DESIGN AND SIMULATION OF MULTI-ROOT MULTI-GENERATION PACKAGE 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

Xin Zhang, B.A.

San Marcos, Texas

December 2004

ACKNOWLEDGEMENTS

I would like to thank Dr. Wuxu Peng, my thesis and research advisor, for his year long supervision and contribution. Without his guidance and the hardware equipment he provided, my research work and thesis writing could not have taken place. I owe a huge debt of gratitude to his kindness and patience. My deep appreciation also goes to Dr. McCabe and Dr. Drissi for the corrections and advice they provided to help me go through the thesis process.

This manuscript was submitted on November 1, 2004.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Digitalization goes hand in hand with computer networking. Todays pervasive usage of digital devices dictates the renovation of networking technologies. The emergence of a variety of networking protocols – from 802.11 to Bluetooth, from Wi-Fi (Wireless Fidelity) to cellular networks – as well as the standardizing bodies behind them are all answers to this renovation demand. No matter how different or advantageous one claims itself over others, one common theme is that computer networks have to go wireless. "Wireless" is the buzz word that will rule networks of the next generation.

The concept of Mobile Ad-hoc Network (MANET) [1], spearheaded by the Internet Engineering Task Force (IETF), has emerged in recent years as response to the high demands of the flexibility of digital networks. An ad-hoc network is the cooperative engagement of a collection of mobile nodes without the required intervention of any centralized access point or existing infrastructure. Fueled by the rapid innovation of wireless technologies and the wide availability of wireless devices, MANET has become a research hotbed for computer scientists. Although by its name this type of network does not pose limitations on communication channels, in practice it is often experimented or deployed with packet radio as the transmission media. Therefore, in this thesis I will use the terminologies such as MANET, wireless ad-hoc network, and packet-radio ad-hoc network interchange-

ably.

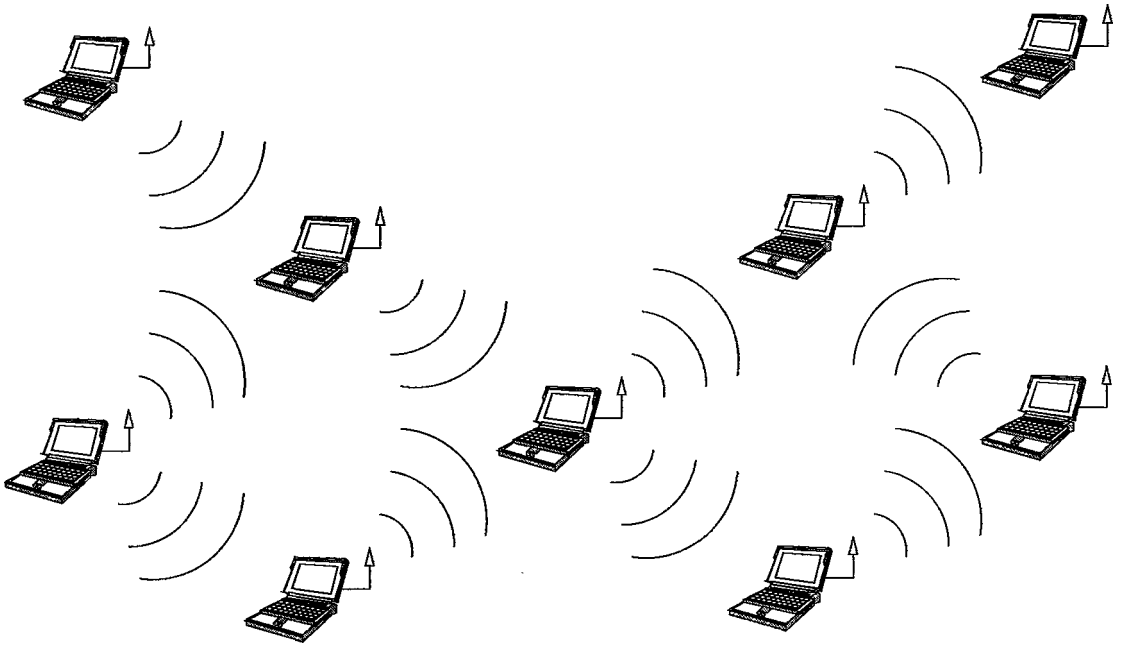


Figure 1.1: Typical MANET, a mobile ad hoc network consisting of a set of mobile hosts that roam at will and communicate with one another. The environment does not host base stations – communication takes place through wireless links. Each mobile host serves as a router, and several hosts may need to relay a packet before it reaches its final destination.

MANET, Figure 1.1, has its unique advantage over traditional wired or wireless networks. By turning every node in the network into an autonomous router, it disposes of the need of deployment of complex routing infrastructures. In particular, MANET does not require base stations; this differentiates it substantially from WI-FI setups and cellular networks. The mobility of the network as a whole and the asynchronous topology adjustment make MANET especially suited for demands of temporary network setups and information exchange over rough terrains. Example scenarios include ad hoc meetings, disaster recovery missions and battle field communications, in all of which traditional methods are either unavailable or disrupted. In general, MANET has the following salient characteristics [2]

- Dynamic topologies: Nodes are free to move arbitrarily; thus, the network

topology – which is typically multi-hop – may change randomly and rapidly at unpredictable times, and may consist of both bidirectional and unidirectional links.

- Bandwidth-constrained, variable capacity links: Wireless links will continue to have significantly lower capacity than their hardwired counterparts. In addition, the realized throughput of wireless communications – after accounting for the effects of multiple access, fading, noise, and interference conditions, etc. – is often much less than a radio's maximum transmission rate. One effect is that congestion is typically the norm rather than the exception.

- Energy-constrained operation: Some or all of the nodes in a MANET may rely on batteries or other exhaustible means for their energy. For these nodes, the most important system design criteria for optimization may be energy conservation.

- Limited physical security: Mobile wireless networks are generally more prone to physical security threats than are fixed cable nets. The increased possibility of eavesdropping, spoofing, and denial-of-service attacks should be carefully considered.

All these characteristics suggest the fact that the existing routing protocols for fixed or semi-mobile networks (e.g. cellular networks) have to be adapted in order to fit themselves in this new environment. That is exactly what researchers have been doing in universities and standardization bodies. As part of Dr. Wuxu Peng's project, we have studied and further redeveloped one such MANET routing protocol. Being an enhancement to the original work of Dr. Chen and Dr. Jia [3], we call it LCMRMG (Locality Caching Multi-Root Multi-Generation) routing algorithm. As the 1000-node simulation shows, LCMRMG achieves significant improvement on the packet delivery ratio, while at the same time keeps minimum sacrifice in areas of end-to-end delay and network maintenance. In comparison to

most studies by other researchers , we chose a network with size of 1000 nodes in response to some envisioned scenario of large networks (e.g. mobile military networks or highway networks).

The following chapters are organized as such:

Chapter 2 is an overview of some existing routing protocols and some new ones that are still under construction.

Chapter 3 is the presentation of LCMRMG algorithm. In this chapter, the entire process of topology setup, maintenance, and message communication will be discussed.

Chapter 4 is the discussion of simulation procedures and the analysis of simulation results.

Chapter 5 concludes the thesis.

CHAPTER 2

ROUTING IN MANET

2.1 Routing in Traditional Networks

Routing is the task of finding a path from a sender to a desired destination. Routing in traditional networks (fixed, or wireless networks) generally takes one of two approaches – distance vector (DV) routing or link-state (LS) routing.

Distance vector routing, as represented by Routing Information Protocol (RIP) [4], is often referred to as DBF (distributed Bellman-Ford) protocols, since its entire operation is derived from a shortest path computation algorithm described by R. E. Bellman [5] and its distributed version attributed to Ford and Fulkerson [6]. In distance vector routing, each router maintains its own routing table (vector), which describes the cost (e.g. hop distance) and path (next hop) to all the possible destinations originating from itself. These tables are periodically updated by messages exchanged between neighboring routers (30 seconds as specified by [4]). In early stages of MANET routing experiment, DV algorithm has been adapted to support ad hoc mobile hosts in proposals such as Destination-Sequenced Distance-Vector (DSDV) routing [7]. However, in terms of performance and scalability, DV protocols still are unable to avoid suffering from slow route convergence and a tendency to create loops in mobile environments.

Link state routing protocols, on the other hand, overcome these defects

by allowing routers to store route maps of the entire network. These maps are used to compute accurate and precise routes to anywhere in the network. Open Shortest Path First (OSPF) [8] is the standard LS protocol advertised by IETF. In OSPF, each router maintains a database, referred to as the link-state database, describing the networks topology. These databases are identical on all routers. Any regional topology change causes flooding, a means to propagate the change to all participating routers. Eventually yet shortly, all routers converge to the same database again. In a wired network environment, LS is preferable over DV for several reasons: [9]

- Fast, loopless convergence
- Support of precise metrics and, if needed, multiple metrics
- Support of multiple paths to a destination
- Separate representation of external routes

However, in a wireless situation such as in MANET, the database approach and the flooding scheme could potentially generate huge control overhead. In a network with population N, LS updating generates routing overhead on the order of $O(N^2)$. In large networks, the transmission of routing information will ultimately consume most of the bandwidth and consequently block applications, rendering the network useless. Thus, reducing control overhead becomes a key issue in achieving network scalability. Relevant works attempting to adapt LS routing into MANET include Wireless Routing Protocol (WRP) [10] and Global State Routing (GSR) [11]. In these proposals, the enhancements are mainly focused on avoiding flooding update messages. Instead, updates will be localized to neighboring nodes.

2.2 Routing in MANET

Research in the last decade has produced enormous amount of proposals on how to route messages efficiently and how to keep track of topology changes in the

context of MANET. Along with literatures proposing new protocols, there are myriad of survey and review papers documenting the pros and cons of these protocols [12][13][14][15]. One common theme – it is rather pervasive among MANET routing researchers – is to classify MANET routing protocols into two distinct categories, "proactive" vis-à-vis "reactive" or "table-driven" vis-à-vis "on-demand."

In a proactive approach, the routing algorithm attempts to maintain as much routing information for mobile hosts as possible. Routes are kept and updated in caches on each mobile host in a distributed fashion. The start of routes updates can be driven by topology change, timer timeout or both. In the topology-driven model, the trigger of route updates is in the form of any change in the network topology that is substantial enough to affect the existing routing tables cached by mobile hosts. In the timeout-driven model, the routing protocol is responsible for periodically initiating route updates among neighboring hosts. In contrast, reactive routing protocols determine routes only when there is data to send. If no route to the destination is known at the sender end, a route-searching process will take place and typically a route-request messages will be propagated throughout the network until a route is found at certain point, which then will be piggybacked with an acknowledge message and echoed back to the sender. Most reactive protocols do not exclude usage of routing caches. In fact, some protocols depend heavily on careful maintenance of routing caches to gain superior performance.

As research studies show, neither proactive nor reactive protocols alone can address all the issues pertaining to routing in MANET. Proactive protocols establish routes much faster at the expense of storage and control overhead. On the opposite, reactive protocols obtain routes on an as-needed basis but route-searching procedures degrade network performance. That realized, plus location-awareness services made available through other technologies, many newer routing protocols for MANET have been proposed that simply cannot be classified as either proactive or reactive. Thus, the rest of this chapter will outline some of the

major protocols along a different line of categorization, Figure 2.1:

- Flat routing schemes, where all the original proactive and reactive protocols belong
- Hierarchical routing
- Geographic position assisted routing

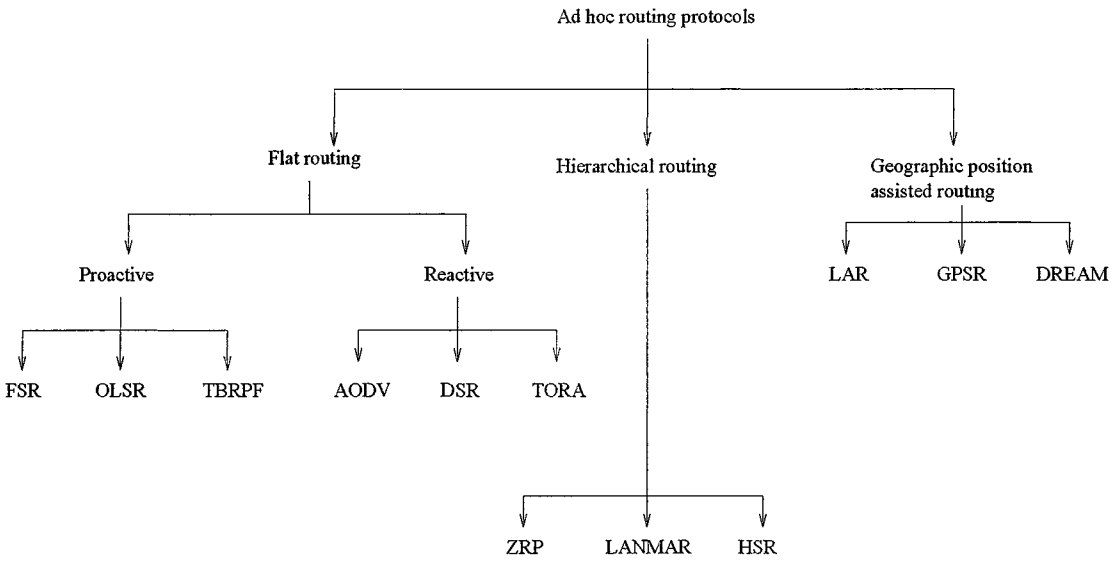


Figure 2.1: Classification of ad hoc routing protocols

Flat routing approaches adopt a flat addressing scheme. Each node participating in routing plays an equal role. In contrast, hierarchical routing usually assigns different roles to network nodes. Some protocols require a hierarchical addressing system. Routing with assistance from geographic location information requires each node to be equipped with the Global Positioning System (GPS). This requirement is quite realistic today since such devices are inexpensive and can provide reasonable precision.

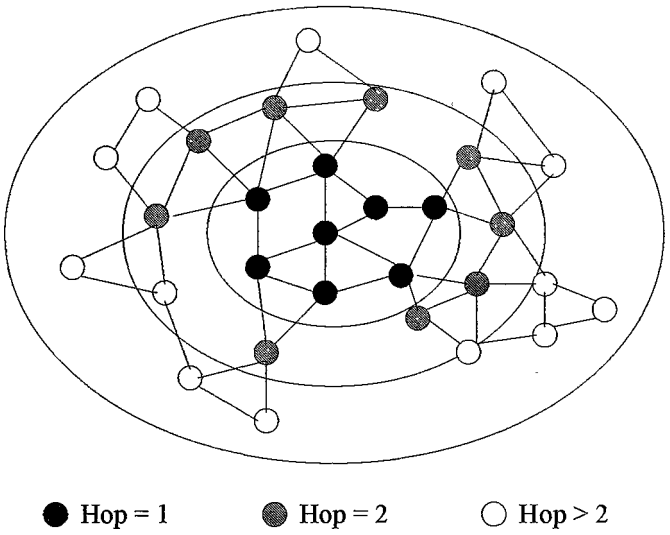


Figure 2.2: FSR: Scope of fisheye

2.2.1 Flat Routing Schemes

Fisheye State Routing (FSR)

Fisheye State Routing (FSR) [16] is a simple, efficient LS type routing protocol that maintains a topology map at each node and propagate link state updates. As opposed to traditional link state protocols, FSR employs different ways to disseminate routing information. First, FSR exchanges the entire link state information only with neighbors instead of flooding it over the network. The link state database is kept up to date based on the information received from neighbors. Second, the link state exchange is periodical rather than event-triggered, which avoids frequent link state updates caused by link breaks in an environment with unreliable wireless links and high mobility. Moreover, FSR includes periodical broadcast of the link state information that serves at different frequencies for different database entries depending on their hop distances to the current node. Updates to entries corresponding to faraway destinations are propagated with lower frequency than those to nearby destinations, Figure 2.2. As a result, a considerable fraction of entries are suppressed from link state exchange packets. In this way, FSR produces accurate distance and path information about the immediate neighborhood of a node, and imprecise knowledge of the best path to a distant destination. However, this

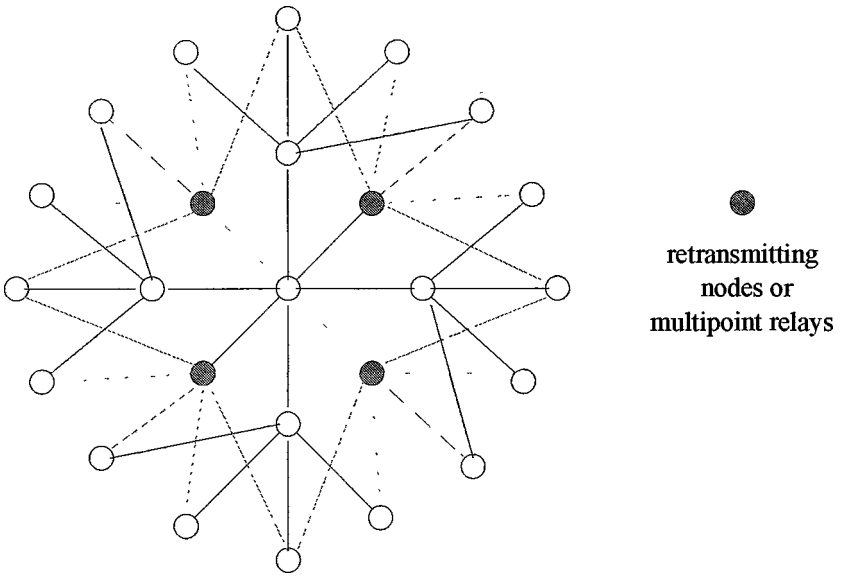


Figure 2.3: OLSR: Multipoint relays

imprecision is compensated by the fact that the route on which the packet travels becomes progressively more accurate as the packet approaches its destination.

Optimized Link State Routing Protocol (OLSR)

Optimized link state routing protocol (OLSR) [17] is another way to adapting link state routing to suit the need of MANET. The optimization happens in two respects. First, OLSR reduces the size of control packet. Instead of all links, it only declares a subset of links with its neighbors who are its *multipoint relay selectors* and transmits topology changes to this subset. Second, it minimizes flooding of this control traffic by using only the selected nodes, called *multipoint relays (MPRs)*, Figure 2.3, to diffuse its messages in the network. Only the *MPRs* of a node retransmit its broadcast messages.

MPRs is a subset of neighboring nodes. That means there are other neighbors who do not belong to a nodes *MPR* set. These non-*MPR* nodes read and process the control packets but do not retransmit them received from the originating node. For this purpose, each node maintains a set of its neighbors which are called the *MPR* selectors of the node. Every broadcast message coming from these *MPR* selectors of a node is assumed to be retransmitted by that node. This

set can change over time, which is indicated by the selector nodes in their HELLO messages.

The *MPR* set of each node is built out of its one hop neighbors in such a manner that the set covers (in terms of radio range) all the nodes that are two hops away. The multipoint relay set of node N, called $MPR(N)$, is an arbitrary subset of the neighborhood of N which satisfies the following condition: every node in the two hop neighborhood of N must have a bi-directional link toward $MPR(N)$. The smaller the multipoint relay set, the more optimal the routing protocol. The optimum (minimum size) *MPR* computation is NP-complete. Efficient heuristics are used.

OLSR protocol relies on the selection of multipoint relays, and calculates its routes to all known destinations through these nodes, i.e. *MPR* nodes are selected as intermediate nodes in the path. To implement this scheme, each node in the network periodically broadcast the information about its one-hop neighbors which have selected it as a multipoint relay. Upon receiving of this *MPR* selectors information, each node calculates and updates its routes to each known destination. Therefore, the route is a sequence of hops through the multipoint relays from source to destination. OLSR is particularly suited for large and dense networks. When the network is sparse, every neighbor of a node becomes a multipoint relay. The OLSR then reduces to a pure LS protocol.

Topology Broadcast Based on Reverse Path Forwarding (TBRPF)

Topology Broadcast Based on Reverse Path Forwarding (TBRPF) [18] is inherently also a link state protocol. TBRPF improves traditional LS routing by introducing a conceptual broadcast tree rooted at the source node of topology update. The correct operation of TBRPF presumes the existence of an underlying routing protocol that can find in finite time the next hop on the shortest path to each destination. This destination in turn will serve as the broadcast source, i.e. the root of the spanning tree for propagating its link state updates. In most cases, these

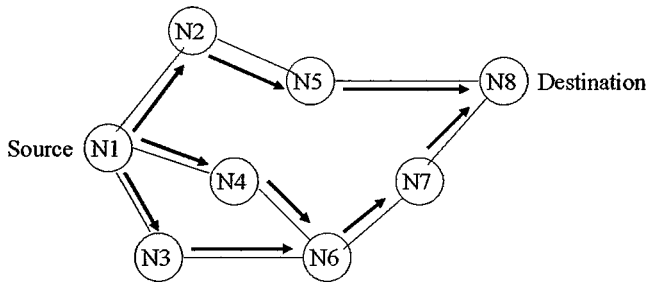
updates are differential, which means only information of changed links will be disseminated to the neighborhood. However, the option is reserved for broadcasting full topology information.

TBRPF works as follows. Assume node i selects the next node $pi(v)$ on the shortest path to each destination v. The node $pi(v)$ then becomes the parent of i on the broadcast tree rooted at source v. Each node informs its parent of this selection, so that each parent becomes aware of its children for each source. A node i receiving a broadcast message originating from source v from its parent $pi(v)$ forwards the message to its children for source v (if it has children). Because the updates are differential, the propagation only follows a small subset of the source trees. Furthermore, the leaves of the broadcast tree do not forward updates, which saves huge amount of bandwidth compared to LS flooding. There are also additional optional configurations, such as full topology broadcast and periodical broadcast. They can be turned on or off depending on the characteristics of the network. But in general, differential updates should occur more frequently than others to ensure in time the route validity.

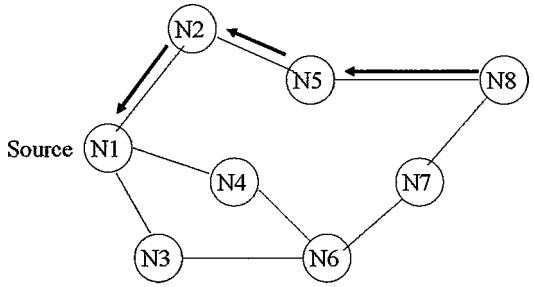
Ad-Hoc On-Demand Distance Vector Routing (AODV)

The Ad hoc On-Demand Distance Vector (AODV) [19] algorithm offers on-demand, hop-by-hop routing from a source to a destination. AODV is a pure on-demand route acquisition protocol. Nodes that do not lie on active paths neither maintain any routing information nor participate in any periodic routing table exchanges. Further, a node does not have to discover and maintain a route to another node until the two need to communicate, unless the former node is offering its services as an intermediate forwarding station to maintain connectivity between two other nodes. This distinguish AODV from a pure DS protocol, which normally uses network-wide broadcast of full topology information.

To keep the freshness of a route, it utilizes sequence numbers embedded in each control message that is transmitted in the route discovery and maintenance



(a) Propagation of the RREQ

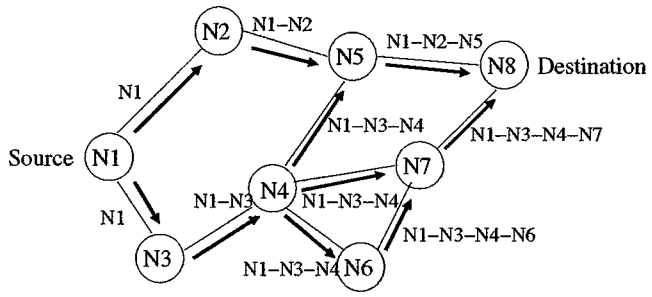


(b) Path of the RREP to the source

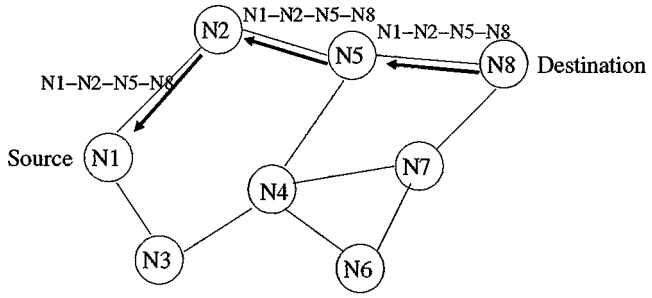
Figure 2.4: AODV: Route discovery

phases. To detect the validity of links between nodes, AODV requires broadcasting periodic beacons between nodes and their neighbors. When a node S needs a route to some destination D, it broadcasts a Route Request ($RREQ$) message to its neighbors, including the last known sequence number for that destination, Figure 2.4. The $RREQ$ is flooded in a controlled manner through the network until it reaches a node that has a route to D. Each node that forwards the $RREQ$ creates a reverse route for itself back to S in its own routing table. When the $RREQ$ reaches a node with a route to D, that node generates a Route Reply ($RREP$) message that contains the number of hops necessary to reach D and the sequence number of D most recently seen by the node generating the $RREP$. Each node that participates in forwarding this $RREP$ back toward S creates a forward route to D. The state created in each node along the path from S to D is hop-by-hop state; that is, each node remembers only the next hop and not the entire route.

As said earlier, AODV normally requires that each node periodically transmit a HELLO message. Failure to receive three consecutive HELLO messages from a neighbor is taken as an indication that the link to the neighbor in question is



(a) Building of the route record during route discovery



(b) Propagation of the route reply with the route record

Figure 2.5: DSR: Creation of the route record

down. If this happens, any upstream node that has recently forwarded packets to a destination using that link is notified via a Route Error (*RERR*) message containing an infinite metric for that destination. When receiving a *RERR*, a node invalidates the appropriate routing table entry. Any new request of this route has to use the route discovery procedure above.

Dynamic Source Routing (DSR)

Dynamic Source Routing (DSR) [20] uses source routing rather than hop-by-hop routing as exhibited in AODV. Each packet to be routed carries in its header the complete, ordered list of nodes through which the packet must pass. The key advantage of source routing is that intermediate nodes do not need to maintain up-do-date routing information in order to route the packets they forward, since the packets themselves already contain all the routing decisions. This fact, coupled with the on-demand nature of the protocol, eliminates the need for the periodic route advertisement and neighbor detection packets usually present in other protocols. The on-demand characteristic of DSR applies to both Route Discovery and

Route Maintenance; that routes not immediately needed will not be discovered or maintained.

Route Discovery is the mechanism by which a node S wishing to send a packet to a destination D obtains a source route to D, Figure 2.5. To perform a Route Discovery, the source node S broadcasts a *RREQ* packet that is flooded through the network in a controlled manner and is answered by a *RREP* packet from either the destination node or another node that knows a route to the destination. To reduce the cost of Route Discovery, each node maintains a cache of source routes it has learned or overheard, which it aggressively uses to limit the frequency and propagation of *RREQ*s. Route Maintenance happens when a packet sender S detects a link a long a source route is broken. In this case, S is notified with a *RERR* message generated by the node at the broken link. Then S can attempt to use any other route to D already in its cache or can invoke Route Discovery again to find a new route.

Temporally-Ordered Routing Algorithm (TORA)

Temporally-Ordered Routing Algorithm (TORA) [21] is designed to minimize reaction to topological changes. A key concept in its design is that it decouples the generation of potentially far-reaching control message propagation from the rate of topological changes. Such messaging is typically localized to a very small set of nodes near the occurrence of a topological change. To accomplish this, nodes need to maintain routing information about adjacent (1-hop) nodes.

The protocol performs three basic functions: (a) route creation, (b) route maintenance, and (c) route erasure. Figure 2.6. During the route creation and maintenance phase, nodes use a *height* metric to establish a directed acyclic graph (DAG) based on the relative height metric of neighboring nodes. In times of node mobility, the DAG route is broken and route maintenance is necessary to re-establish a DAG rooted at the same destination. When link failure occurs at the last downstream link, a node generates a new reference level which results in

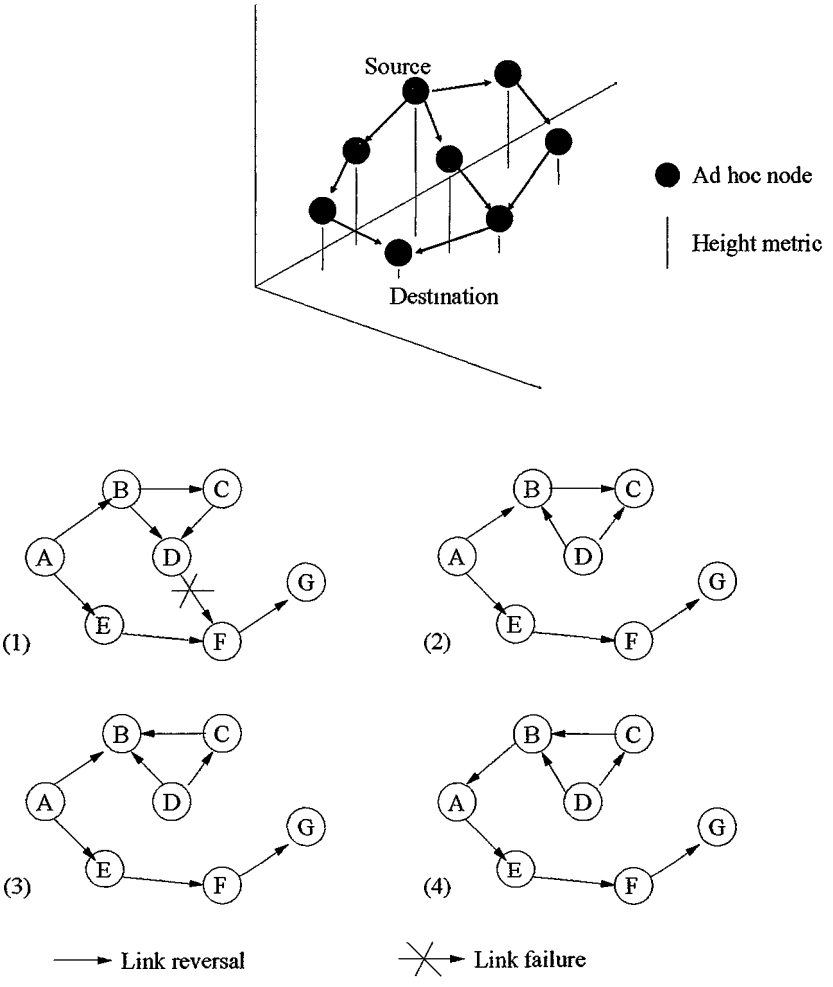


Figure 2.6: TORA: Route creation and maintenance

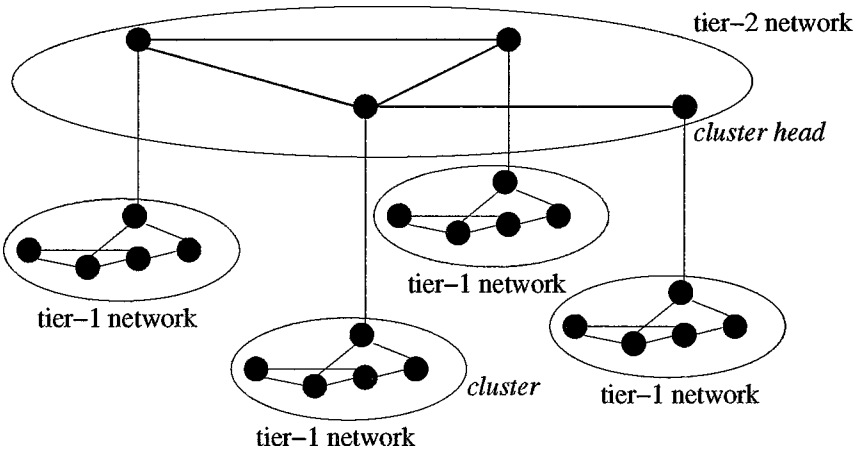


Figure 2.7: ZRP: A 2-tier hierarchy

the propagation of that reference level by neighboring nodes. Links are reversed to reflect the change in adapting to the new reference level.

Timing is an important factor for TORA because the *height* metric is dependent on the logical time of a link failure; TORA assumes all nodes have synchronized clocks. TORA's metric is a quintuple comprised of five elements, namely: (a) logical time of a link failure, (b) the unique ID of the node that defined the new reference level, (c) a reflection indicator bit, (d) a propagation ordering parameter, and (e) the unique ID of the node. The first three elements collectively represent the reference level. A new reference level is defined each time a node loses its last downstream link due to a link failure. TORA's route erasure phase essentially involves flooding a broadcast *clear packet* (CLR) throughout the network to erase invalid routes.

2.2.2 Hierarchical Routing

Zone Routing Protocol (ZRP)

Zone Routing Protocol (ZRP) [22] is a hybrid routing protocol that combines both proactive and reactive routing strategies and benefits from advantages of both. The basic idea of ZRP is that each node has a predefined zone centered at itself in terms of number of hops (zone radius), Figure 2.7. Each node is required to know the topology of the network within its routing zone only and nodes are

updated about topological changes only with their routing zone. This can be achieved by employing any proactive routing protocol within the routing zone. For those nodes outside its zone, it does not maintain routing information on a permanent basis. Instead, on-demand routing strategy is usually adopted when inter-zone connections are required.

Thus, two essential components exist for ZRP – a proactive Intrazone Routing Protocol (IARP) and a reactive Interzone Routing Protocol (IERP). IARP can be any LS or DV routing protocols depending on the implementation. IERP, like AODV or DSR, uses the route query (RREQ) / route reply (RREP) packets to discover routes to destinations. IARP always provides a route to nodes within a nodes zone. When the intended destination is not known at a node, that node must be outside of its zone. As a result, a RREQ message is broadcast via nodes on the border of the zone. Such broadcast is called Bordercast Resolution Protocol (BRP). Route queries are only broadcast from one nodes border nodes to other border nodes until one node knows the exact path to the destination node (i.e. the destination is within its zone).

The behavior of ZRP can be adjusted by changing the value of the zone radius. In particular, for large zone radius, the coverage area is a single zone and ZRP is a traditional proactive protocol. For small zone radius, the protocol is more reactive, and becomes pure flooding at zone radius of one.

Landmark Ad Hoc Routing (LANMAR)

Landmark Ad Hoc Routing (LANMAR) [23] is largely a proactive routing protocol combining elements from both LS and DV schemes. It is also hierarchical in that nodes are designated in groups (subnets) with other nodes that share common interest and are likely to move together, Figure 2.8. LANMAR puts an IP-like address consisting of a group ID and a host ID in the header of each packet. It uses the notion of landmarks to keep track of logical groups. Each logical group has one dynamically elected node serving as a landmark. A global distance

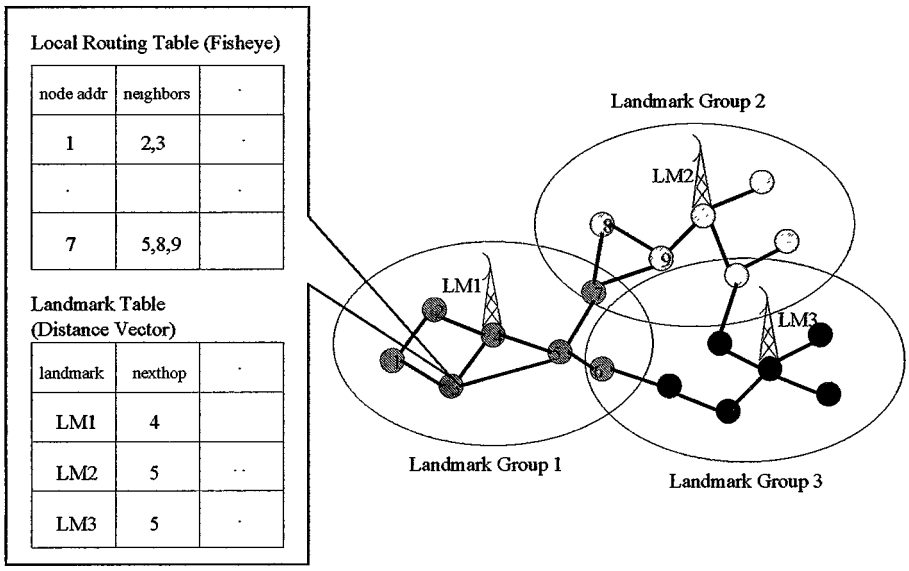


Figure 2.8: LANMAR routing scheme

vector mechanism such as DSDV propagates the routing information about all the landmarks in the entire network. On the other hand, a local flat proactive routing scheme such FSR is used to maintain detailed routing information for nodes within a given group. As a result, each node has all the topology information at hand regarding to destinations within its own group and has a distance vector to all landmarks. When a node needs to relay a packet to a destination within its group, it routes the packet directly. Otherwise, the packet will be routed toward the landmark corresponding to the destinations logical subnet, which is read from the logical address carried in the packet header. When the packet arrives within the group of the destination, it is routed using local tables, possibly without going through the landmark.

In general, by adopting different local routing schemes, LANMAR provides a flexible and scalable routing framework, which reduces both routing table size and control overhead effectively.

Hierarchical State Routing (HSR)

Hierarchical State Routing (HSR) [24] is a multilevel clustering-based LS routing protocol. It maintains a logical hierarchical topology by using the clustering

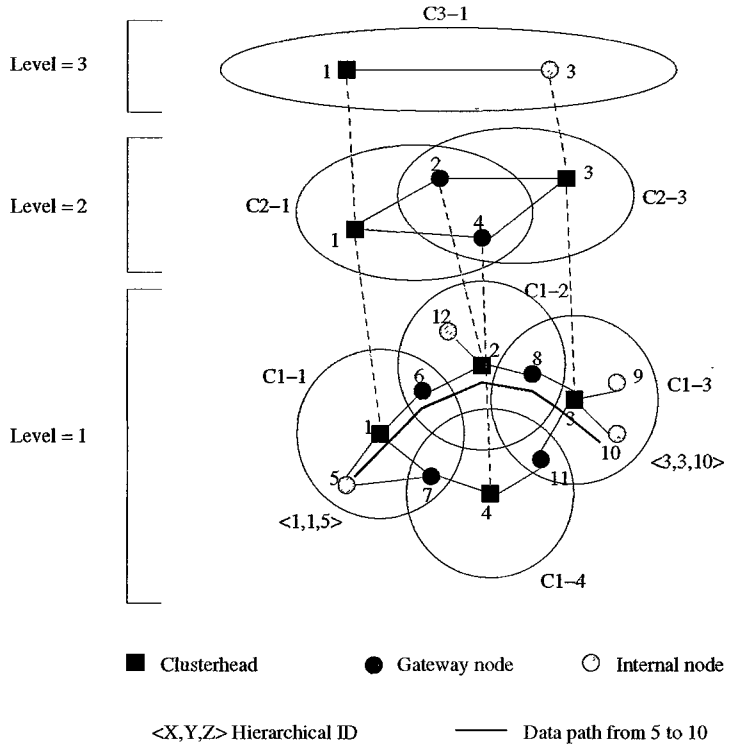


Figure 2.9: HSR: An example of multilevel clustering

scheme recursively. Nodes at the same logical level are grouped into clusters. The elected clusterheads at the lower level become members of the next higher level. These new members in turn organize themselves in clusters and so on, Figure 2.9. The goal of clustering is to reduce routing overhead at each level. Generally, there are three kinds of nodes in a cluster: clusterheads, gateways, and internal nodes. A clusterhead acts as a local coordinator for transmission within the cluster.

At the first level of clustering (also the physical level), each node monitors the state of the link to each neighbor and broadcasts it within the cluster. The clusterhead summarizes link state information within its cluster and propagates it to the neighbor clusterheads. The knowledge of connectivity between neighbor clusterheads leads to the formation of level 2 clusters. Link state entries at level 2 nodes contain virtual links, which can be viewed as tunnels implemented through lower level nodes. Applying the clustering procedure recursively, new cluster heads are elected at each level, and become members of the higher-level cluster. After obtaining the link state information at one level, each virtual node floods it down to nodes of the lower-level clusters. As a result, each physical node has hierarchical

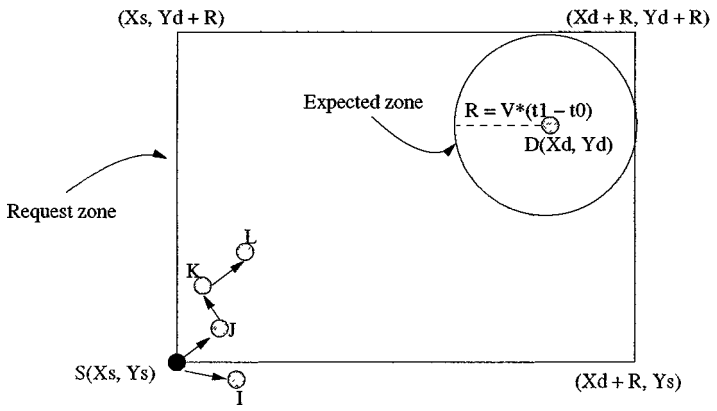


Figure 2.10: LAR: Expected zone and request zone

topology information through a unique hierarchical address (HID) as opposed to a full topology view as in flat LS schemes. HSR defines HID of a node as the sequence of MAC addresses of the nodes on the path from the top hierarchy to the node itself. Due to the page limitation of this thesis and the complexity of HSR itself, I will not get into the details of the HID construction process.

Comparatively speaking, HSR is a rather complicated and novel protocol. Its complexity makes actual implementation difficult. The large amount of hierarchical information and updates associated with nodal movement could potentially produces unbearable overhead.

2.2.3 Geographic Position Assisted Routing

Location Aided Routing (LAR)

Location Aided Routing (LAR) [25] is an on-demand protocol similar to DSR. What makes LAR unique is that it utilizes location information to limit the searching area during the route discovery phase. To be able to operate, LAR assumes each node has the ability to pinpoint its own location upon the assistance of any form of external position services such as GPS (Global Positioning System). Two concepts exist in LAR: *expected zone* and *request zone*. Figure 2.10. *expected zone*, referring to any destination D, is the circular region where D is expected to be reached at the current time. The position and size of this circle is calculated based

on the knowledge of the previous destination location, the time instant associated with the previous location record, and the average moving speed of the destination. *Expected zone* is at best a close estimation since one can never measure the exact moving speed of the destination. However, the inaccuracy can be made up by the idea of *request zone*. *Request zone* is the smallest rectangular region that includes the *expected zone* and the source itself. A node forwards a route request only if it belongs to the *request zone*. *Request zone* effectively limits the route query flooding thus conserves bandwidth and resource usage.

Furthermore, the limited flooding can be guided to follow an even narrower direction by carrying in the message header the geographic position of the destination and the distance between the source and the destination. When a node receives the request, it calculates its distance to the destination. The node will relay the request message only if its distance to the destination is less than or equal to the distance carried in the message header. Before it relays the request, the node will update the distance in the header with its own distance to the destination.

During the network startup phase, it is not possible for a node to know the position of any other nodes. Hence, the traditional flooding method will be used to complete route discovery. One more disadvantage of LAR is in case nodes move fast, the *request zone* could become close to the entire network.

Greedy Perimeter Stateless Routing (GPSR)

Greedy Perimeter Stateless Routing (GPSR) [26] is a routing protocol that uses only neighbor location information in forwarding data packets. It requires only a small amount of per-node routing state, has low routing message complexity, and works best for dense wireless networks. In GPSR, beacon messages are periodically broadcast at each node to inform its neighbors of its position, which results in minimized one-hop-only topology information at each node. To further reduce the beacon overhead, the position information is piggybacked in all the data packets

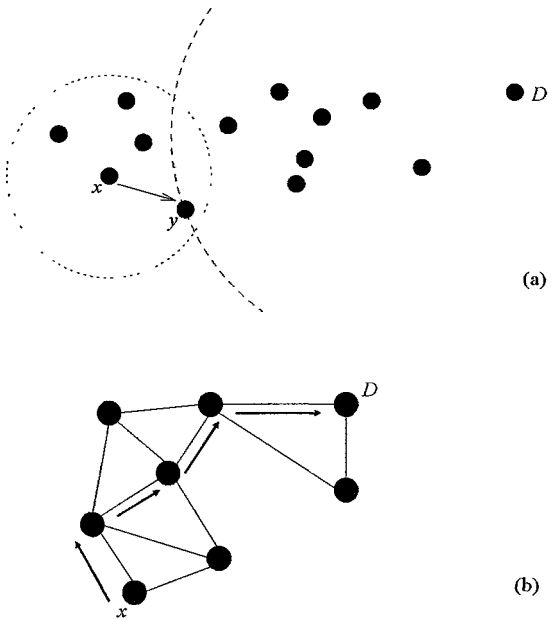


Figure 2.11: (a) Greedy forwarding where y is x's closest neighbor to D. (b) Perimeter forwarding where D is the destination and forwarding hops are solid arrows.

a node sends. GPSR assumes that sources can determine through separate means the location of destinations and include such locations in the data packet header. A node makes forwarding decisions based on the relative position of destination and neighbors.

GPSR consists of two methods for forwarding packets: *greedy forwarding*, which is used wherever possible, and *perimeter forwarding*, which is used in the regions where *greedy forwarding* is not an option. *greedy forwarding* works this way: when a node receives a packet with the destination's location, it chooses from its neighbors the node that is geographically closest to the destination and then forwards the data packet to it. This local optimal choice repeats at each intermediate node until the destination is reached or arriving at a situation in which all of a node's neighbors are farther away from the destination than itself. In the latter case, *perimeter forwarding* will be used, Figure 2.11.

perimeter forwarding requires calculating a *relative neighborhood graph (RNG)*, that is, for all the neighbor nodes, the following inequality holds:

$$\forall w \neq u, v : d(u, v) \leq max[d(u, w), d(v, w)]$$

where u, v and w are nodes, and $d(u, v)$ is the distance of edge (u, v). A distributed algorithm of removing edges violating the above inequality from the original neighbor list yields a network without crossing links and retaining connectivity.

Perimeter forwarding traverses the *RNG* using the right-hand rule hop by hop along the perimeter of the region. During *perimeter forwarding*, if the packet reaches a location that is closer to the destination than the position where the previous *greedy forwarding* of the packet failed, the greedy process is resumed. As far as unreachable destination is concerned, hop counters will be used to avoid infinite routing loops.

Distance Routing Effect Algorithm for Mobility (DREAM)

Distance Routing Effect Algorithm for Mobility (DREAM) [27] is a proactive protocol with the help of location information. DREAM assumes the existence of a mechanism that allows each node to be aware of its own location with respect to a predefined positioning system, typically GPS. These coordinates are exchanged between nodes so that each node constantly obtains location information about the other nodes in the network for routing purposes. In order to be bandwidth and energy efficient, DREAM considers two factors in regard to route update frequency and route lifetime – *distance effect* and *mobility rate*. *Distance effect* states the notion that the greater the distance separating two nodes, the slower they appear to be moving with respect to each other. Hence, nodes that are far apart need to update each others' location less frequently than nodes closer together. This is realized by association with each control message and *age* which corresponds to how far from the sender that message travels. *Mobility rate* indicates that the faster a node moves, the more often it must communicate its location information. This allows each node to self optimize its dissemination frequency, thus transmitting location information only when needed and without sacrificing the route accuracy.

With the location information stored at routing tables, data packets are partially flooded to nodes in the direction of the destination. The source first

calculates the direction toward the destination, then it selects a set of one-hop neighbors that are located in the direction. If this set is empty, the data is flooded to the entire network. Otherwise, the set is enclosed in the data header and transmitted with the data. Only nodes specified in the header are qualified to receive and process the data packet. They repeat the same procedure by selecting their own set of one-hop neighbors, updating the data header, and sending the packet out. When the destination receives the data, it responds with an ACK to the source in a similar way. However, the destination will not issue an ACK if the data is received via flooding. The source, if it does not receive an ACK for data sent through a designated set of nodes, retransmits the data again by pure flooding.

CHAPTER 3

LOCALITY CACHING MULTI-ROOT MULTI-GENERATION ROUTING (LCMRMG)

The design of LCMRMG [28] is largely inspired and assisted by the original spanning tree based protocol presented in [3]. As a further improvement of the original protocol, LCMRMG holds at least the same assumptions of the basic radio channel availability and channel link properties (e.g. bi-directional links). LCMRMG also assume that all nodes in the network act in a cooperative way, that is, any node will not fail their duties in sending and receiving valid messages. In this chapter, I will describe the detailed operation of the LCMRMG protocol. However, prior to that an introduction to the original protocol is in order.

3.1 Spanning Tree Based Routing

In their proposal of a package routing algorithm for MANET [3], Chen and Jia elaborate the ideas of using 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, which could be any node within the network, and its offsprings. Besides being a member of the spanning tree, each node also records locally its generation number *(GEN)* and child ID *(CID)*. The generation number, in terms of a tree structure, is the depth of its position in the tree comparing to the root. Child ID is used to keep track of individual nodes within the same depth level.

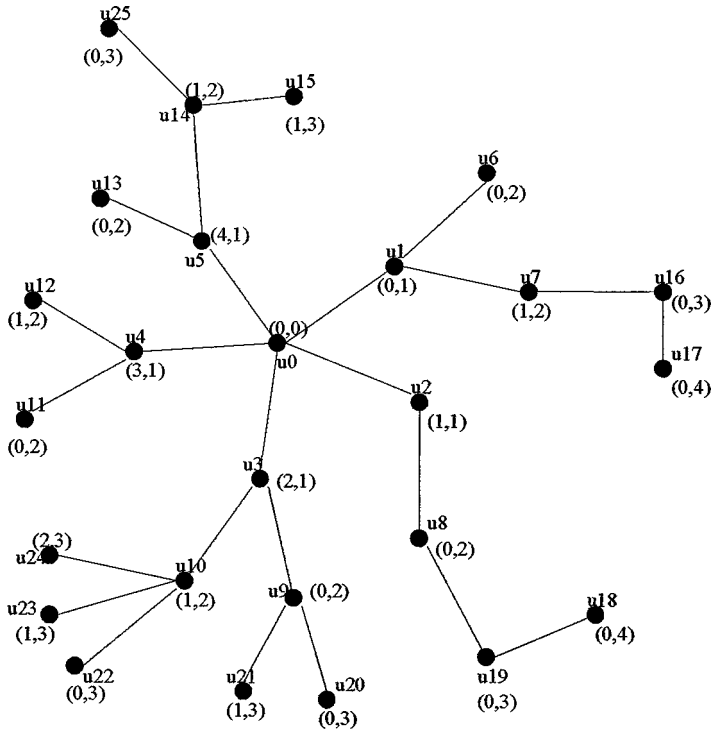


Figure 3.1: Spanning tree based mobile network

In Figure 3.1, $u0$ acts as the root of the network. The label $(0, 0)$, in the form of (*CID*, *GEN*), denotes that $u0$ has a *CID* of 0 and a *GEN* of 0. It can be seen that $u0$ has 5 direct children $u1, u2, u3, u4, u5$, all of which have their *GEN* equal to 1. Their *CID*s are a sequence of integers starting from 0. Applying the same rule, $u1$ has children $u6$ and $u7$; $u2$ has child $u8$; $u3$ has children $u9$ and $u10$, and so forth.

3.1.1 Spanning Tree Construction

The generation number and child ID information do not come with each mobile host. They are assigned to each node in a top-down fashion during the network startup phase. In the foregoing example, $u0$ elects itself as the root node. Therefore it will broadcast soliciting messages requesting other nodes to join him (i.e. the tree). Those nodes within the $u0$'s radio transmission range will respond with a ready signal if they are not yet part of the ST, and in return $u0$ will assign them with sequentially increasing CIDs and a GEN equal to $GEN(u0) + 1$. At this point, the ST has depth 2 and rooted at $u0$ with leaf nodes $u1 \ldots u5$. Once signed on to the ST, $u1 \ldots u5$ will start soliciting their own children just as $u0$ has done previously. For nodes that are out of range from $u0$ and missed $u0$'s request, chances are that they may be able to join the ST as $u0$'s grandchildren. The same procedure will be carried out again and again recursively until the ST reaches all the nodes in the network. This can be determined by presetting a wait period when no leaf nodes are able to receive more ready messages, i.e. all nodes have joined the ST. By then, the topology will finally comes into existence as shown in Figure 3.1.

As soon as the network topology is realized, the leaf nodes will start the process of creating generation tables *(GT)*; thereafter, the GTs will be propagated upward along the tree branches until reaching the root. Take $u25$ in Figure 3.1 as an example. Once $u25$ concludes itself a leaf node, it will build a GT containing its parent ID ($u14$), its GEN (3), its CID (0), its own ID ($u25$), and its future generation information. Since $u25$ is a leaf node, the last field of GT (future generation) will be left empty. See Table 3.1.

$u25$'s parent's ID: $u14$		
$u25$'s ID: $u25$	GEN: 3	CID: 0
$u25$'s future generation		
NIL		

Table 3.1: Node $u25$'s generation table

Next, $u25$ will send this GT to its parent $u14$, and $u14$ in turn will start building its GT. $u14$'s GT will be similar to $u25$'s with appropriate adjustments. However, the future generation field will contain a list of $u14$'s offspring's IDs. $u14$'s GT will not be completed until all the GTs from its children have been processed. In other words, $u14$'s GT have to contain information sent from $u15$ as well. See Table 3.2.

$u14$'s parent's ID: $u5$		
$u14$'s ID: $u14$	*GEN*: 2	*CID*: 1
$u14$'s future generation		
CID	*ID*	Future Generation
0	$u25$	NIL
1	$u15$	NIL

Table 3.2: Node $u14$'s generation table

In a similar way, $u5$ builds its GT as Table 3.3; $u0$, Table 3.4.

$u5$'s parent's ID: $u0$		
$u5$'s ID: $u5$	*GEN*: 1	*CID*: 4
$u5$'s future generation		
CID	*ID*	Future Generation
0	$u13$	NIL
1	$u14$	$u25$, $u15$

Table 3.3: Node $u5$'s generation table

$u0$'s parent's ID: NIL		
$u0$'s ID: $u0$	*GEN*: 0	*CID*: 0
$u0$'s future generation		
CID	*ID*	Future Generation
0	$u1$	$u6$, $u7$, $u16$, $u17$
1	$u2$	$u8$, $u19$, $u18$
2	$u3$	$u9$, $u10$, $u20$, $u21$, $u22$, $u23$, $u24$
3	$u4$	$u11$, $u12$
4	$u5$	$u13$, $u14$, $u15$, $u25$

Table 3.4: Node $u0$'s generation table

3.1.2 Spanning Tree Maintenance

MANET is characterized by fluid topology change. Links between any pair of nodes (routers) are likely to be formed or broken at unpredictable moment. The *ST* based algorithm deals with the changing situation by taking into account three different scenarios – *nodal sign-on*, *nodal sign-off*, and *nodal movement*.

Sign-on

The nodal sign-on procedure is quite similar to that used during the *ST* construction process. The sign-on node in question 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 *ST*. The sign-on node will choose from the responders the one with the lowest generation as its parent. By doing that, the sign-on node becomes a new leaf node in the existing *ST*. As other leaf nodes did, 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.

Sign-off

When a node that belongs to the *ST* 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. *ST* based algorithm proceeds this way. 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 all the node's future generation. Following is again the bottom-up propagation of *GT*s until reaching root, the *GT*s 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 *ST*. 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 *ST*, and begins the process of signing on to the *ST* as

a new node.

Movement

In the ST based algorithm, before its 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. In this way, the relatives, upon receiving start-of-move notification, will wait for the end-of-move message. 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 taken are the same as if the moving node signed off the network, as described earlier. The moving node itself will try to join the network again as a new node.

3.1.3 Spanning Tree Based Routing

ST facilitates MANET routing by proactively maintaining routes. A node makes decisions locally as regards the next hop destination.

Routing Algorithm:

REPEAT

1. The current station w checks its GT to see if the 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 its offsprings.
3. If not, send package m to w's parent y.

UNTIL package m reaches destination v.

Using Figure 3.1, suppose $u8$ is to forward a packet to $u10$. $u8$ checks its GT and finds that $u10$ is not one of its offsprings, i.e. $u10$ does not belong to the subtree rooted at $u8$. Thus $u8$ sends the packet to its parent $u2$. $u2$ checks its GT and finds that $u10$ is not one of its offsprings. $u2$ then sends the packet to its parent $u0$. $u0$ checks its GT and finds that $u10$ is one of its offsprings and specifically down the branch of $u3$. Thereby $u0$ sends the packet to $u3$. $u3$ in turn checks its GT and forwards the packet to $u10$, the destination.

Compared to other routing protocols, the proactive nature of this routing scheme assures zero time of route acquisition because no route probing steps need to be performed in order to send a message. Because the sign-on procedure used in the protocol, i.e. signing on to the lowest generation possible, the protocol guarantees an optimal route from any source to any destination. These two characteristics are essential for time-critical applications in a wireless mobile environment. However, there is a serious drawback in the present approach. By grouping nodes into generations, the spanning tree inevitably introduces imbalance among the nodes in terms of the routing information storage and distribution of traffic volume. The lower the generation a node resides at, the more resource is needed to maintain routing information, and in general the more network traffic it will oversee. If the network size becomes significantly large, some nodes, such as the root of the spanning tree, will suffer severe traffic congestion and performance loss, and the entire network is likely to slow down or come to stall. Besides, to improve network performance, the number of maintenance messages has to be carefully controlled or reduced. As a remedy of and an enhancement to the protocol, in next section we propose Locality Caching Multi-Root Multi-Generation (LCMRMG) routing, and in next chapter an LCMRMG simulation environment and simulation results are presented.

3.2 Locality Caching Multi-Root Multi-Generation Routing (LCMRMG)

LCMRMG routing is based on an intuition that by turning the existing single ST structure into a multiple rooted graph, routing overhead could be successfully distributed. Each root in the graph is in charge of its own ST resembling the single root in [3]. The relationship between different STs are symmetric, i.e. they are not directly related to one another. However from a node's point of view, these STs are largely overlapped. Any node at any time could belong to more than one ST; its generation number (GEN) varies from ST to ST. This implies that the root in one ST could be a far descendent in another ST. Within any ST singled out from the graph, the maintenance operations – sign-on, sign-off, movement – proceed as stated in the previous section. Mobile nodes store the multiple ST information locally, through which multiple routing path can be provided.

3.2.1 Motivation: Caching traffic locality for better performance

More often than not, network traffic in MANET tends to spike in certain network portions. It would be a good idea to find a regional route to cope with such situations if such a route shortens the hops while affecting the rest of the network to a minimum degree. It is especially so in a spanning tree mobile network. To illustrate, consider Figure 3.2, where node r is a current root host that covers hosts s, a, and b. Host a is a descendent of host s. Host b is not in the subtree rooted at host s. Therefore when a and b communicate with each other, the packets are routed $a \leftrightarrow s \leftrightarrow r \leftrightarrow b$. The total distance of this route is $l_1+h_1+h_2$. (In MANET bigger distance usually implies more hops from a start node to the destination.) We observe the following.

> If host s has to function as a router for a high volume of network traffic that flows through it using routes similar to that of $a \leftrightarrow s \leftrightarrow r \leftrightarrow b$,

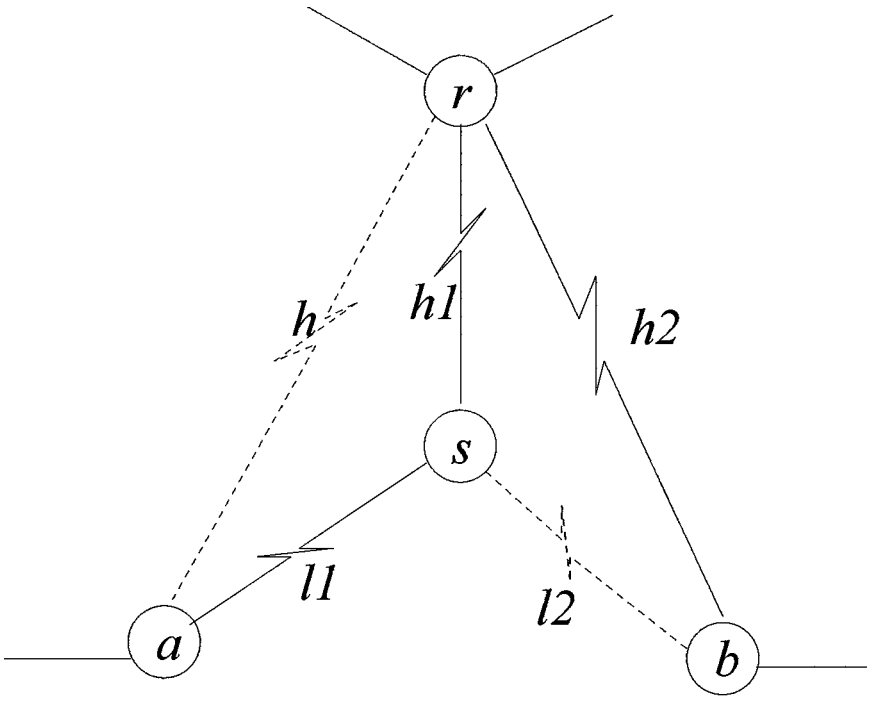


Figure 3.2: Traffic locality in spanning tree MANET

enabling host s to function as a root that covers both host a and b could significantly reduce the network traffic volume. Once s functions as such a root host, the previous route between a and b will be replaced by route $a \leftrightarrow s \leftrightarrow b$. The distance for this new route is $l_1 + l_2$ which in general is smaller than $l_1 + h_1 + h_2$.

It can be seen from above discussion that in order for host s to know that it can reduce network traffic by becoming a new root node, it has to be aware of its location, as well as location information of hosts r, a, and b. Thus we assume as a premise of our algorithm that each mobile node has the capability to know its own location information. Given the current state of technology such as GPS and possibly others, this assumption can well be satisfied. In order for host s to know the values l_1, l_2, h_1 and h_2, the algorithm requires that each message carries three more pieces of information: the positional coordinates of the source host, the positional coordinates of the destination host, and the positional coordinates of the root node r. The acquisition of these coordinates will be explained below. For each message routing through a host like s, the host performs a simple calculation

on the possible savings if it were a root node. Traffic locality statistics collected by such calculations over a period of time is used by the host to decide if it should become a new root node.

3.2.2 Algorithm for traffic locality caching

Let's consider a particular communication session between two hosts like host a and b in Figure 3.2. Assume 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_x, a_y), as well as 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 it sends to the peer.

In order to help locality caching, every root host like node r in Figure 3.2 will simply 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 l_1, l_2, h_1, h_2. Thus we have the locality caching algorithm.

Locality Caching Algorithm:

1. Each host maintains a counter variable ct that records the number of communication packets it 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 $l_1 + h_1 + h_2$ as illustrated in Figure 3.2. For each packet that goes through s, the variable α is modified:

$$\alpha = \alpha + (l_1 + h_1 + h_2)$$

(b) The variable β records the accumulated distance values $l_1 + l_2$ as illustrated in Figure 3.2. For each packet that goes through s, the variable β is modified:

$$\beta = \beta + (l_1 + l_2)$$

The algorithm is based on the following assumptions:

- Each mobile host knows its own coordinates (x, y) through a technique such as GPS.
- Each *request* message carries the coordinates of the source host.
- Each *reply* message carries the coordinates of the requesting mobile host, as well as those of the reply sender. The reply sender simply first inserts the coordinates of the requesting host it previously saved and then inserts its own coordinates as part of the *reply* message.
- For each incoming message, a root host inserts is own coordinates before forwarding the message along appropriate outgoing link.

3.2.3 Algorithm for new root election

Still using the example in Figure 3.2. We can conclude that if compared to $l_1 + h_1 + h_2$, $l_1 + l_2$ is unproportionately small, host a and host b must be very close. Thus to make the route from a to b going through s instead of $s \leftrightarrow r$ is fairly

valuable in saving bandwidth and shortening route distance. Moreover host r can be spared from participating the routing process for its own good. Since α is the cumulation of $l_1 + h_1 + h_2$ and β is of $l_1 + l_2$, 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, we derive algorithm for new root election.

Algorithm for new root election:

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 $\frac{(\alpha-\beta)}{\alpha}$.

2. If the following two relationships hold

$$\alpha > \beta$$

$$\frac{(\alpha - \beta)}{\alpha} > \gamma$$

   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 subtree rooted at itself to become the new ST by propagating NEW signal throughout the subtree. The propagation is an up-down, generation-by-generation recursive process common in the ST maintenance operation.

Once there are more than one root node in the network, the topology maintenance operations proceed as usual except that they must take into account all the existing STs when changing topology status. In other words, when a node signs on to the network, it will attempt to sign on as many STs as it can; when a node signs off, it will also take care of all its status changes in every ST it belongs to.

The value γ is chosen to control the degree of difficulty in creating new root hosts. It address the issue stated earlier on the value that β should be. A smaller value of γ ($0\% < \gamma < 100\%$) indicates new root nodes can be more easily created. In our simulation if $\gamma > 50\%$, new roots are rarely created. With $\gamma < 30\%$, large number of roots are usually generated very quickly. The issue of the number of root hosts needed to support efficient network operations will be discussed in the simulation chapter.

The root termination issue was not addressed in the single root ST algorithm. In fact if the single root terminates, the entire network will cease to exist; all the living nodes will have to rebuild the network from scratch. In LCMRMG, the availability of multiple root nodes and multiple routing paths renders the root termination a non-issue. A root in LCMRMG can sign off the network just as other regular nodes without affecting the network in a whole. In this sense, LCMRMG is much more robust and reliable than the original ST algorithm.

One more point to note is that the subtree salvaging process preserves existing topology information and routes. Instead of being released, the old subtree rooted at the new root becomes the first members of the new ST, thus many potential sign-on operations are avoided.

3.2.4 LCMRMG Routing

Putting previous discussions together, LCMRMG can be viewed as a composition of four distinct operations – ST maintenance, nodal locality caching, new root election, and packet forwarding. ST maintenance, as in [3], consists of operations

to construct ST and GT, operations for mobile hosts to sign on to the network, operations for mobile hosts to sign off the network, and operations for mobile hosts movement. One difference of LCMRMG is that all of these operations have to be extended to accommodate multiple root nodes and STs.

Packets in LCMRMG are routed in a similar way as in a single root scenario, except that a decision has to be made when there are more than one forwarding path available, since any mobile host may simultaneously belong to multiple STs. The strategy we adopted is to choose the path along the ST in which the forwarding host has the lowest generation number. The reason is that in general lower generation nodes cover more offspring nodes and have wider routing knowledge than higher generation nodes. By forwarding along low generation paths, routing hops can be significantly reduced.

LCMRMG Routing

Every host in LCMRMG MANET:

1. performs ST maintenance operations.
2. for each incoming packet:
   (a) performs the algorithm for traffic locality caching.
   (b) routes the packet accordingly:
      i. if the destination host of the packet is in one of its subtrees, forward the packet to the root of that subtree. If there is a tie, choose the one with relatively low generations.
      ii. otherwise forward the packet to the parent host that has the lowest generation number.
3. performs the algorithm for new root election.

Coming back to Figure 3.2, if host s becomes a new root node, the ST maintenance operations dictate that host b will join the the new ST rooted at s with a lower generation number. Therefore two paths now exist from a to b, that is $a \leftrightarrow s \leftrightarrow r \leftrightarrow b$ and $a \leftrightarrow s \leftrightarrow b$. Based on LCMRMG algorithm, s will choose the lower generation path to forward packets. As a result $a \leftrightarrow s \leftrightarrow b$ will be used instead of $a \leftrightarrow s \leftrightarrow r \leftrightarrow b$, the goal that we have set out to achieve.

CHAPTER 4

LCMRMG SIMULATION

Protocol simulation is a rather popular approach to verify the validity of protocol proposals. It is especially so in the MANET community since by far there are still no prevailing affordable MANET applications in the market with which to experiment. In short a simulation is the creation of a set of computer programs that imitate the physical requirements of the proposed protocols, and the observation of the execution of these programs in a digital environment. We proposed LCMRMG as an improvement of the original *ST* algorithm; we also designed and implemented a simulation environment to verify the superiority of LCMRMG.

There are pre-implemented, full-blown simulators for networking in general and MANET in particular. Two prominent ones are *NS-2* [29] and *GloMoSim* [30]. We decided not to use them but rather create our own simulation environment based on two considerations:

- Our main purpose is to verify the enhancements that LCMRMG makes over the original *ST* algorithm. We do not concern ourselves with the physical radio implementation, the medium-access control (MAC) channels, and the real-world packet forwarding mechanism based on IP stacks, all of which constitutes the bulk of aforementioned simulators.

- The learning curve of these simulators could be steep. Users not only have to master some scripting techniques, but also have to understand how to

develop modules that can be plugged into the simulators. In addition, their documentation is often lacking. Using simulators requires time-consuming and effort-exhausting dedication.

In the rest of this chapter, I will first describe the design of our simulation environment, then present the statistical data output from the simulation and the analysis thereof.

4.1 The Spanning Tree Based Simulator (*ST-SIM*)

ST-SIM was created by the author from scratch on Unix platforms. It is written entirely in *C* language and built with GNU *C* (*GCC*) compiler. The sole purpose of *ST-SIM* is to implement the operations of the original *ST* routing algorithm and the LCMRMG modifications. Although, as a network routing protocol, LCMRMG involves participation of multiple mobile hosts, *ST-SIM* is designed to execute on a single Unix box and to be invoked from command prompt. To be able to simulate with various scenarios and under different conditions, *ST-SIM* provides a set of command line switches to control the program execution. *ST-SIM* outputs several log files as simulation proceeds, with which users can look into the simulation process and observe the behavior of the simulated protocol. These log files form the basis of post-simulation analysis. There could be better ways for outputting data than writing to log files. However for convenience's sake these files are easy to implement, and scripts can be created on the fly to extract the data embedded in them.

ST-SIM makes extensive use of the *POSIX* thread library, which are usually resident of most Unix boxes. In fact each mobile host in the *ST* is represented by one running thread; network communication is thus transformed to thread communication. Our simulation has successfully generated networks of 1000 mobile hosts (1000 threads). We believe that even more hosts can be added into our

simulation if computing resources and OS limitations, particularly memory space, allow us to do so. However in practice most systems cannot handle scenarios much more than 1000 hosts. Modeling mobile hosts as system threads poses challenges of communication and synchronization. Although API calls allow limited control of this issue, the size of the simulated network does depend on OS implementation. An example is that the OS scheduler decides the running state of each thread/host, which means, against intuition, that for most of the time a host is inactive in terms of thread state, but is active in terms of the simulation. Therefore the simulation may behave somewhat differently across different Unix platforms, such as Linux, FreeBSD and Solaris, against all of which the simulation has been tested. Also the API syntax differs slightly across these boxes too, which does require some changes in the source code.

4.1.1 Data Structures

ST-SIM represents each mobile host with a single thread. To assure thread mutual exclusion, each host has to claim its own execution space. *ST-SIM* achieves this by dynamically allocating nodal spaces based on the number of nodes put into simulation. Thus an index (node ID) of a particular space can be passed to the corresponding thread; the thread will only execute in its own territory and not interfere with others. Furthermore, the indexes will also become indispensable when messages are to be forwarded among different threads.

The actual representation of a node in LCMRMG is expressed in the following *C struct*:

```
struct node {
    pthread_t ptid;     /* thread identifier for this node */
    int nid;            /* ID of this node */
    struct gen_table relat[MAXROOT]; /* generation table for
                                        each root, max 32 */
    uint32_t root_map;       /* which root tables I belong */
    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 */
};
```

`nid` is the aforementioned index value to each node structure. `root_map` is a 32-bit vector holding flags indicating how many and which STs this node is a member of. If the node becomes a new member of a ST, the corresponding bit will be toggled on (1); otherwise, toggled off (0). The size of this vector (32 bits) also indicates the maximum number of roots that can coexist in the simulated network, that is, in the current implementation, the maximum is 32. `location` and `direction`, taking the form of

```
typedef struct {
    double x, y;
} location_t;
```

are the coordinates of the node's current position and the next position. With the help of `speed`, the time used by the node to travel from location a to location b would be

$$\frac{\sqrt{(a.x - b.x)^2 + (a.y - b.y)^2}}{speed}$$

A thread is considered in the moving state within this period of time after leaving the starting point. `traf2`, `r` and `s` are the implementations of ct, α and β in the locality caching algorithm.

The `messages` is a simple implementation of a circular queue structure

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

while 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 statistical purpose and the message has a dummy message body of size 3 bytes. Remember that each message in LCMRMG 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.

`relat` is the most important structure pertaining to topology maintenance at each node. `MAXROOT` is currently set to 32 matching the size of the bit vector `root_map`. Thus only those generation tables with its bit toggled on in the bit vector are valid. Each table looks like this,

```
struct gen_table {
    int pnid;
    int gen;
    int cid;
    int cnum;
    struct node_list *children;
    int rootid;
};
```

resembling Table 3.4 in Chapter 3. Note that `cnum` is the number of next generation children the node has, not the number of its entire offsprings. `rootid` is the index

of the root node that this generation table corresponds to. `children` is a single linked list that in turn contains a second level linked list.

```
struct node_list {
    struct node *self;
    struct id_list *offspring;
    struct node_list *sibling;
};

struct id_list {
    int nid;
    struct id_list *next;
};
```

The idea, like Table 3.4 in Chapter 3, is that each generation table records each next generation child and the entire offsprings spawned from that child.

4.1.2 User Interface

ST-SIM exports 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, *ST-SIM* will keep running until explicitly terminated by the user with *Ctrl-C* command or the time period specified at command line has elapsed. As mentioned earlier, the running *ST-SIM* dumps simulation data into several log files, whose size will grow as simulation proceeds. Each running instance of *ST-SIM* represents a single configuration of the *ST* MANET, hence to simulate different scenarios the *ST-SIM* program can be place into a batch script to automate the entire simulation process.

Command line argument to *ST-SIM* can be illustrated with the following screen shot:

```
nick@butterflies ~/sim_bak> ./main -h

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 .1
        -t <raTio>       ---- (x-y)/x, criteria to make new
                              roots, default to 0.5

nick@butterflies ~/sim_bak>
```

-r specifies whether to simulate a single root or a multiple roots network. -n specifies the simulation time in minutes, upon the elapse of which the program will return. Without using -n the user has to use *Ctrl-C* to terminate the program; otherwise it will go on forever.

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

Sample action.log

```
Node: 89 starts sending message to node 509, with 2 nodes
within node 89's transmission range.
Node: 400 starts sending message to node 906, with 2 nodes
within node 400's transmission range.
Node: 117 starts sending message to node 739, with 1 nodes
within node 117's transmission range.
Node: 132 starts sending message to node 445, with 2 nodes
within node 132's transmission range.
Node: 277 starts sending message to node 409, with 2 nodes
within node 277's transmission range.
Node: 127 starts sending message to node 155, with 1 nodes
within node 127's transmission range.
```

Sample message.log

```
Node 445 (-35.00, -34.00) received a message from node 560
(-30.00, -45.00), with 4 hops.
Node 451 (7.00, -12.00) received a message from node 857
(-5.00, -12.00), with 6 hops.
Node 603 (9.00, -32.00) received a message from node 42
(0.00, -23.00), with 4 hops.
Node 877 (-47.00, -36.00) received a message from node 10
```

```
(-24.00, -46.00), with 7 hops.
Node 123 (27.00, -14.00) received a message from node 900
(34.00, 1.00), with 6 hops.
Node 202 (26.00, 35.00) received a message from node 319
(25.00, 33.00), with 4 hops.
```

Sample monitor.log

```
660 sec.:       sent=52476, received=9733, t_routing=187925,
e_routing=76318, maintenance=698, roots=18
690 sec.:       sent=54816, received=10175, t_routing=196263,
e_routing=79643, maintenance=1180, roots=19
720 sec.:       sent=57156, received=10670, t_routing=204954,
e_routing=83526, maintenance=912, roots=19
750 sec.:       sent=59508, received=11158, t_routing=213596,
e_routing=87597, maintenance=875, roots=20
780 sec.:       sent=61848, received=11675, t_routing=222419,
e_routing=91482, maintenance=1387, roots=20
810 sec.:       sent=64137, received=12173, t_routing=230946,
e_routing=95304, maintenance=1350, roots=21
```

Sample status.log

```
Node 0:
        Location: (48.00, -35.00)
        Direction:(48.00, -35.00)
        Speed: 0.59
        'traf1': 0
        'traf2': 0
        'r': 0.000000
        's': 0.000000
                ROOT TABLE#0
                Root: 0
                Parent: -1
                Generation: 0
                Child ID: 0
                Number of children: 6
                Children are:
        79:     983, 394, 342, 773, 657, 526, 200, 95, 1,
931, 867, 789, 736, 905, 590, 410, 279, 264, 147, 69, 17,
999, 946, 868, 921, 853, 620, 542, 489, 448, 332, 201, 935,
752, 711, 621, 606, 464, 411, 962, 884, 790, 674, 543, 858,
806, 753, 990,
        326:    247, 116, 993, 589, 457, 573, 915, 783, 705,
704, 248,
        536:
        641:
        799:    132, 458, 316, 211, 721, 579, 474, 658,
        851:
```

action.log and *message.log* describe network traffic information. Specifi-

cally the former denotes the source host sending packets out and its surrounding environment; 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. *monitor.log* traces and calculates all the important statistics periodically, which allow us to measure and compare network performances afterwards. In the next section, I will transform these flat text files into two-dimensional diagrams and evaluate the simulation result.

4.2 Simulation Results

To compare the routing performance between LCMRMG and the single-root *ST* protocol and to verify the performance gain of the former over the latter, we have done extensive simulation using *ST-SIM*. The simulation result can be observed with three set of performance matrices – packet delivery ratio, network response, and total of network control messages. Packet delivery ratio is computed by $\frac{Packet\ Sent}{Packet\ Received}$; it is an important metric for measuring network reliability. Network response is represented by the average number of hops each packet actually traveled. A large number of hops usually means long transmission delay and slow response. The total of network control messages, or the maintenance cost, measures the topology efficiency. The smaller this number is, the larger bandwidth can be directed toward regular network traffic and the more likely the topology is amenable to scalability. Besides the three matrices, we have also experimented the influence of network size on LCMRMG, and the efficiency of root population.

Packet delivery in LCMRMG MANET is much more reliable than in a single root network, Figure 4.1. In the former case, since each node can belong to more than one generation tree, on the average each node has better knowledge on routing paths. This knowledge substantially broadens the choices that can be made by any forwarding node when relaying packets, and by the same token avoids to a large extent dropped packets because of not knowing the next hop destination.

By contrast, a single root with one generation tree by itself can hardly incorporate as many nodes into the routing scheme, especially when nodes are free to move about randomly. Even worse, as the simulation shows, the delivery ratio with a single root tends to drop in the long run, which makes it a not very appealing candidate for routing in MANET.

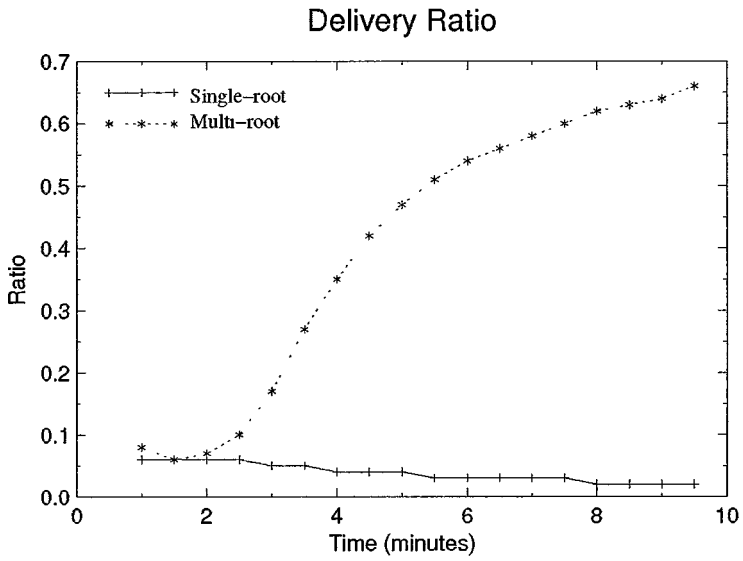


Figure 4.1: Packet delivery ratio

Simulation also reveals that, unlike many other protocols, host number does not constitute a major influence on the performance of LCMRMG in terms of packet delivery ratio. Figure 4.2. This result is particularly desirable in contrast to the single root *ST* protocol, where packet delivery ratio drops as the number of mobile hosts increases.

The average number of hops is related to the signal strength parameter λ in our simulation. This number increases as λ decreases and vice versa. Figure 4.3 shows the average number of hops for both LCMRMG and the single root protocol for a same fixed λ value. Both cases exhibit a fairly leveled plot on the hops number, in the range of 2 to 4 for that λ parameter. To a certain degree of satisfaction, the leveled plots imply network traffic efficiency (loop-free) and stability. With fewer hops, packets can be delivered more quickly thereby throughput will improve. Against intuition, Figure 4.3 also reveals that the average number of hops in LCMRMG is more than that in the single root protocol. However, in

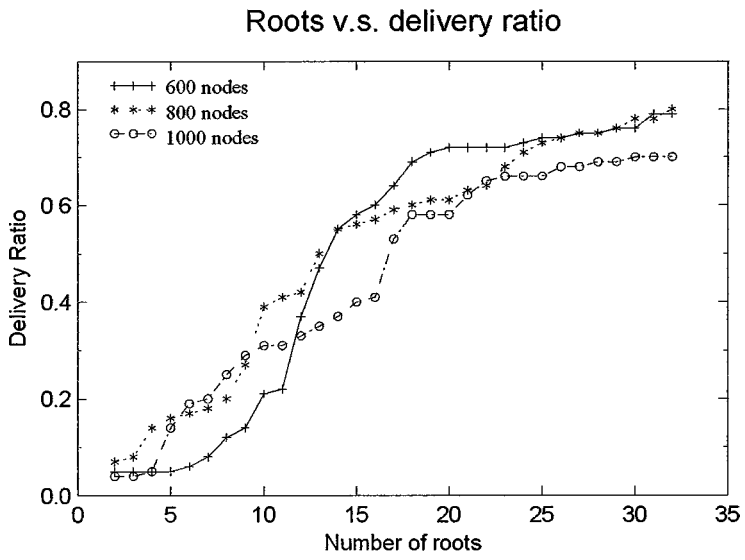


Figure 4.2: Roots vs. delivery ratio

light of the much higher delivery ratio presented previously, this phenomenon can be explained and accepted.

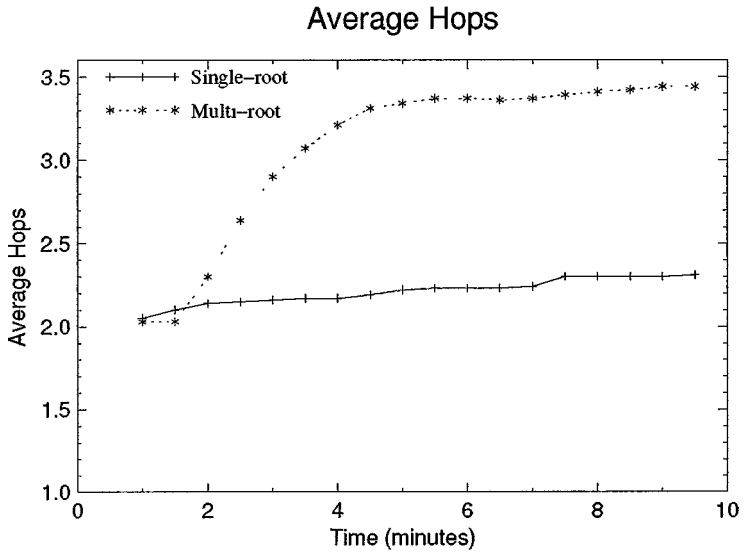


Figure 4.3: Average hops

Adding more roots in the network inherently increases maintenance cost. Figure 4.4. Maintenance cost refers to the control messages needed for nodes to join or leave the network, and for new root creation and termination in the case of LCMRMG. As shown, the number of control messages is linear with respect to the total number of packets delivered. Both lines are quite flat. The difference between the two is rather small in terms of their absolute values. The slightly higher cost

of LCMRMG is clearly offset by the overall performance gain of adding more roots. Furthermore, we notice that as the number of delivered packets goes up, the number of control messages in LCMRMG and in the single root algorithm seemingly converge.

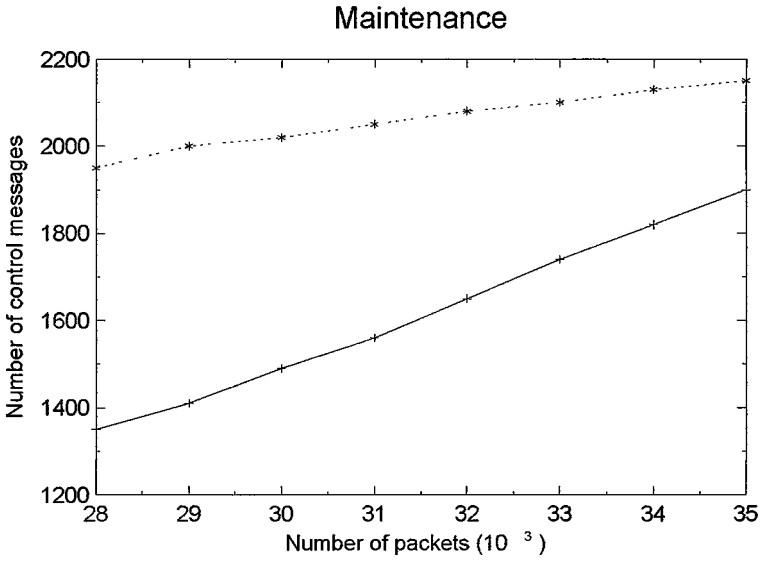


Figure 4.4: Maintenance cost

An interesting issue is the number of root hosts needed to support efficient operations in a given LCMRMG MANET. Intuitively this number is dependent on several factors, including the size of the network (number of hosts), host positions relative to each other, their moving speed, the radio transmission strength, and the geographic spread of hosts. However, once all these factors are fixed, there ought to be a threshold, beyond which adding more roots only increases routing overhead rather than improves performance. Figure 4.5 confirms this, where, in this particular setting, the number of roots actually needed to be sufficient is around five to six.

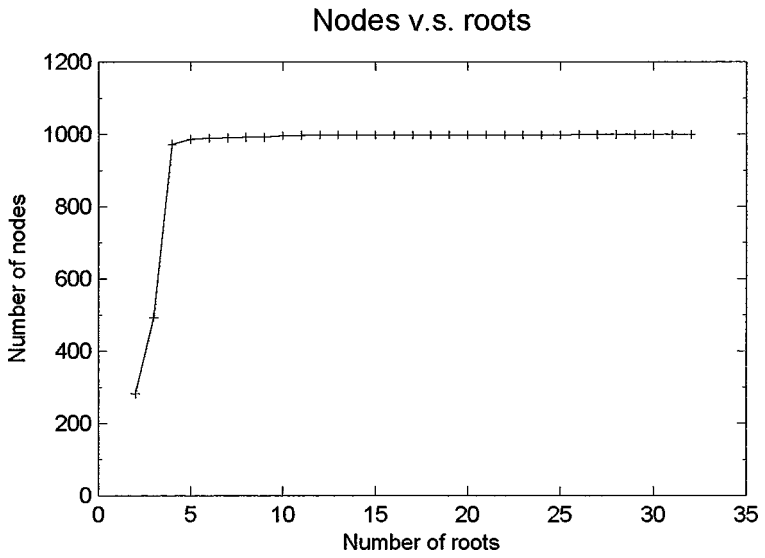


Figure 4.5: Nodes vs. roots

CHAPTER 5

CONCLUSION

In this thesis we developed a multi-root multi-generation spanning tree routing protocol for MANET (LCMRMG) on top of the single-root protocol previously proposed in [3]. In the single-root protocol all the routing information in a given network is maintained and updated by a single-root spanning tree. A generation table is associated with the tree to keep track of relations between mobile hosts. The advantage of the algorithm is its simplicity. A mobile host knows how to route every outgoing packet when needed, without performing any route acquisition procedures. However, this algorithm poses several problems. First, it places heavy loads on low generation hosts, especially when routing demands are high. Second, the possibility of the crash of the root station was not considered. In case of root crash, the topology is rendered useless and it has be reformed from scratch. Third, the performance of the algorithm is problematic. The algorithm did not take into account geographic location information of mobile stations. The spanning-tree structure itself is not sufficient to find an optimal route from a source to a destination. In many cases, the location information of mobile hosts has to be considered to make the best choice of routes and to avoid congesting root and low-generation stations. Based on these observations, we decided that by considering geographical information and thereby introducing multiple roots into the network, all of the above problem can be alleviated. Specifically the existence of multiple

roots splits network traffic from a single root, and the problem of root crash can also be taken care of.

To verify the enhancement made by LCMRMG, we designed a two-protocol simulation environment *ST-SIM* and did extensive simulation test. To our expectation, LCMRMG does outperform the single-root protocol in terms of packet delivery reliability and nodal involvement in packet routing. Although slightly more overhead is introduced on maintaining multiple roots and spanning trees, it can be largely justified by the significant performance gain. On the whole, LCMRMG achieves more reliability, scalability and efficiency.

BIBLIOGRAPHY

[1] Mobile ad-hoc networks (manet) charter. `http://www.ietf.org/html.charters/manet-charter.html`.

[2] S. Corson and J. Macker. Mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations. *Request for Comments (RFC) 2501*, January 1999.

[3] X. Chen and X. D. Jia. Package routing algorithms in mobile ad-hoc wireless networks. *Proceedings of the International Conference on Parallel Processing Workshops (ICCPW)*, September 2001.

[4] G. Malkin. Rip version 2. *Request for Comments (RFC) 2453*, November 1998.

[5] R. Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[6] L. Ford Jr. and D. Fulkerson. *Flows in Networks.* Princeton University Press, 1962.

[7] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *Proceedings of the ACM Special Interest Group on Data Communications (SIGCOMM) Conference*, September 1994.

[8] J. Moy. Ospf version 2. *Request for Comments (RFC) 2328*, April 1998.

[9] C. Huitema. *Routing In The Internet.* Prentice Hall, 1995. Chapter 5, page 106.

[10] S. Murthy and J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *ACM Mobile Networks and Applications Journal*, October 1996.

[11] T. Chen and M. Gerla. Global state routing: A new routing scheme for ad-hoc wireless networks. *Proceedings of the International Conference on Communications (ICC)*, June 1998.

[12] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. *Proceedings of the International Conference on Mobile Computing and Networking (MOBICOM)*, October 1998.

[13] E. Royer and C. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, April 1999.

[14] S. Das, R. Castaneda, and J. Yan. Simulation-based performance evaluation of routing protocols for mobile ad hoc networks. *ACM Mobile Networks and Communications*, September 2000.

[15] A. Boukerche. A simulation based study of on-demand routing protocols for ad hoc wireless networks. *Proceedings of the 34th Annual Simulation Symposium (SS)*, April 2001.

[16] G. Pei, M. Gerla, and T. Chen. Fisheye state routing: A routing scheme for ad hoc wireless networks. *Proceedings of the International Conference on Communications (ICC)*, June 2000.

[17] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. *Proceedings of the International Multitopic Conference (INMIC)*, December 2001.

[18] B. Bellur and R. Ogier. A reliable, efficient topology broadcast protocol for dynamic networks. *Proceedings of the Conference on Computer Communications (INFOCOM)*, March 1999.

[19] C. Perkins and E. Royer. Ad-hoc on-demand distance vector routing. *Proceedings of the Workshop on Mobile Computing Systems and Applications (WMCSA)*, February 1999.

[20] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. *Mobile Computing*, page 153, 1996. Edited by T. Imielinski and H. Korth, Kluwer Academic Publishers.

[21] V. Park and M. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. *Proceedings of the Conference on Computer Communications (INFOCOM)*, April 1997.

[22] Z. Haas. A new routing protocol for the reconfigurable wireless networks. *Proceedings of the International Conference on Universal Personal Communications (ICUPC)*, October 1997.

[23] G. Pei, M. Gerla, and X. Hong. Lanmar: Landmark routing for large scale wireless ad hoc networks with group mobility. *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, August 2000.

[24] G. Pei, M. Gerla, X. Hong, and C. Chiang. A wireless hierarchical routing protocol with group mobility. *Proceedings of the Wireless Communications and Networking Conference (WCNC)*, September 1999.

[25] Y. Ko and N. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. *Proceedings of the International Conference on Mobile Computing and Networking (MOBICOM)*, October 1998.

[26] B. Karp and H. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. *Proceedings of the International Conference on Mobile Computing and Networking (MOBICOM)*, August 2000.

[27] S. Basagni, I. Chlamtac, and V. Syrotiuk. A distance routing effect algirithm for mobility (dream). *Proceedings of the International Conference on Mobile Computing and Networking (MOBICOM)*, October 1998.

[28] W. Peng, X. Zhang, and K. Makki. Locality caching multi-root multi-generation routing algorithm in mobile ad hoc networks. *Proceedings of the International Conference on Computer Communications and Networks (ICCCN)*, October 2003.

[29] The network simulator, version 2 (ns-2). `http://www.isi.edu/nsnam/ns/`.

[30] Global mobile information systems simulation library (glomosim). `http://pcl.cs.ucla.edu/projects/glomosim/`.

VITA

Xin Zhang was born in Zibo, China, on February 5, 1978, the son of Yuhua Zheng and Rui Zhang. In 1995, he went to Peking University in China after high school. After four years of study, he earned the degree of Bachelor of Arts. In 2001, he entered Texas State University–San Marcos, pursuing a master's degree in Computer Science. In Texas State University–San Marcos he has gathered extensive experience in lab assistance. Together with Dr. Wuxu Peng, he has submitted to ICCCN *"Locality caching Multi-Root Multi-Generation Routing Algorithm in Mobile Ad Hoc Networks"* in 2003.

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

This thesis was typed by Xin Zhang.