Leader Election Algorithm using Fibonacci Heap Structure in Mobile Ad hoc Network

Kshama Tiwari
Department of Computer Science and Technology
UPTU, LUCKNOW

Brajesh Kumar Umrao
Department of Computer Science & Technology

ABSTRACT
Mobile Ad hoc network is a self-configured network of devices connected using a wireless medium. Ad hoc network is a temporary network connection created for a specific purpose. MANET can be seen as a distributed computing environment, where Leader Election mechanism is used, for the purpose of synchronization. Election algorithms are used to find the leader for Distributed System. Various election algorithms are already proposed for electing a leader. In this paper, the main challenge is to find the new leader in lesser time with minimum number of message communication. In this paper, an algorithm is proposed to find the leader in lesser amount of time and messages through the use of Fibonacci Heap structure. The better time complexity of operations using Fibonacci heap structure makes it suitable for the leader election in Mobile Ad Hoc Network, as compared to other tree structures.

Keywords
Mobile Ad hoc Network, Fibonacci Heap Structure, Leader Election

1. INTRODUCTION
An Election Algorithm is a procedure which elects a leader to coordinate and synchronize tasks in distributed computing system as the nodes can request for resource and services simultaneously under MANET. The challenge is to have a right election algorithm that chooses the right leader based on various factors in MANET. In case of MANET, if any node changes its position (by changing the network), or failed, that was working as a coordinator previously, now it is required to find the new leader for uninterrupted functioning of network. So, in MANET an election algorithm initiates a process to find a new leader.

Election algorithm is used in MANET because it is highly adaptive to topological changes. So an efficient algorithm is proposed with effective data structure to form a network for the purpose of minimizing number of message passing and time complexity. Moreover, Here an approach will be proposed with the use of Fibonacci heap structure to handle the departure and failure of nodes in MANET for election of leader.

This paper is organized as follows: Section II describes the literature review of leader election, MANET & Fibonacci Heap. Section III provides a brief layout of leader election using Fibonacci Heap. Section IV presents Fibonacci heap under MANET. Section V gives the performance analysis under different factors such as time complexity and number of messages etc. Finally, in section VI conclusion & future work is concluded.

2. LITERATURE REVIEW
2.1 MANET
A mobile ad hoc network (MANET) is a temporary self-organizing network of wireless mobile nodes without the support of any existing infrastructure. In order to maintain connectivity in a mobile ad hoc network all participating nodes have to perform routing of network traffic. In MANET, each mobile node acts as a specialized router. To perform synchronization among devices, a specific leader is needed that coordinates between devices, and interconnection of topology with other mobile nodes within the network. Current deployment of ad hoc networks is confined to Mobile Ad Hoc Networks and Wireless Local Area Networks.

MANET does not have a fixed infrastructure due to the dynamic nature of the nodes; leader election becomes an extremely important issue. The other challenges of MANET are scarcity of resources, energy and limited bandwidth of the wireless links, make the transmission complex. Due to this dynamic nature as the nodes change their position, or decreases its battery life, the leader of network may also require to change, if the selection based on some dynamic property of node like battery life.

2.2 Leader Election
Leader election is a process of electing new leader in a distributed network for the purpose of coordination and synchronization. Bully algorithm [2] is one proposed by Garcia-Molina for the coordinator election. This algorithm works on the assumption that the system uses time as a parameter to detect process failure (the coordinator). All processes have a unique number in the system and they know the process number of all other processes. Once an election is held, a process with the highest process number is elected as a coordinator, which is agreed by other nodes (process). [2]
The Bully Algorithm has three types of messages: first is *election message* sent to announce an election, second is an *ok* message sent as the response to an election message, and third is *victory messages* sent to announce the new coordinator among all other alive processes. When any node finds that the leader node has been crashed, then it immediately announces itself as a new leader if no other higher process id node exists. Otherwise it starts election process as bully algorithm had proposed. [3]

Various leader election approaches[specify few references of the other algorithm] are already defined on the basis of various criteria like highest id of nodes, by selecting set of k nodes with highest priority at once, by sending acknowledgment etc. Here the Leader election will be based on some dynamic properties of nodes, and the network for which it used will be MANET.

2.3 Fibonacci Heap

Fibonacci Heap is tree based data structure, which is a set of Heap ordered trees and maintains a min heap pointer. It is advancement over existing Binomial heap. That gives the facility of using the binomial heap concept with some better features regarding time complexities in amortized time.

It is a type of heap that supports merging, insertion, decrease-key* and find_min in O (1) amortized time, and delete and delete_min in O (log n) amortized time. [8]

Table 1: Comparison- Time Complexity

Heap Tree	Make heap	Insert	Delete	Decrease
Binomial Heap	1	log n	log n	log n
Fibonacci Heap	1	1	log n	1

Fibonacci heap provides better time complexity than other various tree data structures like Binary Search Tree, Heap Tree, and Binomial Heap etc. Table 1 represents a summary of time complexity based comparison between Binomial Heap and Fibonacci Heap structures. This table represents that Fibonacci heap has better complexity in insert and delete operations as compared to the Binomial Heap structures.

3. FIBONACCI HEAP

3.1 Definition

Fibonacci heap is more efficient data structure as compared to binomial heap data structure as following.

- Fibonacci heap performs merging operation in faster way, as binary heap cannot support in less than Θ (n) time.
- In Fibonacci heap, insert operation that takes O (1) amortized time, whereas in binomial heap O (log n) time is required for the same operation.
- -. A decrease-key operation takes O (1) amortized time in Fibonacci heap, and O (log n) time as with a binomial heap for the same operation

A Fibonacci Heap Structure is a collection of a forest of trees. A given Fibonacci heap *H* is accessed by a pointer *min* [*H*] to the root of the tree containing a minimum key; this node is called the *minimum node* of the Fibonacci heap and this will be elected as the leader node. The pointer *min* [*H]* thus points to the node in the root list whose key is minimum. If a Fibonacci heap *H* is empty, then *min* [*H*] = NIL. The roots of all the trees in a Fibonacci heap are linked together using their *left* and *right* pointers into a circular, doubly linked list called the *root list* of the Fibonacci heap. The siblings at each level are also connected in such manner. The order of the trees within a root list is arbitrary. The advantage of circular, doubly linked list used in Fibonacci heap is that first a node can be removed from a circular, doubly linked list in *O*(1) time and second, given two such lists, can concatenate them (or "splice" them together) into one circular, doubly linked list in *O*(1) time. So, it is an approach to exploit this property of Fibonacci heap.

Node that joins the group records the information about its parent, child, left and right sibling. The nodes need not possess complete information regarding the other nodes of the graph. The values in the nodes (taken as UID) satisfy the heap property. The property of the min-heap is that for every node i, H [PARENT (i)] < A[i], i.e, the value of a node should always be greater than the value of its parent. Thus, the smallest element in a Fibonacci heap is stored at the root, and the sub-tree rooted at a node contains values no smaller than that contained at the node itself. Hence, the root is always the smallest element. [9]

3.2 Algorithm

Here an algorithm for the Fibonacci heap operations is defined. In the below algorithm, only those functions are defined which have proposed some improvement over binomial heap. Rest of the function may be called inside the given functions, are assumed to be already defined.

Make_Fibo_Heap ()

n [H]:= 0

min [H]:= NIL

return H

Fibo_Heap_Min (H)

return min [H]

Fibo_Heap_Link (H, y, x)

remove y from the root list of H

make y a child of x

degree[x]:= degree[x] + 1

mark[y]:= FALSE

Fibo_Heap_Insert (H, x)

degree[x]:= 0

p[x]:= NIL

child[x]:= NIL

left[x]:= x

right[x]:= x

mark[x]:= FALSE

concatenate the root list containing x with root list H

if min [H] = NIL or key[x] <key [min [H]]

then min [H]:= x

n [H]:= n [H] +1

Fibo_Heap_Extract_Min (H)

z: = min [H]

if x <> NIL

then for each child x of z

do add x to the root list of H

p[x]:= NIL

remove z from the root list of H

if z = right[z]

then min [H]:=NIL

else min [H]:=right[z]

CONSOLIDATE (H)

n [H]:= n [H]-1

return z

Fibo_Heap_Decrease_Key (H,x,k)

if k > key[x]

then error "new key is greater than current key"

key[x] := k

y := p[x]

if y <> NIL and key[x]<key[y]

then CUT(H, x, y)

CASCADING-CUT(H,y)

if key[x]<key[min[H]]

then min[H] := x

Fibo_Heap_Delete(H,x)

Fibonacci-Heap-Decrease-Key(H,x,-infinity)

Fibonacci-Heap-Extract-Min(H)

4. EXAMPLE OF MANET

Here a diagram is shown in figure 1 to give a view of Fibonacci heap as MANET .A forest of tress are available where each tree has the minimum key at the root node.

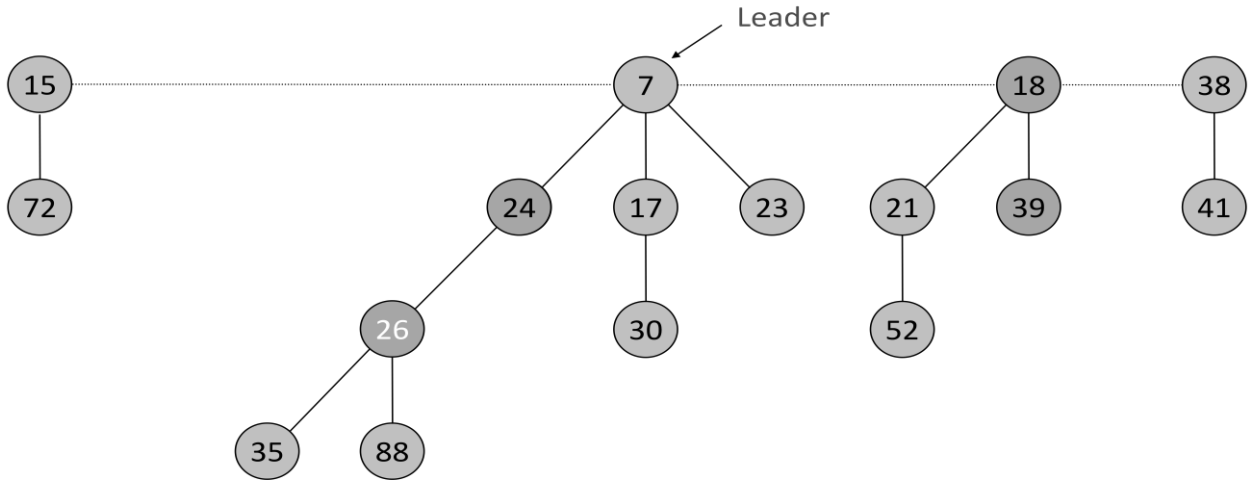


Figure 1: A View of Mobile ad hoc Network With one Election Leader [web resource]

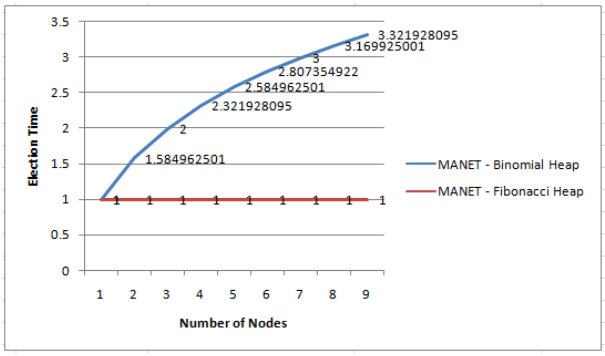
5. PROPOSED WORK

5.1 Using Fibonacci Heap in MANET

Fibonacci Heap data structure can be efficiently used for the design or visualization of MANET. As the MANET is collection of self stabilized multiple networks. Fibonacci heap also the collection of Heap Ordered tress. That creates a similarity between MANET structure and Fibonacci Heap.

In MANET, it is also required to find out the leader for the management of functioning based on synchronization. In MANET leader becomes changes dynamically, as they have the features of moving from one network to another frequently. Due to this the new leader is also required to be elected dynamically and in less time. As it is discussed above the nodes have some special identification, here consumed battery life of devices will be taken as their election criteria for becoming a leader. As the battery life changes dynamically, so may the existing leader becomes shut down as being discharged. A new leader is required to elected immediately within minimum time.

In Fibonacci heap, a doubly circular linked list is maintained that contains min value key nodes as root. So when a leader with min key is failed or discharged, the next min key node can be found immediately in amortized time of O(1).



Graph 1: Comparison between MANET-binomial heap and MANET-Fibonacci heap with respect to number of nodes and election time.

The above graph is representing the performance of electing a leader using the concept of Binomial and Fibonacci Heap, where Fibonacci Heap wins the race based on the time consumption in decreasing key during operation during election of leader which is O(1).

As the battery life is a dynamic criterion, it has taken as the key value for the nodes in network. The Tree is required to change their key node values, which is treated as the operation Decrease key operation in Fibonacci Heap and it takes O(1) amortized time and tree gets dynamically updated. This feature is not available in binomial heap.

6. CONCLUSION AND FUTURE WORK

In this paper, a novel approach is proposed for the election of leader in MANET using Fibonacci Heap. Fibonacci heap results good by their advanced time complexities. The design of Fibonacci heap suits on wireless networks, and especially on Mobile Ad Hoc Networks. Where network changes dynamically, and Fibonacci heap give the efficient way to change the structure in amortized time. In future enhancement in the working of ring algorithm of leader election will be considered , by discarding the simultaneous elections. Trie data structure can also be explored to implement leader election for MANET. As well as also try to upgrade the Ring Algorithm of Election leader, by removing the duplicate elections initiated by nodes in parallel.

7. REFERENCES

[1] Sinha, P.K. Distributed Operating Systems Concepts and Design; Prentice-Hall: Upper SaddleRiver, NJ, USA, 2002; pp. 332–334.

[2] Tiwari, Kshama, & Umbrae Brajesh Comparative analysis of various leader election approaches, JACOTEH, 2015

[3] Garcia-Molina, H. Elections in a distributed computing system. IEEE Trans. Comput. 1982, C-13, 48-59.

[4] Mamun, Q.E.K.; Masum, S.M.; Mustafa, M.A.R. Modified Bully Algorithm for Electing Coordinator in Distributed Systems. In Proceedings of the 3rd WSEAS International Conference on Software Engineering, Parallel and Distributed Systems, Salzburg, Austria, 13–15 February 2004.

[5] S. Vasudevan, J. Kurose, and D. Towsley, "Design and Analysis of a Leader Election Algorithm for Mobile Ad Hoc Networks," ICNP'04, 2004.

[6] G. Singh, "Leader Election in the Presence of Link Failures," IEEE Transactions on Paralle and Distributed Systems, vol. 7, no. 3, March 1996..

[7] P. Basu, N. Khan and T. Little. A Mobility based metric for clustering in mobile ad hoc networks. In *international Workshop on Wireless Networks and Mobile Computing*, April 2001.

[8] Sepehri M., Goodarzi M., ³Leader Election Algorithm Using Heap Structure´,12th WSEAS International Conference on Computers, Heraklion, Greece, July 25, 2008

[9] Jain, Arihant Kumar, Sharma Ramshanhar, "Leader Election Algorithms in Wireless Environments using Fibonacci Heap Structure", May-June 2102.

[10] Spector, A. z. 1989. Achieving application requirements. In Distributed Systems, S. Mullender,