Topology Management in Self- Adaptive MANET: A Distributed Approach

Samir Biswas, Jishan Mehedi and M.K.Naskar

Advanced Digital and Embedded Systems Lab,
Department of Electronics & Telecommunication Engineering, Jadavpur University, Kolkata – 700032
E-mail : samir_etceju@yahoo.com, jmehedi2007@yahoo.co.in, mrinalnaskar@yahoo.co.in

ABSTRACT

This paper proposes a distributed algorithm for the maintenance of topology of a network in a MANET (Mobile Ad-hoc Network). In this algorithm, it is assumed that every node possesses GPS receiver. Nodes receive the position and velocity information of their neighbors at the end of each beacon interval through GPS. Next, these nodes check whether a certain condition is violated or not. A node changes its velocity if the condition is not satisfied; otherwise it tries to keep the velocity constant. In this way each node modifies its velocity for the next beacon interval. The simulation run of the algorithm is carried out on a number of synthetically generated network scenarios and results thus obtained show the effectiveness of the proposed algorithm.

Keywords: *Mobile Ad-Hoc Network, Topology Management, Distributed Algorithm.*

1. INTRODUCTION

A Mobile Ad-Hoc Network (MANET) is a group of wireless autonomous mobile nodes forming an environment to communicate with each other dynamically without any fixed infrastructure or administration [1]. In such a network, the nodes can move in any arbitrary manner without any prediction. Each node acts as a usual trans-receiver, which can also find the network routes to aid the network to form a complete connected graph.

As a result, the routing and topology management has become an important issue in a MANET. Many researchers have developed efficient routing protocols which ensure to find the exact route to connect the transmitting mobile node to its intended caller, may be outside the transmitting range of the transmitter, via other node(s) without having much delay and unnecessary control overhead.

Existing routing protocols for MANET can be classified into four different basic categories namely flooding, proactive routing, reactive routing and dynamic cluster based routing [2]. However none of these routing schemes guarantees constant network connectivity during the movement and each of these schemes have constant route maintenance overhead. A particular node may even be disconnected in the worst case.

Centralized topology management schemes in [3, 4, and 5] discuss a self-adaptive movement control algorithm, which ensures the retention of network connectivity even during the positional variation of the nodes. But in this case, a coordinator has to be elected and all other nodes should follow the instructions from the coordinator. The main disadvantages of the centralized topology management scheme are increase in control overhead and non-scalability. Once the coordinator

fails to perform, the whole network becomes non-functional.

In this paper, we have suggested a distributed topology management algorithm where all the nodes of the network will remain connected throughout the operation so that the nodes can follow a fixed routing, which eliminates the routing overhead. Each node is provided with a GPS receiver that receives its position as well as the velocity (in both magnitude and direction) information. All the nodes can separately calculate its own velocity on the basis of the information received from its neighbors for the next beacon interval in such a way that the overall topology of the network remains same.

The paper is organized as follows. Next section provides the formal definition of the topology management problem. In the third section we present the proposed distributed algorithm for maintaining the topology. Simulation results are presented in the fourth section. The next section presents a comparative study between the proposed algorithm and the centralized algorithm in [3]. We finally conclude the paper in section six. The appendix contains the proof of lemma 1 and lemma 2.

2. TOPOLOGY MANAGEMENT PROBLEM

Given the physical topology of a mobile ad-hoc network, the problem is to control the movements of the individual nodes so as to maintain a stable neighborhood topology. The objective is to allow the nodes to communicate amongst themselves without the need of any variable routing protocols.

Let us consider a MANET consisting of N number of nodes n_0, n_1, n_2 …. n_{N-1}. We assume that each node has a maximum transmission range of R_{max}. Now, any two nodes n_i and n_j are called neighboring nodes if they can communicate amongst themselves without the need of any routing. So any two nodes n_i and n_j will be neighbors if the distance between them D(i,j) = $R_{max.}$ The network topology will be maintained if D(i.j) = Rmax for any two neighboring nodes n_i n_j at any time t.

3. THE PROPOSED ALGORITHM

3.1 Assumptions

The algorithm is based on following assumptions:

(1) All nodes are enabled with GPS receivers. These receivers can furnish the current position and velocity information of an individual node.

(2) All the nodes have a predefined maximum velocity, V_{max}.

(3) Acceleration and deceleration of the nodes are taken to be instantaneous.

(4) It is assumed that if the nodes are within the appropriate range, the instruction message will never be lost in transit.

(5) At the beginning, all the nodes have a configuration that creates a connected topology.

(6) Each node has a unique identification number.

3.2 Neighborhood selection algorithm

Let, R_{max} be the maximum range of message communication of each individual node. Then at the beginning a node selects as its neighbor, all nodes which are at a distance less than R_{max} from that node. Next through Hello packets it sends its current position, velocity and node identification number to all its neighboring nodes and also receives the same from its neighbors. A node stores position and velocity information and identification number of its neighbors. This concludes the neighborhood selection procedure.

3.3 Movement Algorithm

Each node sends its current position and velocity information to all its neighboring nodes with the help of hello packets after a fixed duration T. This time interval T is defined as the “Beacon Interval”. A node after receiving the position and velocity information of all its neighbors resolves the velocity along two mutually perpendicular directions; say the X-axis and the Y-axis.

At the beginning, each node is assigned a velocity at random. Then in each subsequent beacon interval

it calculates the current distance from its neighbors and also predicts the new distance from its neighbors after the end of the beacon interval based on the current velocities of the neighbors. It then checks whether the predicted distance along any axes exceeds a predefined threshold distance R_{th} or if the distance becomes less than zero i.e. the nodes cross each other. We call these two cases as the violation conditions. In both the cases the node adjusts its velocity suitably so that these two conditions are not violated. If the predicted distance remains greater than zero and less than R_{th} then the node moves with its previous velocity. We now analyze this algorithm in more details.

For simplicity we assume all the nodes are moving along the positive X axis. For a particular node say A its neighbor is either in front of it (say B) or behind it (say C). We separately treat these two cases.

3.3.1 Case I: For a node in front

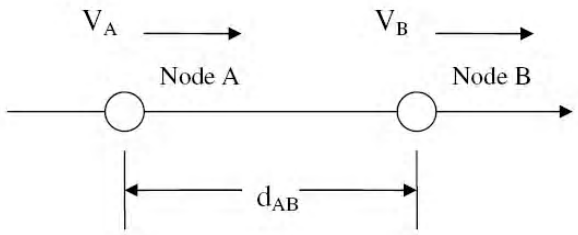

Fig. 1: Illustrating Case I of Movement Algorithm

Let, the current velocity of node A be V_A and that of node B be V_B; the current distance between them be $d_{AB}=d_B-d_A$; and the threshold distance be R_{th}. Choice of R_{th} is given in lemma 1 in appendix. If T be the beacon interval, the new distance between them after time T is given by,

$$d_{AB}|_{New} = d_{AB} + (V_B-V_A)*T$$

Now two conditions may arise when appropriate action must be taken. They are $d_{AB}|_{New}<0$ or $d_{AB}|_{New} > R_{th}$ according as $V_A>V_B$ or $V_A<V_B$. In both the case node A adjusts its velocity to $V_A|_{New}$ so that the distance between them returns to a specific distance R after the time T where $0<R< R_{th}$. The choice of this distance R is given in lemma 2 in appendix. And we have

$$R = d_{AB} + (V_B–V_A|_{New})*T$$

$$\text{Or,} \quad V_A|_{New} = V_B + (d_{AB} – R)/T \qquad ...(1)$$

3.3.2 Case II: For a node which is behind

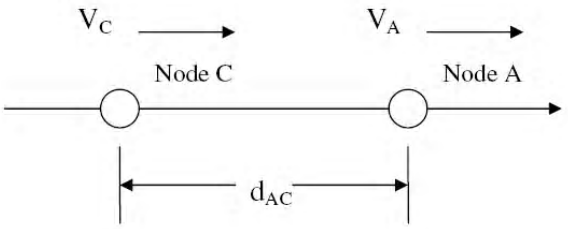

Fig. 2: Illustrating Case II of Movement Algorithm

Let the current velocity of node A be V_A and that of node C be V_C; the current distance between them be $d_{AC}=d_A-d_C$. Now, the new distance between them after time T is given by

$$d_{AC}|_{New} = d_{AC} + (V_A–V_C)*T$$

As before two cases may arise $d_{AC}|_{New} > R_{th}$ and $d_{AC}|_{New} < 0$ when node A must change its velocity to bring the distance between them to a stable distance R where $0<R<R_{th}$ (chosen according to lemma 2). Let the new velocity be $V_A|_{New.}$ Therefore,

$$R = d_{AC} + (V_A|_{New} – V_C)*T$$

$$\text{Or,} \quad V_A|_{New} = V_C + (R – d_{AC})/T \qquad ...(2)$$

If the new predicted distance between the nodes is greater than zero and less than R_{th} then the node keeps its velocity intact. In this way a node calculates its velocity for all its neighbors for the next beacon interval. The final velocity for the next beacon interval is chosen as follows.

We assign a weight W_i to the velocity calculated on the basis of information received from the i^{th} node as,

$$W_i = (d_{New}\text{ (predicted)} – R_{th}) + 1 \quad \text{for } d_{New}\text{ (predicted)} > R_{th}$$
$$= |d_{New}\text{ (predicted)}| + 1 \quad \text{for } d_{New}\text{ (predicted)} < 0$$
$$= 1 \quad \text{otherwise.}$$

where d_{New} (predicted) is the predicted new distance between the nodes after time T.

Now, the node calculates its velocity for the next interval as $V=\Sigma W_i V_i / \Sigma W_i$. Where V_i is the velocity calculated for the i^{th} neighbor and W_i is its

corresponding weight. We note that by choosing velocity in this way the velocity calculated for the nodes for which violation occurs gets greater weight and hence greater precedence than the cases in which no violation occurs. If all the nodes choose their velocity according to the above algorithm the network is able to maintain a fixed topology.

3.3.3 Algorithm Steps

Algorithm : Topology management of a MANET

Input : A MANET comprising of M number of nodes.

Output: A MANET which has a fixed topology.

For the i^{th} node

Initialization:

Vx(i) ← Velocity of the i^{th} node along X axis;

Vy(i) ← Velocity of the i^{th} node along Y axis;

X(i) ← Position of i^{th} node along X axis;

Y(i) ← Position of i^{th} node along Y axis;

N ← Number of neighbors of the i^{th} node;

Receive current position and velocity of all neighbors;

For j:=1 to N do

Calculate relative distance of the j^{th} neighbor

dx=X(j)-X(i);dy=Y(j)-Y(i);

If dx>=0

Calculate dx_new=dx+(V(j)-V(i))*T;

If dx_new>=Rth

If Vx(j)<V_{max}/2

Vx_predicted (j)=V_{max}/2;

Else

Choose Vx_predicted (j) at random between Vx(j) and V_{max};

End if;

Weight_x(j)=dx_new-Rth+1;

Else if dx_new<0

If Vx(j)>V_{max}/2

Vx_predicted (j)=V_{max}/2;

Else

Choose Vx_predicted (j) randomly between 0 and Vx(j);

End if;

Weight_x(j)=|dx_new|+1;

Else

Vx_predicted (j)=Vx(i);

Weight_x(j)=1;

End if;

Else

dx_new=|dx|+ (V(i)-V(j))*T;

If dx_new>Rth

If Vx(j)>V_{max}/2

Vx_predicted (j)=V_{max}/2;

Else

Choose Vx_predicted (j) randomly between 0 and Vx(j);

End if;

Weight_x(j)=dx_new-Rth+1;

Else if dx_new<0

If Vx(j)<V_{max}/2

Vx_predicted (j)=V_{max}/2;

Else

Choose Vx_predicted (j) at random between Vx(j) and V_{max} ;

End if;

Weight_x(j)=|dx_new|+1;

Else

Vx_predicted (j)=Vx(i);

Weight_x(j)=1;

End if;

End if;

Similarly calculate Vy_predicted (j) and Weight_y(j) for the j^{th} node along Y axis;

End For;

Calculate

$$Vx(i) = \frac{\Sigma \text{ Weight\_ x(j)} * \text{Vx\_ predicted (j)}}{\Sigma \text{ Weight\_ x(j)};}$$

[for j = 1 to N]

Similarly calculate Vy(i);

End algorithm: Topology management of a MANET

4. SIMULATION RESULTS

The proposed algorithm was simulated on a synthetically designed environment and encouraging results were obtained. The simulation was done using Matlab software in Windows environment. For our simulation we considered five nodes with random initial velocities along both X and Y axis. The maximum communication range R_{max} was chosen as 70 km. Therefore, the threshold distance R_{th} was chosen according to lemma 1, $R_{th}=R_{max}/v2 - 5 = 44.5$ Km. Also the maximum velocity of the nodes were taken as V_{max} = 80 Km/hr. The beacon interval was taken as 6 minutes.

The initial positions and velocities of the nodes were as follows.

Table 1: Initial Position and Velocity of nodes

Node	Initial Position	Initial Velocity	Neighboring Nodes
1	(0,0)	$35\hat{a}_x + 10\hat{a}_y$	2,3,4,5
2	(40,0)	$25\hat{a}_x + 35\hat{a}_y$	1,4,5
3	(-40,0)	$35\hat{a}_x + 25\hat{a}_y$	1,4,5
4	(0,40)	$45\hat{a}_x + 15\hat{a}_y$	1,2,3
5	(0,-40)	$15\hat{a}_x + 45\hat{a}_y$	1,2,3

The simulation was carried out for an interval of 60 hours=3600 minutes. The results obtained are shown in Fig. 3 to Fig. 6.

From figure 3 and 4 (showing the initial and final topology of the network) we can see that the nodes calculate their velocity for each beacon interval in such a way that the topology of the overall network remains the same. Figures 5 shows how the distances of the neighbors vary for node 1. Similar graphs were also obtained for other nodes. From the graphs it is

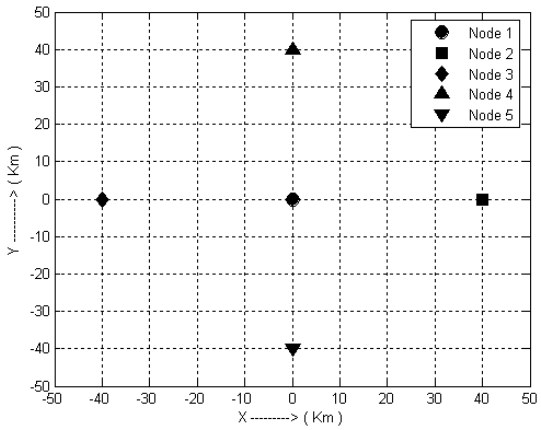

Fig. 3: Initial topology of the network

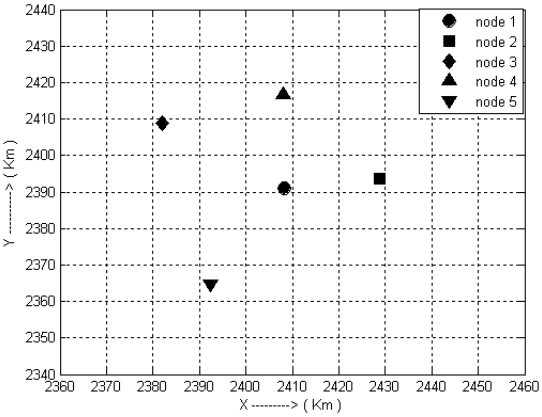

Fig. 4: Final topology of the network

clear that the distance between the neighboring nodes never exceed the maximum communication range R_{max} = 70 Km. this clearly shows the effectiveness of the algorithm in maintaining the topology of a network.

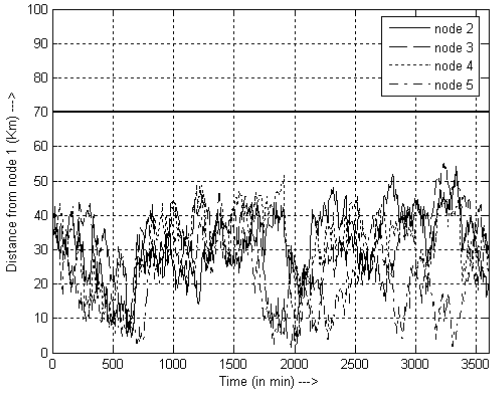

Fig. 6: Distance of the neighbor nodes from node 1

5. PERFORMANCE COMPARISON

We also perform a comparative study between the distributed topology management algorithm presented in this paper and the centralized topology management algorithm. From the results as presented above we see that the distributed algorithm is as effective as the centralized one in maintaining the topology of a mobile ad-hoc network. However, the distributed algorithm is better in the sense that it completely eliminates the control overhead of the coordinator, which was present in the centralized approach. In the centralized topology management algorithm presented in the coordinator had to issue appropriate commands to the nodes in order to maintain the network topology and to ensure that a node always remains in contact with the coordinator. But this greatly increases the control overhead of the coordinator, and hence that of the network. In distributed algorithm we have shown how a node can suitably adjust its velocity so that the topology of the network always remains the same and at the same time the neighboring nodes always remain within the communication range of each other. Thus, the control overhead of a node is reduced to just sending hello messages containing its current position and velocity, as received by their GPS receivers, to its neighboring nodes, after each beacon interval T. Further the centralized algorithm maintained the topology when the nodes possessed velocities only in one direction, but the distributed algorithm is able to maintain the topology even when the nodes posses velocities along any direction. Thus, by eliminating routing overhead by topology management and control overhead by distributed algorithm a highly effective protocol for mobile Ad-hoc network may be realized.

6. CONCLUSION

In this paper, we have presented a distributive algorithm for mobile nodes in a MANET to maintain the network topology. This algorithm may also be applied even when all the nodes are not moving in the same direction. Due to the application of a distributed scheme, the control overhead is reduced as compared to a centralized approach and the topology is not vulnerable if one of the nodes becomes non-functional, as there is no concept of central coordinator. The simulation results show the effectiveness of the algorithm. Presently, we are working on a distributive scheme where the nodes will have more flexibility in choosing its velocity and still maintain the network topology.

REFERENCES

1. National Institute of Standard and Technology, "Mobile Ad-hoc Networks (MANETs)", *http:// w3.antd.nist.gov/wahn_mahn.shtml.*
2. Elizabeth M.Royer and Chai-Keong Toh, "A Review of current routing protocols for Ad-Hoc Mobile Networks", *IEEE Personal Communications, Vol. 6, No. 2, pp. 46-55, April 1999.*
3. S.Samanta, S.S.Ray, S.SenGupta, M.K.Naskar, "A Novel Algorithm for Managing Network Configuration", Asian International Mobile Computing Conference 2006, Kolkata, Jan 04-07, pp. 51-58.
4. S.S.Basu and A.Chaudhari, "Self-adaptive Topology Management for Mobile Ad- hoc Network", *IE(I) Journal-ET, Vol.84, July 2003.*
5. Soumya Sankar Basu, Atal Chaudhari, "Self-Adaptive MANET: A CentralizedApproach", *Foundations of Computing and Decision Sciences, Vol.29, 2004.*

APPENDIX

Lemma 1: Choice of threshold distance R_{th}.

If R_{max} be the maximum range of communication, the threshold distance chosen for this algorithm to ensure that the distance between the neighboring nodes does not exceed R_{max} is $R_{th}<R_{max}/v2$.

Proof: In this algorithm the velocities of the nodes are resolved along two mutually perpendicular directions X and Y axes and the algorithm ensures that the distance between any two nodes doesn't exceed the threshold distance R_{th} along any of these axes.

Now, let d_x and d_y be the distance of a node from its neighbor along X axis and Y axis respectively. Therefore the absolute distance between the nodes

$d=v(d_x^2+d_y^2)$.

Our algorithm ensures that $0<d_x<$Rth and $0<d_y<R_{th}$. Therefore, the maximum absolute distance between any two nodes occur when $d_x=d_y=R_{th}$ for which $d_{max}=v2*R_{th}$. This distance d_{max} must be less than the maximum communication range R_{max}. Therefore

$d_{max} < R_{max}$

Or $v2*R_{th} < R_{max}$

Or $R_{th} < R_{max}/v2$

Lemma 2: Selection of the stable distance R and new velocity V_{New}.

If V_{max} be the maximum velocity of the nodes, T be the beacon interval and d be the current distance between any two nodes, then in case the predicted distance d_{New} becomes $> R_{th}$ or < 0 the stable distance R to which the node must return by changing its velocity must be such that the new velocity V_{New} becomes, for a node B in front and d_{New} (predicted) $> R_{th}$ or for a node B in back and d_{New}(predicted) < 0

$V_{New}=V_{max}/2$; when $V_B=V_{max}/2$

$V_B<V_{New}<V_{max}$; $V_B>V_{max}/2$

for a node B in front and d_{New} (predicted) < 0 or for a node B in back and d_{New}(predicted) $> R_{th}$

$V_{New}=V_{max}/2$; when $V_B=V_{max}/2$

$0<V_{New}<V_B$; $V_B<V_{max}/2$

Proof: Let us consider the situation shown in figure 1. Let V_A and V_B be the current velocities of the nodes and d_{AB} be the distance between them. Therefore the distance between them after time T as predicted by both the nodes

$d_{AB}|_{New}$ (predicted) $= d_{AB} + (V_B-V_A)*T$

Let there be a violation condition for which both the nodes must alter its velocity i.e. either $d_{AB}|_{New}$ (predicted) <0 or $d_{AB}|_{New}$ (predicted) $>R_{th}$. Let node A changes its velocity to $V_A|_{New}$ and node B changes its velocity to $V_B|_{New}$, which are given by

$V_A|_{New} = V_B + (d_{AB} - R_1)/T$

$V_B|_{New} = V_A + (R_2 - d_{AB})/T$ [from (1) and (2)]

If V_{max} be the maximum velocity of the nodes then we must have $0< V_A|_{New} <V_{max}$ and $0< V_B|_{New} <V_{max}$ from which we have the following conditions.

$d_{AB} + V_B*T - V_{max}*T < R_1 < d_{AB} + V_B*T$ ...(3)

and $d_{AB} - V_A*T < R_2 < d_{AB} - V_A*T + V_{max}*T$ ... (4)

The actual final distance between the two nodes,

$d_{AB}|_{New}$ (actual) $= d_{AB} + (V_B|_{New} - V_A|_{New})*T$

$= R_1+R_2 - (d_{AB} + (V_B-V_A)*T)$

Now two cases may arise,

Case 1: If $d_{AB}|_{New}$ (predicted) $> R_{th}$

In this case node A increase its velocity and node B decrease its velocity. So we must ensure $d_{AB}|_{New}$ (actual) doesn't become less than 0. Hence $d_{AB}|_{New}$ (actual) > 0 which gives,

$R_1+R_2 > (d_{AB}+(V_B-V_A)*T)$ ...(5)

Case 2: If $d_{AB}|_{New}$ (predicted) < 0

In this case node A decrease its velocity and node B increase its velocity. So we must ensure $d_{AB}|_{New}$ (actual) doesn't become greater than R_{th}. Hence $d_{AB}|_{New}$ (actual) $< R_{th}$ which gives,

$R_1+R_2 < (d_{AB} + (V_B-V_A)*T) + R_{th}$ ...(6)

Hence we must choose $R_1$and R_2 such that inequalities (3), (4), (5) and (6) are simultaneously satisfied. From (3) and (4) if we choose $R_1 = d_{AB} + V_B*T - V_{max}*T/2$ and $R_2 = d_{AB} - V_A*T + V_{max}*T/2$ i.e.

we take the mean of the two extreme values we see that $R_1+R_2 = 2d_{AB}+ (V_B-V_A)*T$ which satisfies both inequalities (5) and (6). Hence ideally the stable distance R must be chosen as

$R=d + V_B*T - V_{max}*T/2$ for a node B in front.

$R=d- V_B*T + V_{max}*T/2$ for a node B which is behind.

For both the cases $V_{New}=V_{max}/2$ (from (1) and (2)). But, again for the network to be stable we would want that when d_{New} (predicted)$>R_{th}$, $V_{New}>V_B$ when node B is in front and $V_{New}<V_B$ when node B is behind, and when d_{New} (predicted)<0, $V_{New}<V_B$ when node B is in front and $V_{New}>V_B$ when node B is behind. Combining this with the previously obtained stable velocity we determine the final stable velocity as,

for a node B in front and d_{New} (predicted) $> R_{th}$ or for a node B in back and d_{New}(predicted) < 0

$V_{New}=V_{max} / 2$; when $V_B=V_{max} / 2$

$V_B<V_{New}<V_{max}$; $V_B>V_{max} / 2$

for a node B in front and d_{New} (predicted) < 0 or for a node B in back and d_{New}(predicted) $> R_{th}$

$V_{New}=V_{max} / 2$; when $V_B=V_{max} / 2$

$0<V_{New}<V_B$; $V_B<V_{max} / 2$