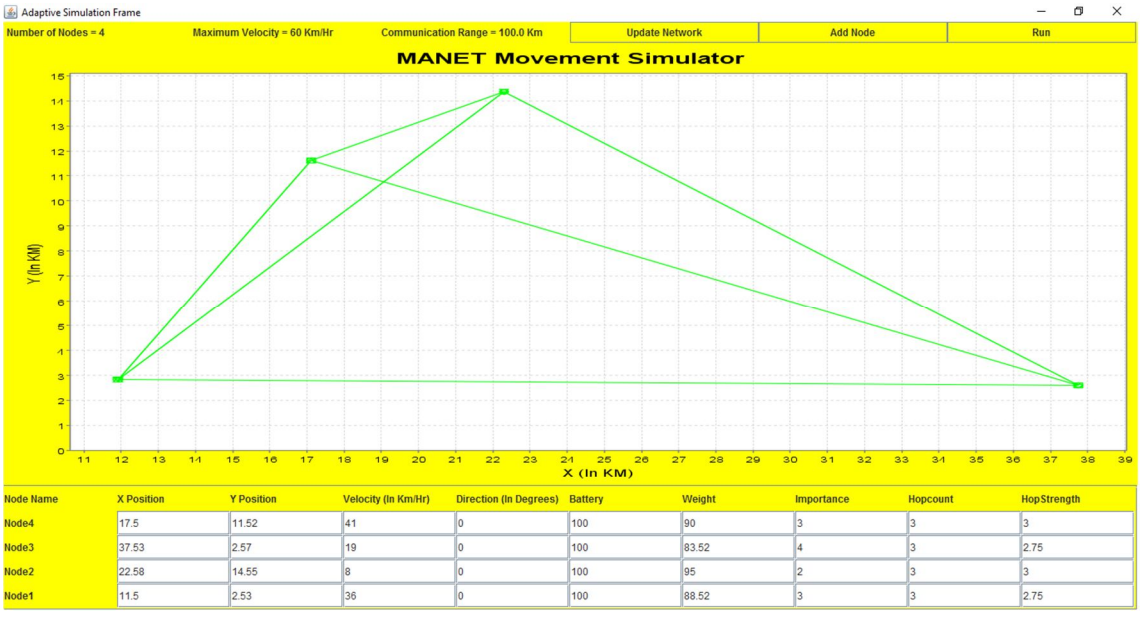


Figure 4. Simulation creates a random network of 3 nodes

When simulation is run, the nodes move along Y-axis only. Horizontal movement is restricted so as to simplify the calculation of various node parameters.
The simulation is capable of adding new nodes as well as updating the network with user defined values of X Position, Y Position, Velocity, Battery and Importance as shown in Figure 5.
There is no provisioning to show the two priority queues in the simulation. These queues are maintained internally.
When a node travel too far from network, as shown in Figure 7, link colour changes from green to red, indicating that link may break if the nodes are allowed to move further. If a node is lagging behind the network, then all the other nodes will not proceed further to allow the lagging node to catch up. Similarly, if a node is ahead of the network, then this node will stop moving to allow other nodes to catch up.
To compare the three algorithms proposed, the simulation is run with same number of nodes three times, with each node having the same parameters i.e. - same value for X Position, Y Position, Velocity, Battery and Importance. Then battery life of all the nodes are recorded and average battery life of the whole network is calculated at pre-defined time intervals.
The above steps are repeated several times to find a mean of the average network battery life.

When battery level of at least 1 node >= LEADER_HANDOVER_BATTERY_LEVEL
When battery level of all the nodes < LEADER_HANDOVER_BATTERY_LEVEL



Node Name	X Position	Y Position	Velocity (In Km/Hr)	Direction (In Degrees)	Battery	Weight	Importance	Hopcount	HopStrength
Node4	17.5	11.52	41	0	100	90	3	3	3
Node3	37.53	2.57	19	0	100	83.52	4	3	2.75
Node2	22.58	14.55	8	0	100	95	2	3	3
Node1	11.5	2.53	36	0	100	88.52	3	3	2.75

Figure 5. Simulation can add new node and update the network

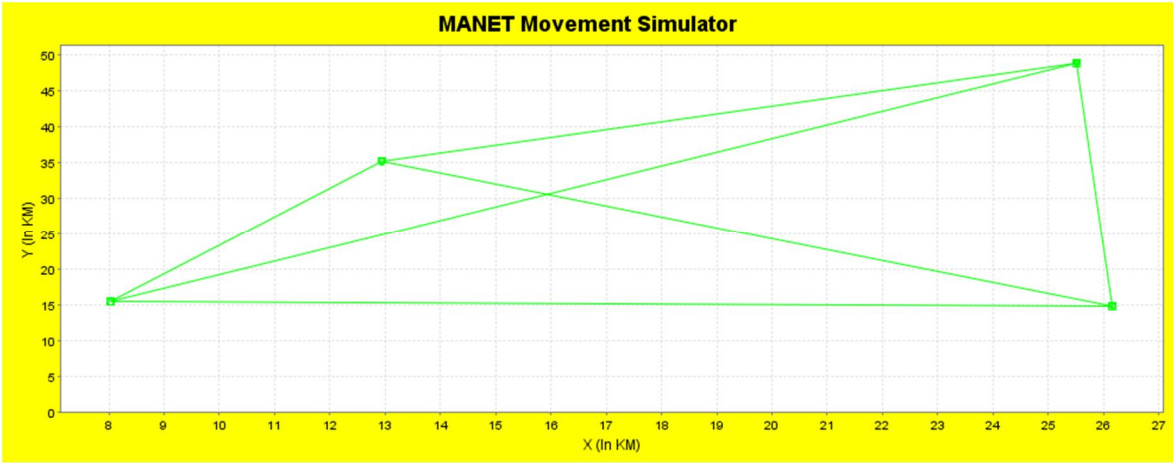


Figure 6. New node is added to the network. Total nodes are 4

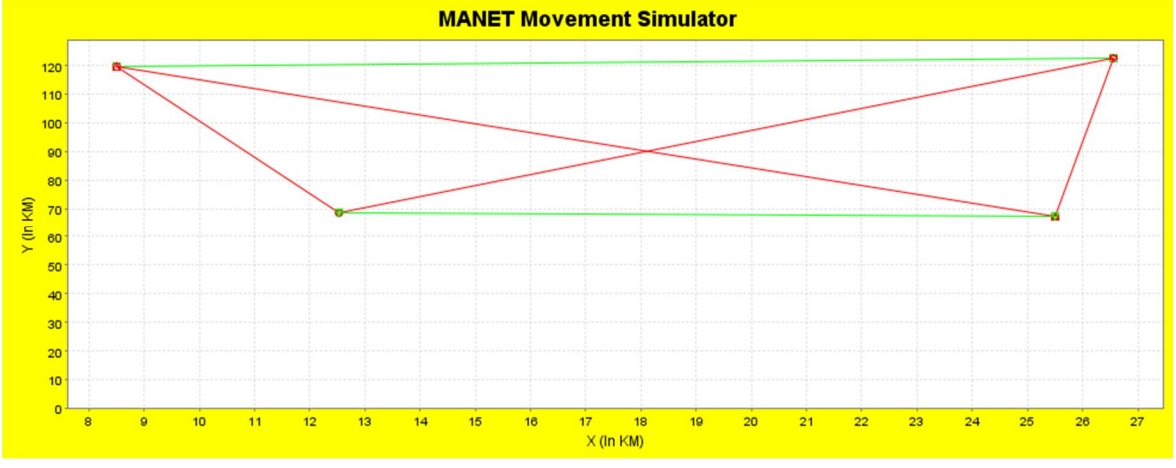


Figure 7. Simulation is run

Table 1: Time instance considered for each observation

t1	1st leader handover time
t2	2nd leader handover time
t3	3rd leader handover time
t4	4th leader handover time
t5	5th leader handover time
t6	6th leader handover time
t7	7th leader handover time

The average battery life of the network is calculated.
Let battery life of Node1 be b1 % at some particular time.
Let battery life of Node2 be b2 % at some particular time.
….
….
….
Let battery life of Node n be bn % at some particular time.

Average Battery Life of Network at some particular time

$= (b1 + b2 + \ldots + bn) / n$

$= \sum_{i=1}^{i=n} (bi / n)$

The simulation is run several times to get the Arithmetic Mean (A.M.) of this Average Battery Life at some particular time as discussed above. This A.M. is plotted against time intervals t1, t2, ... t7.
Note that time instance t_n (say) is different for all three algorithms. t_n denotes the time instance of the n^{th} leader handover time. In this comparison, time has not been measured quantitatively.

Table 2: A.M. of Average Battery Life at a particular Time Instant

	Static Weight Election Algorithm	Dynamic Weight Election Algorithm	Senior Static Weight Election Algorithm
t1	87.413	78.979	88.931
t2	78.001	45.421	80.663
t3	63.164	39.224	68.140
t4	53.380	27.229	60.482
t5	48.006	23.693	53.111
t6	29.774	23.685	30.597
t7	22.896	18.293	27.590

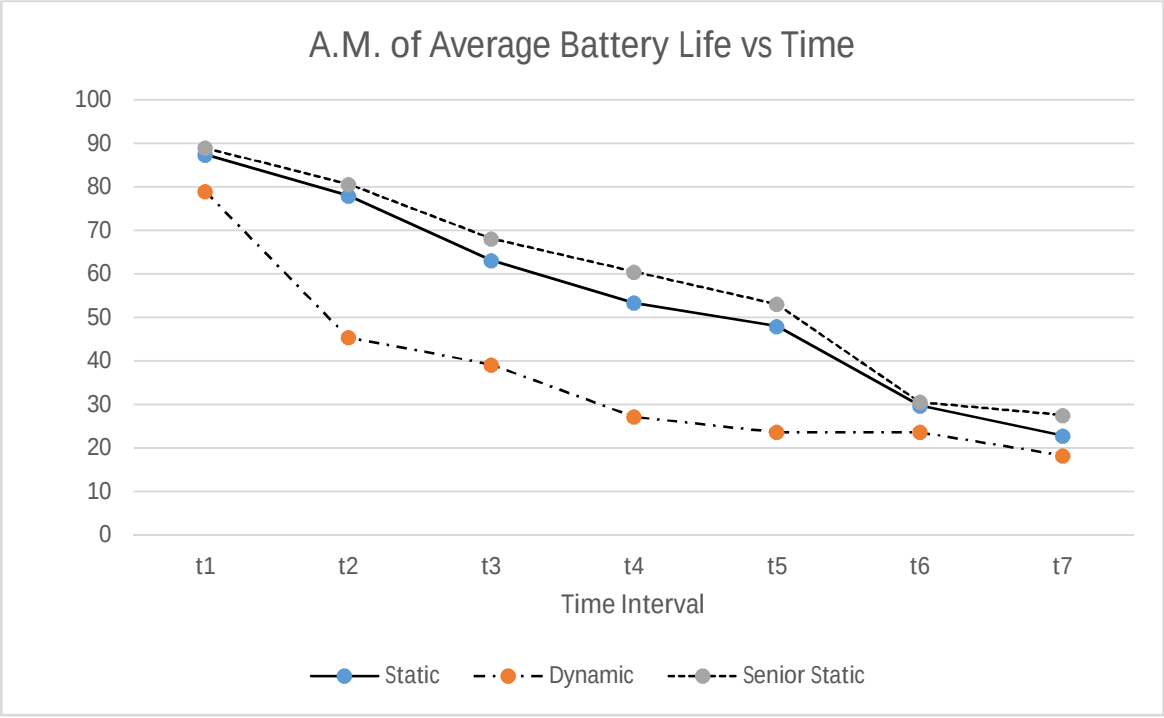


Figure 8. Graph showing battery drain against time

Some inferences that can be drawn from this comparison are:
The Dynamic Weight Leader Election Algorithm is most battery consuming. However, this algorithm takes into consideration the current node parameters, hence values are calculated in real time. Thus, this algorithm is more suitable in scenarios, where the accuracy of node parameters are desired. A trade off in battery life is made for greater accuracy.
The Static Weight Leader Election Algorithm and Senior Static Weight Leader Election Algorithm almost consistently give the same performance. However, at the later stages of simulation run-time i.e. - when battery of all nodes are critically low, Senior Static Weight Leader Election Algorithm gives better performance.
Both these approaches are using stale values for Node Weight calculation, thus accuracy of node parameters are low.

Which approach is the best, it is not possible to comment, as one algorithm provides better performance than the others in certain scenarios.

Conclusion

In Mobile ad-hoc Network (MANET), each and every node collectively work towards a common purpose. Thus, it is critical for all the nodes to remain connected to enable communication to continue. Therefore, network partitioning needs to be avoided at all cost. In this paper, we have proposed three algorithms to elect a leader node to facilitate network communication to continue. All three algorithms can elect a capable leader, which is clear from the fact that network remains connected throughout the simulation run-time. The elected leader controls the movement of all the other nodes. Two priority queue are maintained which hold the nodes which are going to become leader node in the future, when the current leader node steps down due to poor battery status. When necessary, the leadership transfer is made from the priority queue without re-election. This reduces message overhead, bandwidth overhead, election latency which in turn saves battery power of all the nodes. New nodes added to the network are also eligible to become leader of the network. Hence, we were successful in implementing a Leader Election Algorithm with lesser energy needs. Future works can address leader's load balancing and fault tolerance. Also, leader node election algorithm can be refined as the Node Weights are calculated by taking certain fixed percentage of Hop Count, Battery, Hop Strength values. It may be possible to find a better combination of network property percentage which gives better performance.

Acknowledgment

The author wishes to thank Dr. Abhijit Das, mentor and the co-author of this paper.

References

[1] A. Das, S. S. Basu, A. Chaudhuri, "An Autonomous-Leader Driven Adaptive Topology Management for MANET", American Journal of Advanced Scientific Research (AJASR), Vol. 3, Issue. 2, 2015, 231-245.

[2] A. Das, A. Rahman, S.S. Basu, A. Chaudhuri, "Energy Aware Topology Security Scheme for Mobile Ad Hoc Network", Proceedings of the 2011 International Conference on Communication, Computing & Security, ICCCS, Odisha, India, 2011, 114-118.

[3] Wireless Ad Hoc Network, https://en.wikipedia.org/wiki/Wireless_ad_hoc_network

[4] List of ad hoc routing protocols, https://en.wikipedia.org/wiki/List_of_ad_hoc_routing_protocols

[5] Internet Engineering Task Force (IETF) Mobile Ad Hoc Networks (MANET) Working Group Charter, www.ietf.org/html.charters/manet-charter.html

[6] H. A. Karimi, P. Krishnamurthy, "Real-Time Routing in Mobile Networks Using GPS and GIS Techniques", Proceedings of the 34th IEEE Hawaii International Conference on System Sciences, 2001, 1-11.

[7] H. Xu, X. Wu, H. Sadjadpour, J. Garcia-Luna-Aceves, "A Unified Analysis of Routing Protocols in MANETs", IEEE Transactions on Communications, vol. 58, March 2010, 911-922.

[8] A. Gill, C. Diwaker, "Comparitive Analysis of Routing in MANET", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 7, July 2012, 309-314.

[9] G. Mario, G. Pei, S. Lee, "Wireless, Mobile ad-hoc Network Routing", Computer Science Department, University of California, Los Angeles. Presented at IEEE/ACM FOCUS'99, New Brunswick, NJ, May 1999. [Online]. Available: www.cs.ucla.edu/NRL/wireless/PAPER/focus99.pdf

[10] M. Joa-Ng, I. T. Lu, "A Peer-to-Peer Zone-Based Two-Level Link State Routing for Mobile Ad Hoc Networks", IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks, Aug 1999, 1415-1425.

[11] C. K. Toh, "Long-lived Ad-Hoc Routing Based on the Concept of Associativity", IETF Draft, March 1999, 8 Pages. [Online]. Available: www.ietf.org/internet-drafts/draft-ietf-manetlonglived-adhoc-routing-00.txt

[12] R. Dube, et al., "Signal Stability Based Adaptive Routing for Ad Hoc Mobile Networks", IEEE Pers. Comm., February, 1997, 36-45. [Online]. Available: www.cs.umd.edu/projects/mcml/papers/pcm97.ps

[13] N. Meghanathan, "A MANET Multicast Routing Protocol for Stable Trees Based on the Inverse of Link Expiration Times", in Proceedings IEEE Consumer Communications and Networking Conference (CCNC 2012), Jackson, MS, USA, January 14-17, 2012, pp. 947-951.

[14] S. S. Basu, A. Chaudhuri, "Self-Adaptive Topology Management for Mobile Ad-hoc Network", Journal of The Institution of Engineers (India) Volume 84, July 2003, 7-13.

[15] S. S. Basu, A. Chaudhuri, "Self-Adaptive MANET: A Centralized Approach", Foundation of Computing and Decision Sciences, vol. 29, no. 4, 2004, 271-286.

[16] T. S. Rappaport, Wireless Communications: Principles and Practice. (Prentice Hall, Upper Saddle River, NJ), Oct 1995.

[17] T. Martin, D. Siewiorek, A. Smailagic, M. Bosworth, M. Ettus, J. Warren, "A Case Study of a System-Level Approach to Power-Aware Computing", ACM Transactions on Embedded Computing Systems, vol. 2, no. 3, August 2003, 255-276.

[18] Y. Afek, S. Kutten, M. Yung, "Local Detection for Global Self Stabilization", In Theoretical Computer Science, Vol 186, No. 1-2, 339, October 1997, 199-230.

[19] S. Dolev, A. Israeli, S. Moran, S. Toueg, "Uniform Dynamic Self-Stabilizing Leader Election Part 1: Complete Graph Protocols", Proceedings of 6th International Workshop on Distributed Algorithms, LNCS 579, 1992-1993, 167-180.

[20] D. Coore, R. Nagpal, R. Weiss, "Paradigms for Structure in an Amorphous Computer", Technical Report 1614, Massachusetts Institute of Technology Artificial Intelligence Laboratory, October 1997.

[21] C. Lin, M. Gerla, "Adaptive Clustering for Mobile Wireless Networks", IEEE journal on Selected Areas in Communications, 15(7), Sep 1997, 1265-1275.

[22] A. Das, S. S. Basu, A. Chaudhuri, "A Parameterised Leader Election Algorithm for MANET", International Journal of Electrical Systems and Control (IJESC), International Sciences Press, Vol.2 No.1, June 2010, 1-6.

[23] S. S. Basu, A. Das, A. Chaudhuri, "An Extensible Movement Simulator for Mobile Ad hoc Network", Proceedings of the First International Conference on Emerging Applications of Information Technology (EAIT 2006), Computer Society of India, Science City, Kolkata, India, ELSEVIER Publication, February 10-11, 2006, 231-234.