MULTI-ROOT MULTI-GENERATION WITH COLOR SCHEMA ROUTING ALGORITHM FOR MOBILE AD-HOC NETWORKS

THESIS

Presented to the Graduate Council
of Texas State University-San Marcos
in Partial Fulfillment
of the Requirements

for the Degree

Master of SCIENCE

by

Zhuojing Li, B.A.

San Marcos, Texas

May 2005

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Prof. Wuxu Peng for his invaluable guidance through the course of my graduate education. His important comments during our discussions made me grasp the issues involved in my research and helped me steer my work in the right direction.

I wish to thank Prof. Furman Haddix and Prof. Thomas McCabe for examining my thesis and offering me some useful insights. Interacting with them helped me develop both the conceptual and implementation aspects of my work.

Finally, this thesis would not have been possible without the support of my husband, my daughter and my parents. They have been a constant source of inspiration.

This manuscript was submitted on April 8, 2005.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

A mobile Ad-Hoc Network (MANET) is a collection of mobile, wireless hosts that cooperatively form a network based on peer-to-peer interactions among the hosts, independent of a centralized access or control [1]. In comparison with traditional mobile networks, each node can function as a router in MANET, where a packet can travel from a source to a destination either directly, or through some set of intermediate packet forwarding nodes. Ad hoc wireless networks eliminate the constraints of infrastructure and enable devices to create and join networks any time, anywhere. Several applications for ad hoc networks include: (1) Satellite to satellite in random orbits, (2) Automotive applications in which cars and traffic lights are communicating nodes, (3) Military applications such as battlefield communications among soldiers, fixed bases, and field encampments.

Research on MANET has mainly focused on developing routing protocols. As part of Dr. Wuxu Peng's project, my thesis presents a locality caching multi-root multi-generation with color schema (LCMRMGCS) routing protocol for MANET. The protocol uses a forest of spanning trees to keep track of the topology information within a mobile ad hoc network. Each tree is marked as a different color. In order to avoid the routing traffic to concentrate in one area, multiple roots are generated. Each root of a tree maintains host information of its own tree and is aware of at least one other to cooperatively route messages. Our routing algorithm improved the locality caching multi-root multi-generation (LCMRMG) routing algorithm [2]. In LCMRMG routing algorithm, multiple trees are overlap so that each root has to maintain the host information of the entire MANET. LCMRMG generates 32 roots for 1000 nodes. At the worst case, each root has to maintain all 1000 nodes. This presents significant maintenance for the root nodes In LCMRMGCS, the whole tree is divided into several smaller trees and the trees do not overlap. Therefore, the maintenance of each root node is considerably decreased. LCMRMGCS can be applied to a huge network while LCMRMG routing protocol cannot. As we expected, simulation results of LCMRMGCS showed that it did exceed LCMRMG in terms of maintenance and delivery ratio. Our maintenance is less than LCMRMG because each root only maintains its own tree and those trees do not overlap. In addition, we obtained a higher delivery ratio than LCMRMG because we made use of the color information of the destination nodes.

The remainder of the thesis is organized as follows:

Chapter 2 provides a background of the MANET routing protocols.

Chapter 3 is my explanation of the Locality Caching Multi-Root Multi-Generation with Color Schema (LCMRMGCS) routing protocol.

Chapter 4 presents the simulation procedures used and the analysis of simulation results.

Chapter 5 presents the conclusions and our future work.

CHAPTER 2

ROUTING ALGORITHM IN MANET

Routing in ad hoc networks has become a popular research topic. It is a process to find a path from a source to a destination to deliver a message. Before describing the protocols, it is important to explain the development goals for an ad hoc routing protocol so that the design choices of the protocols can be justified [3]. The following are typical design goals for ad hoc network routing protocols:

1. Minimal control overhead and minimal processing overhead. Control messaging consumes bandwidth, processing resources, and battery power to both transmit and receive a message. Therefore routing protocols should send the minimum number of control messages. Simple protocols require less processing cycles and then consume less battery power.

2. Dynamic topology maintenance. Nodes in MANET may move arbitrarily. Thus, the network topology may change randomly and rapidly. Once a route is established, it is possible that some link in the route may break up due to node movement. Therefore, a feasible routing path must be maintained.

3. Loop freedom. A routing loop occurs when some node along a path selects a next hop that occurred earlier in the path. Routing loops are extremely inefficient so that it should be avoided.

4. Power efficiency. Nodes in ad hoc networks rely on batteries for the power source. The nodes in mobile networks usually use some sort of standby modes to save power. It is therefore important that the routing protocol has support for these sleeping and temporarily inactive modes.

With these goals in mind, numerous routing protocols have been developed for ad hoc networks. In this chapter, we describe the characteristics of classes of routing approaches and the operation of particular routing protocols within those classes. Subsection 2.1 discusses current proactive/table-driven protocols, while subsection 2.2 describes reactive/on–demand protocols. Subsection 2.3 discusses hybrid protocols. Subsection 2.4 describes the other routing protocols.

2.1 Proactive/table-driven Routing

The proactive routing approaches designed for ad hoc networks are derived from the traditional distance vector [4] and link state [5] protocols that were developed for the wire line internet. The primary feature of proactive approaches is that each node in the network maintains a route to every possible destination in the network. Route creation and maintenance are accomplished through some combination of periodic and event-triggered routing updates. The advantage of a proactive protocol is that routes are available at the moment they are needed. However, the disadvantage of these protocols is that the control overhead can be significant in large networks or in networks with rapidly moving nodes. Proactive protocols perform well in networks where there are a significant number of data sessions within the network.

Clusterhead Gateway Switch Routing (CGSR):

CGSR [6] uses a hierarchical cluster-head-to-gateway routing approach to route traffic from the source to the destination. Gateway nodes are those within communication range of two or more cluster heads. A packet sent by a node is first routed to its cluster head and then to another cluster head via a gateway until the cluster head of destination node is reached. Figure 2.1 [7] illustrates an example of this routing scheme. The advantage of CGSR is less communication information. However it has a heavy overhead when the cluster head is changed very frequently.

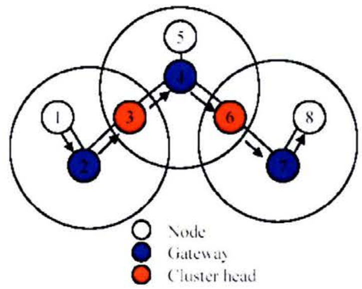

Figure 2.1: Clusterhead Gateway Switch Routing

Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)

TBRPF [8] nodes compute a shortest-path tree to all network nodes. The shortest-path tree is also known as the source tree. To minimize bandwidth utilization, the nodes only propagate a part of the tree to their neighbors. The partial source tree is called the reported tree (RT), which is proposed by Partial Tree-sharing Protocol (PTSP) [9]. The procedure to generate a reported tree at a router is briefly described as following. Check links that are in this router's shortest path tree. If such a link is estimated to be in the neighbors' shortest path trees, it is added to the reportable tree; otherwise, it is not included in the reportable tree.

2.2 Reactive/on-demand Routing

Reactive routing techniques, also called on-demand routing, use a different approach to routing from proactive protocols. In reactive routing, routes are created only when a source needs to communicate with another node whose path is unknown to the source. Therefore, route discovery becomes on-demand. If two nodes never need to talk each other, they do not need to utilize their resources to maintain a path between each other. The benefit of this approach is that signaling overhead is likely to be reduced compared to proactive approaches, particularly in networks with low to moderate traffic loads. The drawback of reactive approaches is the introduction of route acquisition latency. That is, when a route is needed by a source node, there is some finite latency while the route is discovered.

Labeled Distance Routing (LDR) Protocol

LDR [10] is based on a dual invariant consisting of destination sequence number and feasible distance. The feasible distance is the smallest distance to a destination reached by a node for its current sequence number to a destination. The feasible distance is used to avoid loops in routing. If the state distance of an advertisement is less than a node's feasible distance, there is no loop. The destination sequence number is used to reset the distance to a destination. Using feasible distance in LDR makes it more likely for nodes other than the destination to resolve route requests, which improves the performance significantly.

Position-aided On Demand Routing (PAR)

Motivation of PAR [11] is to effectively reduce the communication overhead associated with path discovery and maintenance with the assistance of position information. In PAR, a restricted directional flooding mechanism is designed, which creates an ellipse-forwarding zone and it can lead to paths with low communication overhead by imposing a restricted flooding across such a zone. The protocol uses the method of Location-guided Expanding Ring Search (LERS). LERS works by gradually enlarging the forwarding zone to search for a path. This procedure repeats until a path is found or the destination is unreachable. LERS can effectively prevent unnecessary network-wide flooding operations.

2.3 Hybrid Routing

The characteristics of proactive and reactive routing protocols can be integrated in various ways to form hybrid networking protocols. Hybrid networking protocols may exhibit proactive behavior in a certain set of circumstances and reactive behavior in different circumstances.

The Zone Routing Protocol [12] was used in ad hoc wireless networks consisting of many fast-moving nodes dispersed over a large geographical area. Neither a pure proactive nor a pure on-demand method will be adequate in such networks because of high degree of node mobility and potentially large number of destinations. Therefore ZRP takes a hybrid approach combining both types of routing. Around each node, ZRP defines a zone whose radius is measured in terms of hops Each node utilizes proactive routing within its zone and reactive routing outside of its zones

2.4 Other Routing

In mobile ad hoc network, battery power is an important consideration in designing routing protocols. Motivation of Maximum Life Routing [13] algorithm is to maximize the network lifetime. This algorithm works for both static networks and networks where the edge costs are changing slowly. The advantage of the algorithm is that it is a distributed, local-control approach and does not require a central node with global knowledge of the network.

2.5 Conclusion of Routing Protocols

Through reviewing the routing protocols in MANET, we can conclude three tendencies. The first is combination of proactive and reactive protocols The second is to dynamically optimize the routes. The third is that battery power has become an important consideration in designing routing protocols.

CHAPTER 3

LOCALITY CACHING MULTI-ROOT MULTI-GENERATION WITH COLOR SCHEMA ROUTING (LCMRMGCS)

In this chapter, we present our locality caching multi-root multi-generation with color schema routing (LCMRMGCS) We improve upon the limits of the protocol of locality caching multi-root multi-generation routing (LCMRMG) [2] protocol by employing a color schema to represent different trees. The most significant difference of our work with respect to [2] is that we create a forest of trees to represent the whole network to reduce the maintenance of the root nodes in contrast to a graph used in LCMRMG. As a further improvement of the LCMRMG protocol, LCMRMGCS holds the same assumptions as LCMRMG of the basic radio channel availability and channel link properties (e.g. bi-directional links). LCMRMGCS also assumes that any node will not fail their duties in sending and receiving valid messages.

3.1 Locality Caching Multi-root Multi-generation Routing (LCMRMG)

3.1.1 Single-root Spanning Tree Routing Protocol

Local Caching Multi-root Multi-generation routing algorithm [2] is on the top of the single-root spanning tree routing protocol [14], in which authors used a spanning tree (ST) along with generation tables (GT) to keep track of the topology information within a mobile network.

The spanning tree consists of a single root that could be any node in the network or its offspring. Each node has locally its generation number and Child ID. The generation number is the depth. Child ID is used to keep track of individual nodes within the same depth level. The single-root spanning tree algorithm is following:

REPEAT

1. Current station w checks its GT to see if destination station v is its offspring.
2. If so, send package m to its child x which is either the destination itself or has v as one of x's offspring.
3. If not, send package m to w's parent y.

UNTIL package m reaches destination v.

3.1.2 Locality Caching Multi-root Multi-generation Routing (LCMRMG)

The motivation of LCMRMG [2] routing algorithm is to cache traffic locality for better performances. To illustrate this idea, consider Figure 3.1, where *r* is a current root that covers hosts *s*, *a* and *b*. Host *a* is an offspring of host *s* while host *b* is not a descendent of host *s*. Therefore when *a* and *b* communicate each other, the packages are routed *a*—*s*—*r*—*b*. The total distance of this route is l1+h1+h2. However, if host *s* functions as a root, the previous route between *a* and *b* will be replaced by route *a*—*s*—*b*. The distance is thus reduced to l1+l2.

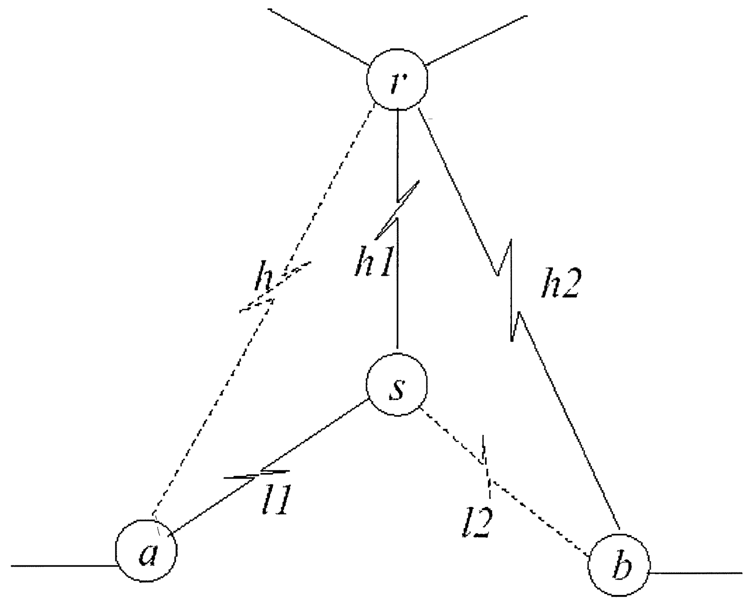

Figure 3.1: Traffic locality in spanning tree MANET

Algorithm for traffic locality caching

Let's consider a particular communication session between two hosts like host *a* and *b* in Figure 3.1. Suppose that hosts *a* initiates the communication. Host *a* first sends a request message to *b*, via *s* and *r*, attaching its coordinates (a_x, a_y) as part of the message. When host *b* receives the request message, it records the values (a_x, a_y). When *b* sends its reply message back to *a*, it attaches *a*' coordinates (a_x, a_y) and its own coordinates (b_x, b_y) as part of the reply. After a host knows the coordinates of its peer host in a communication session, the host will always attach the coordinates of its peer host on every message that it sends to the peer. In order to help locality caching, every root host like node *r* in Figure 3.1 will insert its coordinates (r_x, r_y) into each message routing through it. Therefore, when a reply message from *b* to a passes by host *s*, the latter now knows four pairs of coordinates: (a_x, a_y), (b_x, b_y), (r_x, r_y), as well as its own coordinates (s_x, s_y). With these pairs of coordinates, *s* can easily calculate the distances l1, l2, h1, h2. Thus the locality caching algorithm is following:

1. Each host maintains a counter variable ct that records the number of communication packets that has routed. Each of these packets carries three pairs of coordinates (source, destination, and root).
2. Each host *s* maintains two variable α and β. These two variables will be modified after each packet that carries the aforementioned three pairs of coordinates routing through *s*.
   a. The variable α records the accumulated distance values $l1 + h1 + h2$ as illustrated in Figure 3.1 For each packet that goes through *s*, the variable α is modified: $\alpha = \alpha + (l1 + h1 + h2)$
   b. The variable β records the accumulated distance values $l1 + l2$ as illustrated in Figure 3.2. For each packet that goes through s, the variable β is modified: $\beta = \beta + (l1 + l2)$

Algorithm for new root generation

Use the example in Figure 3.1 again. We can conclude that if compared to $l1 + h1 + h2$, $l1 + l2$ is very small, host *a* and host *b* must be very close. Thus to make the route from *a* to *b* going through s instead of s -- r is fairly valuable in saving bandwidth and shortening route distance. Moreover host *r* can be spared from participating in the routing process for its own good. Since α is the accumulation of $l1 + h1 + h2$ and β is of $l1 + l2$, it is not difficult to see that the smaller β is compared to α, the more eligible host *s* should become a root node. The question is how small β should be for *s* to be elected as a new root. Depending on the density and mobility of the node population, the decision can land in a fairly large range.

Based on the observation, the algorithm for new root generation is following:

1. Host *s* periodically checks the value of its counter variable ct. If the value of ct becomes sufficiently large (e.g. larger than a preset value), host *s* calculates the value of $(\alpha- \beta)/ \alpha$.

2. If the following two relationships hold $\alpha > \beta$ and $(\alpha- \beta)/ \alpha > \gamma$, the host will elect itself a new root node. In above relationships, $0 < \gamma < 1$ is a value that controls the strictness level of new root creation.
3. If host *s* is elected a new root, it will salvage the sub-tree rooted at itself to become the new spanning tree by propagating NEW signal throughout the sub-tree. The propagation is an up-down, generation-by generation recursive process common in the spanning tree maintenance operation.

LCMRMG routing algorithm

In LCMRMG, any node can simultaneously belong to multiple spanning trees. The strategy is choosing the path along the spanning tree in which the forwarding host has the lowest generation number. The reason is that in general lower generation nodes cover more offspring nodes and wider routing knowledge than the higher generation nodes. The LCMRMG routing algorithm is following:

For every host in LCMRMG MANET:

1. performs spanning tree maintenance operations.
2. for each incoming packet.
   a. performs the algorithm for traffic locality caching.
   b. if the destination host of the packet is in one of its sub-trees, forward the packet to the root of that sub-tree. If there is a tie, choose the one with relatively low generations.
   c. Otherwise, forward the packet to the parent host that has the lowest generation number.
3. performs the algorithm for new root election.

3.2 Locality Caching Multi-root Multi-generation Routing with Color Schema Routing (LCMRMGCS)

We add color schema into LCMRMG to create a forest of trees. LCMRMGCS has two main benefits. One is that we can make use of the color information of the destination in routing. The other is that each root of a tree only has to maintain host information in its own tree. The root nodes are aware of one another through a path that was formed by the tree building procedure and they cooperate to route messages.

3.2.1 Key points of color schema

We consider whether there exists a link between two nodes, rather than the real physical positions of the nodes. That is to say that each link has the same length. From this point of view, we can divide the whole tree into several pseudo trees with clear boundaries. Each pseudo tree has a unique corresponding to the real tree even though their appearances are different. The reason is that in the real trees the green tree may be mixed up with the red tree, while in the pseudo trees the green tree separates from the red tree clearly even when the nodes are moving.

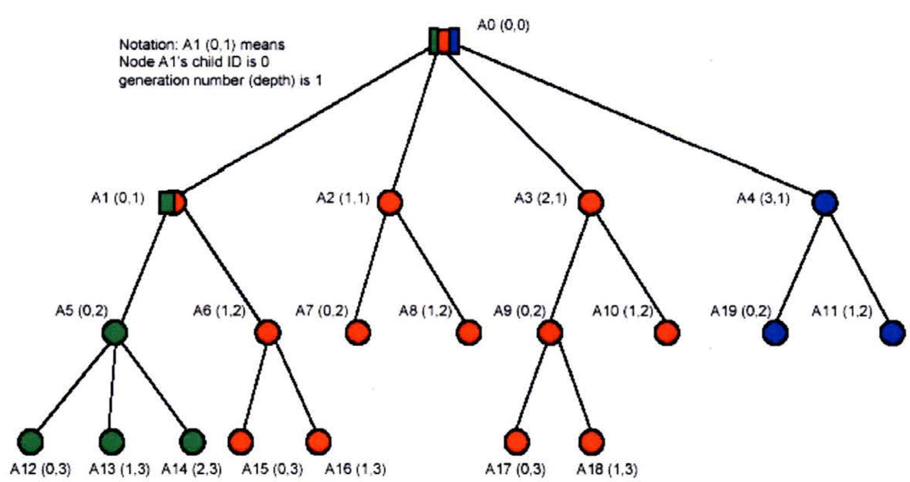

Figure 3.2: Forest example

3.2.2 Network construction

We use generation number (GEN) and child ID (CID) to describe the nodes in the networks. For example A4 (3,1) means node A4's child ID is 3 and its generation number is 1. That is to say A4 is the first generation of the tree and it is the third child of its parent.

At the beginning, we choose a node as a root, which can be any node within the network. In Figure 3.2, A0 is chosen as a root. Then A0 will broadcast soliciting messages to request other nodes to join A0's tree. Those nodes within the A0's radio transmission range will respond with a ready signal if they are not yet part of the tree, and A0 will assign them with sequentially increasing CIDs and a GEN equal to GEN (A0) + 1. At this point, the tree has generation 1 and rooted at A0 with leaf nodes A1 (0,1), A2 (1,1), A3 (2,1) and A4 (3,1). After signed on to the tree, A1, A2, A3 and A4 will start soliciting their children just as A0 has done previously. For nodes that are out of range of A0 and missed A0's request, they may be able to join A0's tree as A0's grandchildren. The same procedure will be carried out again and again recursively until the tree covers all the nodes in the network.

Once the network topology is built, the leaf nodes will start the process of creating generation tables (GT), thereafter, the generation tables will be propagated upward along the tree branches until reaching the root. For example, Table 3.1 is the generation table for A12. Next, A12 will send its GT to its parent A5, and A5 start to build its generation table. The A5's generation table cannot complete until A5's all children have reported their GTs. In other word, A5 have to contain information sent from A13 and A14. Once A5 finished its GT, shown by Table 3.2, A5 will report its GT to A1. Similarly, A6 also send its GT to A1 so that A1 state to build its GT, shown by Table 3.3. Then, A1 sends its GT to root A0.

A12's ID	A12
A12's parent ID	A5
A12's generation number	3
A12's child ID	0
Future generation	
NIL	

Table 3.1: Node A12's generation table

A5's ID	A5	
A5's parent ID	A1	
A5's generation number	2	
A5's child ID	0	
Future generation		
Child ID	ID	Future generation
0	A12	NIL
1	A13	NIL
2	A14	NIL

Table 3.2: Node A5's generation table

A1's ID	A1	
A1's parent ID	A0	
A1's generation number	1	
A1's child ID	0	
Future generation		
Child ID	ID	Future generation
0	A5	A12, A13, A14
1	A6	A15, A16

Table 3.3: Node A1's generation table

3.2.3 Network maintenance

MANET is characterized by fluid topology change. Links between two nodes are likely to be formed or broken at unpredictable moment. In our algorithm, any change in network topology will cause the change in generation table. Therefore, we maintain the generation table in order to keep track of the topology of the networks. Our algorithm deals with the changing situation by taking into account three scenarios, node sign-on, node sign-off, and node movement.

Sign-on

Sign on means that a node wants to join the network. The sign-on node broadcasts soliciting messages expressing intention to join in the network. Nodes within the radio transmission range of the sign-on node respond the request with their positional information in the tree. The sign-on node will choose from the responders the one with the lowest generation as its parent. Therefore, the sign-on node becomes a new leaf node in the existing tree. Like other leaf nodes, the sign-on node will build its GT and then report it to its parent. The recursive bottom-up propagation takes place again to inform the upper level nodes of the emergence of the new member until the propagation gets to the root.

Sign-off

When a node that belongs to the tree is about to power off, all the relating nodes must be informed and proper steps must be followed in order to keep the topology up to date. The sign-off node messages its parent the intension to leave. Its parent updates the GT by removing the entry for this node as well as entries for the entire node's future generation. Following is again the bottom-up propagation of GTs until reaching the root, the GTs of all lower generation nodes relating to the sign-off node are updated. At the same time, the sign-off node sends release messages to all its children in the tree. In the opposite direction, the release message propagates recursively in a top-down fashion toward the leaf nodes. Any node that receives the release message will forgo its membership of the tree, and begins the process of signing on to the tree as a new node.

Movement

Before the movement, a node will notify both the parent and the children of its intension to move by sending start-of-move messages. After the movement, the node will notify them again by end-of-move messages. If for a certain period of time, no end-of-move message is received, the relative nodes will abandon the moving node and take steps to adjust the topology. The steps are the same as if the moving node signed off the network. Then, the moving node itself will try to join the network again as a new node.

Note that in LCMRMG routing protocol, a node can belong to many parents. Therefore LCMRMG uses a function (sign-on-again) to sign on to the different parents so that the relative trees will change the topology. Similarly, when a node signs off, all its relative parents have to change their generation tables until the updates get to the roots. The overlapped trees cause huge maintenance of LCMRMG. LCMRMG often uses 32 roots, so in the worst case, the topology information of the networks is duplicated by 32 times. Any change in topology will lead to the changes of 32 trees.

3.2.4 Algorithm for creating forest

Figure 3.3 presents the flow chart to create the forest. First, we assign the first root a color such as red. Then its all children are also labeled as red. Once the second root is generated, we assign the second root and its children a different color, for example green, and then cut off the generated green tree from the old red tree. We therefore obtain two trees, the red tree and the green tree. At the same time, we record the path from the green root to the red root by labeling those nodes along that path with two colors, red and green. Once a third root is generated from red tree, we assign a third color, such as blue, and repeat the above steps.

Let's show an example to illustrate how to create the forest and the benefit of this method. In Figure 3.2, all the nodes are red at beginning. At some point, A5 becomes a new root. Then A5 is assigned green and A5 notifies its children A12, A13 and A14 to change to green from red. If any of A12, A13 or A14 is not a leaf, it will notify recursively in a top-down fashion the leaf nodes to change their colors to green. At the same time, A5 notifies its parent A1 to change color from red to red-green. The recursive bottom-up propagation takes place again to inform the upper level nodes until to the root A0. So A0 changes to red-green again. Then the green tree is cut off from the red tree. That is to say A1 only knows that A5 is one of its children, but does not know A12, A13, and A14 are its grandsons any more. A5 knows A1 is its parent A5's GT keeps the same Table 3.2 while A1's GT reduces from Table 3 3 into Table 3.4. Therefore, the generation tables of different trees are exclusive. Thus, the maintenance of the generation tables will decrease. For example, when the topology changes in A12, A13 or A14, A1 does not need to maintain such changes

A1's ID	A1	
A1's parent ID	A0	
A1's generation number	1	
A1's child ID	0	
Future generation		
CID	ID	Future generation
0	A5	NIL
1	A6	A15, A16

Table 3.4· Node A1's generation table after cutting off. A1 does not know A12, A13 and A14 any more. Therefore A1's generation table is reduced.

In additional of the generation table, each root has a color table to record the color information of the destinations that passed the root. The color table will be used to guide the inter-tree search. Moreover, each node has a color marker, which was implemented by 32 bits corresponding to 32 colors in our simulation. If a node was located on the path that connects two roots, more than one bit will be marked. This color marker will help the roots communicate to each other.

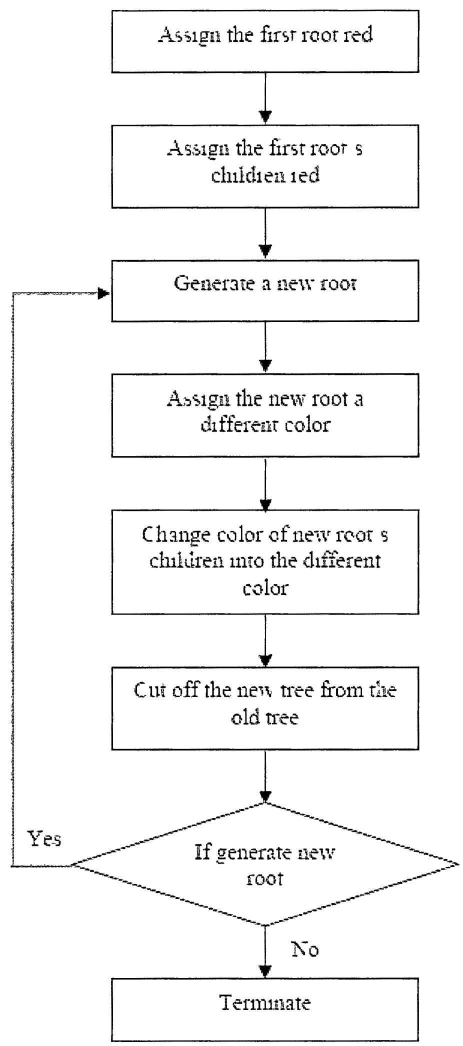

Figure3.3: Flow chart for creating forest

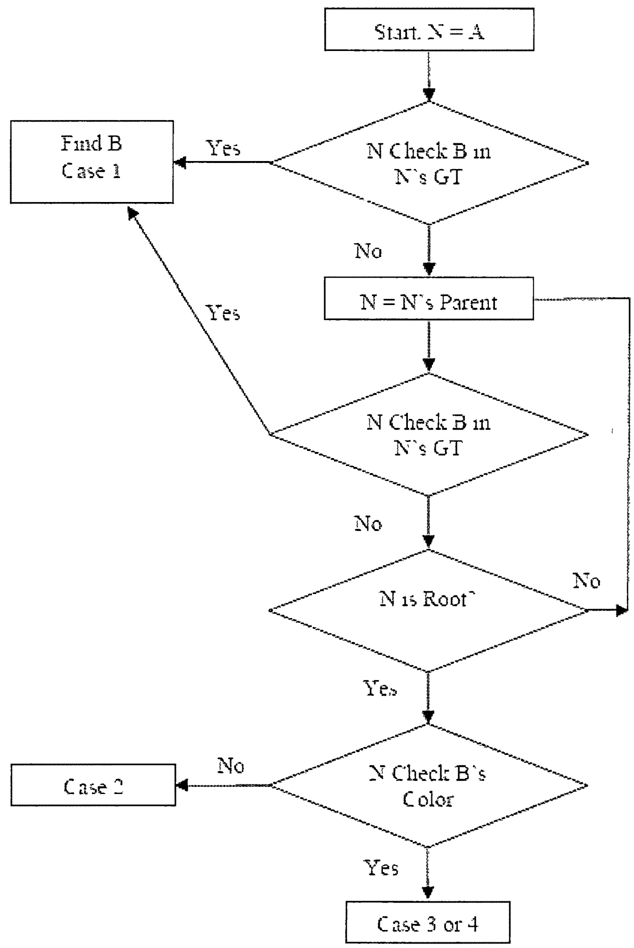

Figure 3.4: Our routing algorithm

3.2.5 Our routing algorithm

To illustrate our procedure, we show one example, routing from source A to destination B, under A given as red. We make use of color information of a node to process this routing. Once a node signs onto the network, it is assigned a color Therefore every node in the network has a color. There are total four cases of B's color. Figure 3 4 shows the routing procedure from A to B.

Case 1: B is the same color as A We can find B through checking A's generation table or A's parent's generation table until A's root's generation table Let's show an example using Figure 3.2. Suppose that the topology is following. A5, A12, A13 and A14 are green and A5 is the root of the green tree. A4, A19 and A11 are blue and A4 is the root of the blue tree. The other nodes are in the red tree. Therefore A1 is green-red and A0 is green-red-blue. The source is A12 and the destination is A14. They are both green. First A12 check its generation and cannot find A14. Then A12 sends the message to its parent, A5. A5 checks its generation table to find A14. A5 finds A14 is one of its children, so A5 forwards the message to A14.

Case 2: A's root does not know B's color. It implies that B is not in the same tree as A, and B is not in the color table of A's root. Otherwise, A's root can check its generation table or its color table to find B. So, we use a blind search to find B's root from A's root to A's root's parent and so on.

Case 3: A's root knows B's color and B is a different color from A. This case means that A's root cannot find B in its generation table but can find B's color in its color table. We use Figure 3.2 again as the example. Suppose that the topology is same as Case 1 and that the source is A12 and the destination is A19. First, A12 checks its generation table. A12 cannot find A19 in its generation table, so A12 sends the message to its parent A5. A5 is the root of the green tree, so A5 checks its table for the A19's color information. A5 finds A19 is blue. So A5 forwards the message to A1. A1 is green-red, not having blue as one of its colors. Then A1 forwards the message to its parent A0. Because A0 is green-red-blue, including the blue color of the destination A19, A0 can check its children's colors and find that A4 also has the destination color, blue. So A0 forwards the message to A4 and A4 can check its generation table to determine the child to which the message should be forwarded. A4 forwards the message to A19 thereby finding the destination. The shortest path was achieved for a tree structure if the destination can be found in some generation tables.

Case 4: B's color changed after its color is stored in A's table. First, according to B's color information stored in A's root, we try to search the blue tree. If B cannot be found, we have to search B like we do not know the color information of B as in Case 2.

Finally, note that the traffic in a network may have some features, even though we think the traffic is random. For example, the traffic in weekdays is different from that in weekend. Forming trees through traffic make the routing inside a tree easier and shorter than routing in different trees. This feature allows our routing algorithm to be more efficient.

3.2.6 Comparison with other routing algorithms

We now compare LCMRMGCS with some other routing protocols to show the advantages of LCMRMGCS.

1. Dynamic Group Construction Routing

Dynamic group construction routing [15] uses an adaptive approach for routing management. In this approach, the network infrastructure is constructed by several communication groups, which are called routing groups. A routing group communicates with other routing groups via the boundary mobile hosts as forwarding nodes. In a routing group the mobile hosts are divided, by means of the dominating values, into two groups, one positive cluster and several non-positive clusters.

Dynamic group construction routing protocol considers the existing links while LCMRMGCS focuses on active links. LCMRMGCS is more applicable than the dynamic group routing protocol because more links existing does not mean more links active. For example, my cell phone has a lot of links with my neighbors, but I will not talk with my neighbors. This is one advantage of our algorithm. The other benefit of our algorithm is using a forest to describe the whole network because a tree is an efficient data structure to search a destination.

2. Locality Caching Multi-Root Multi-Generation Routing (LCMRMG)

In LCMRMG [2], a host can calculate the estimated reduction of the network traffic if it becomes a new root router and can is elected to do so if the reduction exceeds a predefined threshold value. Thus LCMRMG generates multiple roots based on routing traffic for the network.

LCMRMGCS is based on LCMRMG, with a purpose to reduce the maintenance of root nodes by dividing the whole tree into several trees. Each root just needs to maintain the host information of its own trees rather than the whole network. Therefore, LCMRMGCS can apply to a huge network in contrast to LCMRMG because each root of a tree in LCMRMG has to maintain the host information of the entire MANET.

Figure 3.5 and Figure 3.6 show the main difference between LCMRMG and LCMRMGCS. LCMRMG uses a graph to describe the network while LCMRMGCS applies a forest to represent the network.

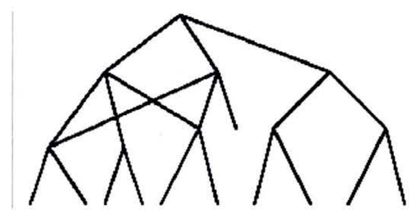

Figure 3.5: Data structure for LCMRMG: graph

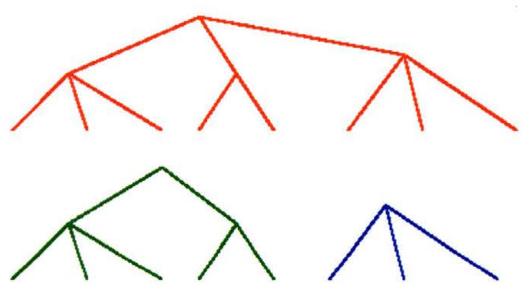

Figure 3.6: Data structure for LCMRMGCS: forest

CHAPTER 4

LCMRMGCS SIMULATION

A simulation is the creation of a set of computer programs that imitate the physical requirements of the proposed protocols and observation of the execution of these programs in a digital environment. Thus simulation can significantly help system engineers to obtain crucial performance characteristics [16], [17]. There exist pre-implemented, full-blown simulators for networking. Two popular simulators are NS-2 [18] and GloMoSim [19]. NS-2 is a discrete event simulator targeted at networking research. It provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless networks. GloMoSim built a scalable simulation environment for wired and wireless network systems. GloMoSim currently supports protocols for a purely wireless network.

We decide to generate our own simulator rather than use those pre-implemented simulators based on the following reasons.

1. We aim to improve the LCMRMG algorithm. We therefore choose the same simulator as that algorithm. We do not concern the physical radio implementation, the medium-access control channels, and the real-world packet forwarding mechanism based on IP stacks, all of which enhance the complication of the simulators.

2. Using the aforementioned simulators is time-consuming because users have to understand how to develop modules that can be plugged into the simulators, and have to master some scripting techniques.

We created the simulator from scratch on UNIX platforms. It is written entirely in C language and built with GUN C (GCC) complier. In order to simulate with various scenarios and under different conditions, our simulator provides a set of command line switches to control the program execution. The simulator outputs several log files as simulation proceeds, through which users can look into the simulation proceeds and observe the behavior of the simulation protocol. These log files are the basis of simulation analysis.

Each mobile host in our simulator was represented by one running thread. Our simulation created 1000 mobile hosts (1000 threads). Thus, network communication was transformed to thread communication. The size of the network depends on OS implementation. Therefore the simulation results may behave somewhat differently across different UNIX platforms, such as Linux, FreeBSD and Solaris.

4.1 Data Structures

A single thread represents each mobile host. Each node has a node id (nid) corresponding to each thread. The thread only executes in its own territory and does not interfere with others. Integer *c* denotes the color of a node. The actual representation of a node is expressed in the following C structure:

```
struct node {
pthread_t ptid;                 /* thread identifier for this node */
int nid;                        /* ID of this node */
location_t location;            /* most recent location */
location_t direction;           /* moving direction */
double speed;                   /* moving speed */
int is_station;                 /* showing if it's turned on or off */
int signed_on;                  /* showing if it's signed onto the tree */
int is_root;                    /* if it's a root */
struct msg_q messages;          /* its routing buffer */
pthread_mutex_t q_mutex;        /* message queue mutex */
struct mesg msg,                /* the latest message it received */
int traf1;                      /* for non-root : total number of messages going up;
                                for root: total number of messages passing by */
int traf2,                      /* for non-root· the number of messages for which
                                current roots are sufficient for root· the number of
                                 messages that cannot be routed */
```

```
double r,                    /* route distance through root */
double s,                    /* route distance through itself */
int c,                       /*  color of a node */
}
```

Location and direction take the form of

```
typedef struct {
   double x, y,  /* x and y are the coordinates of the node's current position and the next position*/
} location_t ,
```

The messages are a simple implementation of a circular queue structure as

---

```
struct msg_q {
    struct mesg buf[BUFSIZE],
    int front,
    int back,
},
```

The actual messages are represented as

```
struct mesg {
   int src,
   int dst,
   char body[2], null,
   location_t sloc,
   struct timeval stv,
   unsigned long time_total,
   unsigned long time_lasthop,
   int hops,
},
```

Most of the fields in the message structure are for the statistical purpose and the message has a dummy message body of size 3 bytes. Note that each message in LCMRMGCS needs to carry up to three set of coordinates as stated in the locality caching algorithm. Instead of actual carrying them in the each message, for the sake of convenience, each message only carries the nodal index of the relevant nodes.

4.2 User Interface

Our simulator supports a command line interface that takes a set of parameters, among which users have the choice of selecting the simulation protocol (e.g. single-root vs. multi-root). Once invoked, the simulator will keep running until explicitly terminated by users with Ctrl-C command or the time period specified at command line has elapsed. The running simulator dumps simulation data into several log files.

The command line arguments are following

```
Options.
        -h            ---- show this help
        -l <Lambda>   ---- transmission range for all nodes, default to 5
        -m <Max>      ---- number of nodes in the simulation, default to 1000
        -n <miNutes>  ---- number of minutes the program will run, default to 0 => forever
        -o <Off>      ----- number of nodes that can randomly power off, default to none
        -q <reQuest>  ---- request frequency, default to 10 second
         -r  <Root>   ---- root number, 0 => multi-root, 1 => single root, default to 0
         -s <Speed>   ---- moving speed for all nodes, 0 => random moving, default to 0.1
         -t <raTio>   ---- (x-y)/x, criteria to make new roots, default to 0.5
```

Four log files will be generated after a simulation session – monitor.log, action.log, message.log, and status.log.

Sample monitor.log

```
30 sec ·  sent=2831, rec=1191, t_routing=423704, e_routing=36631, maint =100615, roots=27
60 sec    sent=5665, rec=2817, t_routing=500648, e_routing=367730, maint =72121, roots=32
90 sec.:  sent=8331, rec=3354, t_routing=555183, e_routing=428576, maint =68570, roots=32
120sec .  sent=10815, rec=3805, t_routing=617715, e_routing=467923, maint =57156, roots=32
150sec    sent=13115, rec=4085, t_routing=670407, e_routing=480703, maint=60499, roots=32
180 sec.: sent=15404, rec=4322, t_routing=715516, e_routing=510009, maint=56685, roots=32
```

Sample action log

```
Node: 995 starts sending message to node 544, with 4 nodes within node 995's trans range
Node· 11 starts sending message to node 267, with 13 nodes within node 11's trans range
Node: 996 starts sending message to node 923, with 12 nodes within node 996's trans range.
Node: 24 starts sending message to node 490, with 9 nodes within node 24's trans range
Node· 25 starts sending message to node 196, with 6 nodes within node 25's trans range.
Node· 997 starts sending message to node 440, with 11 nodes within node 997's trans range.
```

Node: 22 starts sending message to node 944, with 8 nodes within node 22's trans range
Node: 10 starts sending message to node 826, with 10 nodes within node 10's trans range

---

Sample message log

Node 275 (45.00, -26.00) received a message from node 522 (44 00, -33.00), with 2 hops.
Node 525 (-33.00, 28.00) received a message from node 315 (-35.00, 26.00), with 1 hops
Node 589 (-41 00, 22 00) received a message from node 185 (-47 00, 28 00), with 2 hops
Node 31 (-31.00, -8.00) received a message from node 184 (-38 00, -23 00), with 5 hops.
Node 183 (-29 00, 22 00) received a message from node 798 (-30 00, 19 00), with 4 hops.
Node 393 (-27.00, 24 00) received a message from node 16 (-38.00, 33.00), with 6 hops.
Node 798 (-30.00, 19 00) received a message from node 578 (-38 00, 29 00), with 4 hops.
Node 999 (-43 00, 38 00) received a message from node 704 (-36 00, 11 00), with 8 hops.
Node 31 (-31.00, -8.00) received a message from node 945 (-29.00, -18.00), with 3 hops.
Node 247 (-37.00, 16.00) received a message from node 101 (-47.00, 8.00), with 8 hops.
Node 294 (-34 00, 45 00) received a message from node 53 (-41 00, 24 00), with 9 hops.

Sample status.log

```
Node 172·
    Location: (23.00, -14.00)
    Direction (23 00, -14 00)
    Speed: 0 38
    'traf1': 0
    'traf2'  0
    'r': 0.000000
    's': 0.000000
        ROOT TABLE#0
        Root: 0
        Parent· 146
        Generation  5
        Child ID: 0
        Number of children. 2
        Children are.
    224:  891, 434, 355, 108, 644, 565, 775, 55, 985,
    487   697, 907,
```

Monitor.log traces and calculates all the important statistics periodically, which is the most important log file for us to measure and compare network performance afterwards. Action.log and message.log describe network traffic information. The former denotes the source host sending packets out and its surrounding environment while the latter records the packet arrival and the associated parameters. Status.log visualizes the topology situation at any moment from the viewpoint of individual hosts.

4.3 Simulation Results and Analysis

We compared the routing performance between our protocol and LCMRMG [3] protocol to verify the performance gain of the former over the latter. The ad hoc network is composed of 1000 nodes since the LCMRMG cannot support a larger network with the restriction of the operating system. The simulation results are observed with three sets of performance metrics, delivery time, delivery maintenance and delivery ratio. We use a log file (monitor.log) to record the CPU time, number of sent messages, number of received messages, maintenance and the root number at every 30 seconds. The delivery time is the total CPU time spent on successful routing, which increases with the number of received messages. Successful routing means the routing packet reached the destination form the source. The delivery maintenance represents the relationship between the maintenance and the received messages. In our simulation, the maintenance is defined as the accumulated number of the changes in generation tables of all nodes caused by sign-on and sign-off procedures. The delivery ratio is the percentage of the received messages over the sent messages. Our results show lower delivery time and delivery maintenance and a higher delivery ratio than those of LCMRMG protocol. Therefore our protocol is more efficient under the comparisons.

To show the features of our protocol, we analyze the relationship between the delivery ratio, the root number and the hop number. The hop number is the total number of nodes traversed for a routing message delivered from the source to the destination. There exists an optimum root number for a given total node number in the network to obtain a highest delivery ratio.

1. Simulation comparisons:

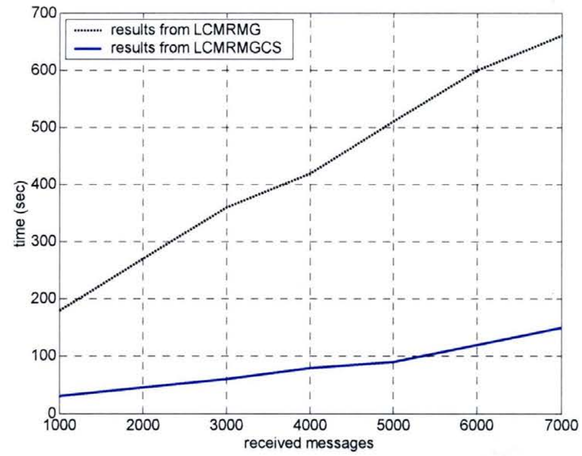

Figure 4.1: Received message vs. CPU time

In Figure 4.1, we measured the CPU time in second that spent on routing for a given number of successfully received the messages. As shown in Figure 4.1, the solid-curve is our result and the dot-curve is from LCMRMG. Both curves show an increasing trend of the time with more received messages. The slope of our curve is less than that of LCMRMG, which means our routing is faster than LCMRMG. Therefore we conclude that our protocol spends less time than LCMRMG to receive a given number of received messages and thus our protocol is more efficient. The color information used in our protocol can guide the destination searching, especially when the destination node is not in the same tree as the source, i.e., not the same color as the source. In LCMRMG, a blind search is always used to find which parent has a generation table to inform the position of the destination. Once the table is found, the advantage of a tree structure in knowing the shortest path to the destination is equal for our protocol and LCMRMG.

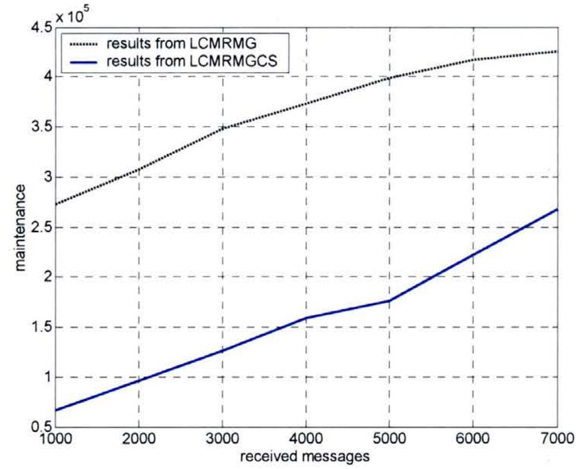

Figure 4.2: Received message vs. maintenance

In Figure 4.2, we give the maintenance associated with a given number of received messages. A high required maintenance usually causes a longer routing time. The maintenance is measured by the accumulated number of the changes in generation tables of all nodes caused by sign-on and sign-off procedures in updating the network topology. It is clear in Figure 4.2 that our protocol needs less maintenance than LCMRMG. The extra maintenance in LCMRMG is spent by the multiple sign-on and sign-off procedures of a node since a node frequently belongs to several trees in LCMRMG. Every tree that has the node will update the topology if a node signs on or signs off. In contrast, a node only belongs to a single tree in our protocol, which makes the maintenance more efficient.

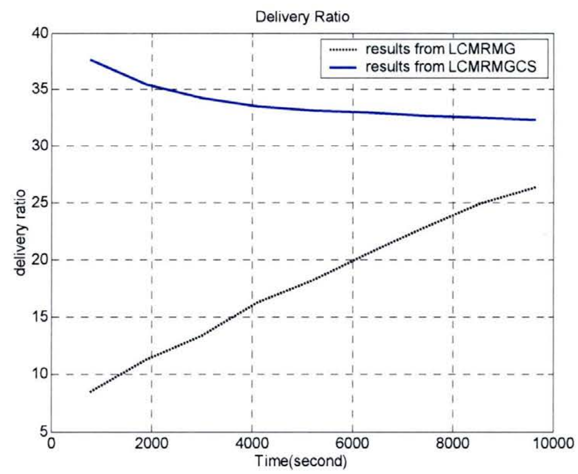

Figure 4.3: Delivery ratio vs. time

In Figure 4.3, we show the delivery ratio that is the ratio of the received messages over the sent messages as reported at different observation times. Our delivery ratio denoted by the solid-curve decreased slowly and finally stabilized at 32%. The delivery ratio of LCMRMG denoted by the dot-curve showed an increasing trend with time. The ratio started with less than 10% and climbed to 27% in 3 hours. After continuing running without restarting for 3 days, LCMRMG will reach a high delivery ratio of 60%. Our protocol has a higher delivery ratio than LCMRMG during the first 3 hours but a lower delivery ratio than LCMRMG in the long term since LCMRMG continues to increase. The higher delivery ration of LCMRMG in the long term is due to the overlap of the trees, where each node has multiple paths to route the messages. LCMRMG does not control the degree of such overlap so that most of nodes belong to 32 trees simultaneously in the long term. This kind of overlap is not allowed for a larger network even though a high delivery ratio has been reported. Our higher initial delivery ratio and moderate delivery ratio in the long term is more desirable for such applications that need a quick deploying or that last for few hours, such as disaster recovery and conferences.

2. Root number:

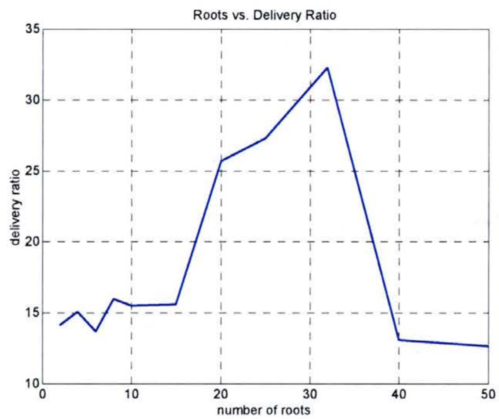

Figure 4.4: Roots vs. delivery ratio

The delivery ratio is a function of root number for a given ad hoc network. Different root number will generate a different delivery ratio. Our simulation results show that 32 roots yield a best performance in terms of delivery ratio even though we cannot provide a proof. As shown in Figure 4.4, before 32 roots, the more roots, the better is the delivery ratio. However, after 32 roots, the more roots, the worse is the delivery ratio. The reasons to support an optimum root number are that more roots mean more color information, which help routing and that too much color information is apt to provide wrong information. For a single root, our protocol turns out to be the spanning tree protocol [28], which only gives a delivery ratio at 12%. Therefore, our protocol is better than the spanning tree protocol once we introduce multiple roots to relieve the traffic and color information to guide the search.

3. Hop number:

The hop number is the total number of nodes traversed for a routing package delivered from the source to the destination. A larger hop number associated with a received message usually implies a longer routing time for the message to reach the destination. As we discussed before, the delivery ratio depends on the root number allowed in the network. In order to analyze the relationship among the delivery ratio, the root number and the hop number, we present several simulation examples in Figures 4.5 (a-c).

In Figure 4.5(a), there are 32 roots in a network with 1000 nodes. We measured all the hop numbers for each received message until the delivery ratio is steady at 32%. Then we plot the histogram of messages received according to their hop numbers. The median value of the hop number is 12 and messages with hop numbers between 5 and 15 account for 61% of all received messages.

In Figure 4.5 (b) and Figure 4.5(c), we change the root number to adjust the delivery ratio. There are 26 roots in a network with 1000 nodes in Figure 4.5(b) and 18 roots in Figure 4.5(c). The delivery ratio dropped from 32% to 28% in Figure 4.5(b) and to 21% in Figure 4.5(c). As we expected, the median value increased from 12 in Figure 4.5(a) to 18 in Figure 4.5(b) and to 19 in Figure 4.5(c). The percentage of messages with 5-15 hop numbers decreased from 61% in Figure 4.5(a) to 36% in Figure 4.5(b) and to 34% in Figure 4.5(c). Therefore, messages with higher hop numbers usually correspond to a lower delivery ratio.

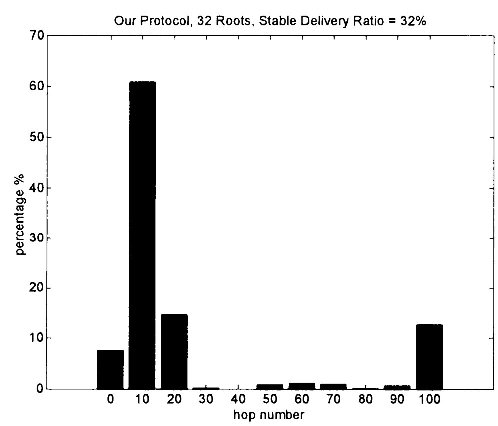

Figure 4.5 (a): Median =12, 5-15 hops = 61%

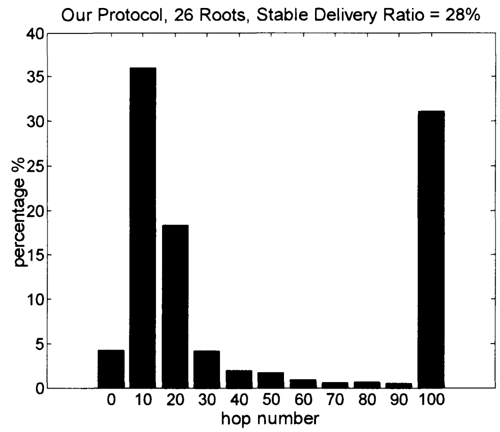

Fig. 4.5 (b): Median =18, 5-15 hops = 36%

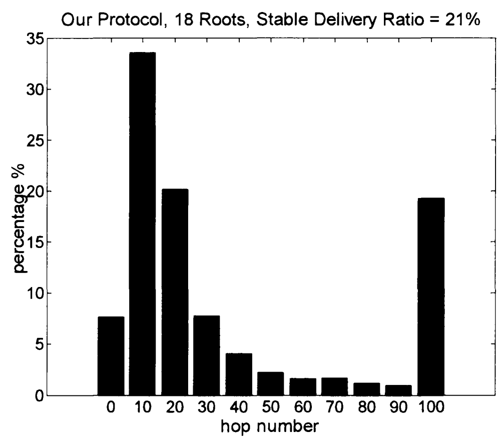

Fig. 4.5(c): Median =19, 5-15 hops = 34%

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

In this thesis, we have presented a multi-root multi-generation with color schema (LCMRMGCS) routing protocol for MANET. Our protocol uses a forest of trees to keep track of the topology information within a mobile ad hoc network. The major advantage of using a tree-similar structure in routing is that the shortest path can be achieved if the destination can be found in some generation table. Any change in topology within the networks will lead to changes in the generation tables of the relative nodes. We generated multiple trees depending on the routing traffic. Each tree is marked as a different color. Each root of a tree maintains host information of its own tree and is aware of at least one other to cooperatively route messages through the path that was formed by the tree building procedure. The path was marked by colors.

The benefit of the color schema is that each root of a tree only has to maintain information about its own tree instead of the whole network. The maintenance of root nodes is significantly reduced as compared with LCMRMG [2]. We introduce multiple roots to reduce the routing traffic. In a short period of time, the network traffic situation can be regarded as stable. Once we obtain this special traffic information and generate roots to consider this situation, we can deal with a similar situation in the following period successfully and efficiently.

Simulation results show that LCMRMGCS is better than LCMRMG in terms of CPU time, maintenance and delivery ratio. LCMRMGCS takes less CPU time than LCMRMG to route a given number of messages successfully since the search guided by the color is better than the blind search. The color table stored in the roots of LCMRMGCS can reduce the unnecessary searching, which is one of advantages of LCMRMGCS over LCMRMG. Moreover, maintenance of the networks with LCMRMGCS is less than with LCMRMG because each root just maintains its own tree and those trees do not overlap. The reduction of the maintenance provides a faster routing speed for LCMRMGCS. LCMRMGCS obtained a higher delivery ratio than LCMRMG because it makes use of the color information of the destination node even though LCMRMG surpassed LCMRMGCS after running continuously for 3 hours. However, the slowly increased delivery ratio of LCMRMG is a weakness for such applications that need a quick deploying or that are used temporally. The high initial delivery ratio of LCMRMGCS makes it more applicable than LCMRMG. For the root number, our simulation has shown that 32 roots always obtain the best performance. This is because more roots mean that there is more color information, which helps routing whereas too much color information is apt to provide wrong information.

5.2 Future Work

The root number of the mobile ad hoc network is an interesting problem. Our future work will be the relationship between the root number and the node number. If there is only one root, maintenance is easy while reliability is bad. If every node is a root, reliability is good while maintenance is hard. Therefore, there may be a balance between maintenance and reliability.

REFERENCES

[1] M. Frodigh, P. Johansson, and P. Larsson. "Wireless ad hoc networking: the art of networking without a network," Ericsson Review, pp. 248-263, No. 4, 2000.

[2] W. Peng, X. Zhang and K. Makki, "Locality caching multi-root multi-generation routing algorithm in mobile ad hoc networks," Proc. of the International Conference on Computer Communication and Network (ICCCN), pp. 229-234, Oct. 2003

[3] S. Basagni, Mobile ad hoc networking, IEEE Press, 2004.

[4] G. S. Malkin and M. E. Steenstrup, "Distance-vector routing," Routing in Communications Networks, pp. 83-98, Prentice Hall, 1995.

[5] J. Mo, "Link-State Routing," Routing in Communications Networks, pp. 255-265, Prentice Hall, 2000.

[6] C. C. Chiang, "Routing in Clustered multihop mobile wireless networks with facing channel," Proc. of IEEE Singapore International Conference on Network (SICON), pp. 197-211, April 1997.

[7] E. M. Royer and C. K. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," IEEE Personal Communications, vol. 6, no. 2, pp. 46-55, April 1999.

[8] R. G. Ogier et al., "Topology dissemination based on reverse-path forwarding (TBRPF)," Request for Comments 3684, Feb. 2004.

[9] R. G Ogier, "Efficient routing protocol for packet-radio networks based on tree sharing," Proc. of International Workshop on Mobile Multimedia Communications (MoMuC), pp. 104-113, Nov. 1999.

[10] J. J. Garcıa-Luna-Aceves, M. Mosko and C. Perkins, "A new approach to on-demand loop-free routıng in ad hoc networks," Proc. of ACM Symposıum on Princıples of Dıstrıbuted Computing (PODC), pp. 53-62, July 2003.

[11] B. Zhang and H. Mouftah, "Position-aıded on demand routıng protocol for wireless ad hoc networks," Proc. of IEEE International Conference on Communıcatıons (ICC), pp. 3764-3768, March 2004

[12] Z. Haas, "A new routıng protocol for the reconfigurable wireless networks," Proc of International Conference on Universal Personal Communıcations, (ICUPC), Oct 1997

[13] A. Sankar and Z Liu, "Maxımum lıfetıme routing ın wıreless ad-hoc networks," Proc of the Conference on Computer Communications (INFOCOM), pp 1089-1097, July 2004

[14] X. Chen and X. D. Jia, "Package routıng algorithms in mobıle ad-hoc wireless networks," Proc. of the International Conference on Parallel Processing Workshops (ICCPW), Sept 2001

[15] Y. Chang and C. Hsu, "Routing in wireless/mobile ad-hoc networks via dynamic group constructıon" Mobıle Networks and Applications, vol. 5, no.1, pp 27-37, March 2000.

[16] L. Bononi, G.D. Angelo and L. Donatıello, "HLA-based adaptıve dıstrıbuted simulation of wireless mobile systems," Proc. of Internatıonal Workshop on Parallel and Dıstrıbuted Sımulatıon (PADS), pp. 40-49, June 2003.

[17] A. Boukerche, S. K. Das, and A Fabbrı "SWiMNet. A scalable parallel sımulatıon tested for wireless and mobıle networks," ACM Wıreless Networks, pp 467-486, 2001

[18] The network simulator, versıon 2 (ns-2), http://www.isi.edu/nsnam/ns/.

[19] Global mobile ınformation systems simulation library (glomosım), http://pcl.cs ucla edu/projects/glomosim/

VITA

Zhuojing Li was born in Tianjin, China, on September 30, 1974, the daughter of Yunqi Chen and Hongtian Li. After completing her high school study, she entered Tianjin University of Traditional Chinese Medicine in 1993. After five years of study, she received the degree of Bachelor in Medicine. In 2001, she entered Texas State University-San Marcos, pursuing a master's degree in Computer Science.

Permanent Address: 1000 N LBJ Dr., #J8
San Marcos, Texas 78666

This thesis was typed by Zhuojing Li