1-21-2014

Optimizing ad-hoc on-demand distance vector (AODV) routing protocol using geographical location data

Remo Cocco

OPTIMIZING AD-HOC ON-DEMAND DISTANCE VECTOR (AODV) ROUTING PROTOCOL USING GEOGRAPHICAL LOCATION DATA

by

Remo Cocco

A Thesis

Submitted to the

Department of Computer Science

College of Science and Mathematics

In partial fulfillment of the requirement

For the degree of

Masters of Science

at

Rowan University

December 24th, 2013

Thesis Chair: Vasil Hnatyshin, Ph.D

Dedication

I would like to dedicate my thesis to my nieces Talia and Eliana and my nephew William, in the hope that they find something that they love as much as I love Computer Science.

Acknowledgments

I would like to express my gratitude to my advisor, Dr. Vasil Hnatyshin, for his guidance, expertise, patience, motivation, enthusiasm, and immense knowledge. Without Dr. VH, I would never have become a Computer Science major or been able to complete my thesis. I would like to thank the other members of my committee, Dr. Andrea Lobo and Dr. Stephen J. Hartley, for their assistance. I also would like to thank Hristo Asenov for his initial research with GeoAODV.

I want to thank Malik Khaleeque Ahmed and Daniel J. Urbano for their help with my research and for their dear friendship. They say college is where you make life-long friends, and that is definitely the case with these two. Lastly, I would like to thank Joseph P. Smith IV for his help with editing my thesis and being one of my best friends.

Abstract

Remo Cocco
OPTIMIZING AD-HOC ON-DEMAND DISTANCE VECTOR (AODV) ROUTING PROTOCOL USING GEOGRAPHICAL LOCATION DATA
2013
Dr. Vasil Hnatyhsin, Ph.D

This thesis summarizes the body of research regarding location-aided routing protocols for mobile ad-hoc networks (MANET). This study focuses on the use of geographical location information to reduce the control traffic overhead caused by the route discovery process in the ad-hoc on-demand distance vector (AODV) routing protocol. During this process, AODV will flood the entire network with route request packets. This introduces significant packet-handling overhead into the network. This thesis introduces Geographical AODV (GeoAODV), which uses geographical location information to limit the search area during the route discovery process to include only promising search paths. Also, this thesis benchmarks GeoAODV's performance against Location Aided Routing (LAR) and examines four mechanisms for reducing the control-packet overhead introduced by the route discovery process: LAR Distance, LAR Zone, GeoAODV, and GeoAODV Rotate. OPNET Modeler version 16.0 was used to implement each of these mechanisms and compare their performance via network simulations. The results indicate that location-aided routing can significantly reduce the aforementioned control-packet overhead.

Table of Contents

Table of Contents (Continued)

List of Figures

List of Tables

Chapter 1

Introduction

Routing is one of the quintessential components of a computer network's functionality. In a broad sense, it is the process of determining a path to a desired destination. For example, a network administrator could compute all the routes in the network and then deploy them to all nodes manually. However, this approach is impractical and very error-prone for even a small network. As a result, a large number of routing protocols have been created that dynamically determine the routes in the network without human involvement. Without routing protocols, routing information would have to be manually and continually maintained by network administrators; networks would have been unable to scale to the size that it must in order to operate within modern day networks.

After route discovery, the data can be delivered to various destinations in the network on a hop-by-hop basis. As a packet travels through the network, each intermediate node (called a "hop") independently determines the interface on which the packet has to be sent out in order to reach its destination. This process is known as forwarding. It utilizes a data structure known as a forwarding table, which maps a network or node address to the next hop on the route to a particular destination. Forwarding is achieved by consulting the forwarding table to determine the next node (i.e., the outgoing interface) to which a data packet must be sent in order to eventually arrive at the desired destination. This process is repeated at each intermediate node on the packet's path.

The forwarding table is sometimes confused with the routing table. A routing table is built and maintained by a routing protocol deployed in the network. A router could be simultaneously connected to multiple networks, each of which may be running a different routing protocol. Each routing protocol process maintains its own routing table. The forwarding table is an aggregation of all of the routing tables at a particular router. Each time a routing protocol receives control information and updates its routing table, it also instructs the router to update the global forwarding table. Thus, a routing table contains routing information collected by a particular routing protocol. This information is internal to a particular routing protocol process and may be used only by that process. On the other hand, a forwarding table contains routing information collected by all the routing processes deployed on a node. It is used to determine where an arriving packet ought to be forwarded to next. Colloquially, the terms routing table and forwarding table are used interchangeably. This study will adhere to that convention in order to avoid confusion.

Table 1.1 A Sample Forwarding Table

Address	Next Hop
Default	150.250.64.1
150.250.64.0/24	150.250.64.69
224.0.0.0/8	150.250.64.69

The primary intent of any routing protocol is to build and maintain the routing and forwarding tables. A routing protocol is a set of rules that defines how nodes should interoperate and exchange data in order to build routing tables and ultimately achieve efficient end-to-end forwarding. Some popular routing protocols are Open Shortest Path First (OSPF), Intermediate System to Intermediate System (IS-IS), and Interior Gateway Routing Protocol (IGRP).

Generally, routing protocols are classified into one of two categories: Link-State or Distance-Vector. In link-state routing algorithms, each node acts independently to map the entire network topology as a graph. This is achieved by sending information about each node's neighbors to all other nodes in the network. Each node then uses the graph of the network topology it has created to compute the most efficient route to each destination in the network.

In contrast to link-state algorithms, distance-vector algorithms do not distribute routing information to all the nodes in the network. Instead, each node only informs its neighbors about topological changes in the network. These changes are represented as an array of distance-vectors (i.e., known distances from the current node to each node in the network). As a result, distance-vector algorithms are considered computationally simpler and create less control traffic overhead than do the link-state algorithms.

An ad-hoc network is a network that can operate in an environment without preexisting infrastructure and allow for minimal configuration during deployment [1]. In an ad-hoc network, each node acts independently both as an end node and as a router. Each node may send, receive, and forward data traffic. These attributes are very desirable in areas where a network is needed but in which no prior infrastructure exists (e.g., search and rescue, disaster relief systems, military operations).

A Mobile Ad-hoc NETwork (MANET) is a wireless ad-hoc network in which nodes may move. Consider an area that was just stricken by a natural disaster (e.g., hurricane, tornado, tsunami). Natural disasters can wipe out existing networking infrastructure, making communication in the affected area difficult or impossible. The ability of a MANET to function in areas with no prior infrastructure makes them extremely useful in disaster scenarios. However, mobility and lack of infrastructure also present an interesting problem in terms of routing, for which specialized routing protocols are needed.

Routing protocols in ad-hoc networks can be classified into three different categories: proactive, reactive, or hybrid. Proactive routing protocols will actively seek routes to destination nodes, even if there is no traffic traveling through the network. A proactive routing protocol seeks routes in anticipation that they will be needed later. The advantage to proactive routing protocols is that routes are readily available as soon as there is data to transmit. However, they may result in unnecessary overhead when searching for routes

that will never be used. In a MANET environment where computing resource and bandwidth are scarce, this could be a major deficiency that will prevent the protocols from being widely deployed.

On the other hand, reactive protocols only compute routes on demand (i.e., only when a node has data to transmit and the path to the destination is unknown). The advantage of reactive routing protocols is that routes are only computed when they are needed, which minimizes the amount of control overhead introduced into the network. However, the data has to wait while the routing protocol searches for the route. Hybrid routing protocols achieve optimal performance by combining the advantages of reactive and proactive approaches.

There are a large number of routing protocols for MANET environments including Dynamic Source Routing (DSR), Zone Routing Protocol (ZRP), Optimized Link State Routing (OLSR), and Ad-hoc On-demand Distance Vector routing (AODV). This work studies the performance of the AODV protocol and various optimizations that reduce the control message overhead through the use of geographical location information.

AODV, as the name implies, is a reactive distance-vector routing protocol. AODV consists of two primary phases: route discovery and route maintenance. The route maintenance phase is responsible for removing outdated or broken path entries from the

routing table and is of no interest to this study. This work examines the route discovery phase, which utilizes a flooding technique to locate a path to the destination.

AODV only initiates route discovery when a node, often referred to as the originator, receives data from the application layer that is to be delivered to some destination for which there is not a known route. The originator starts the route discovery phase by broadcasting a Route Request (RREQ) message. The RREQ message is rebroadcast by each intermediate node until it reaches either the destination node or a node with a fresh route to the destination. At that point, the node generates a Route Reply (RREP) message back to the originator. The route discovery phase terminates when an RREP message that contains a route to the destination arrives at the originator node. As the RREP traverses the network back to the originator node, it retraces the path of the RREQ message, which was recorded by the intermediate nodes as the RREQ message was traveling through the network. Similarly, intermediate nodes that receive an RREP message update their routing tables with the route to the destination node. Once the route discovery phase completes, the originator node sends data to its destination over the newly discovered path.

Once a route has been discovered and stored, it will only remain in a routing table for a finite amount of time. When a route is stored, it is initially marked as active. Active routes will remain useable either for the Lifetime value received in the RREP message or

for a minimum preconfigured default time period. When the timer eventually expires, the routes are marked for deletion and are scheduled to be removed from the routing table.

AODV uses three types of control packets during the route discovery phase. These control packets are Route REQuest (RREQ), Route REPly (RREP), and Route ERRor (RERR) [2]. An RREQ packet is used anytime AODV needs to discover a route to a specific node. An RREP is used to reply to an RREQ with a definitive route to the node. RERR packets are used to disseminate various error details to other nodes in the network. AODV maintains route entries of its active one-hop neighbors by periodically broadcasting Hello messages with the IP header TTL field set to one. Hello messages have the same format as the RREP messages and can carry the IP address and the destination sequence number for the current node. The sequence number is a unique counter created and maintained by each AODV-capable node. This value is included in all messages that carry routing information. The sequence number represents the freshness of carried data and also prevents routing loops. An AODV node with multiple routes to the same destination is required to select the freshest route (i.e., the route that has the largest destination sequence value). An AODV node increments its sequence number each time it initiates a new route discovery process and whenever it generates an RREP message. This ensures that other nodes in the network can differentiate between RREP messages generated from different route request phases.

In order to reduce the overall control message overhead, AODV employs an expanding ring search technique. The originator node sets the TTL field in the IP header of the RREQ message to a certain initial value. If the route discovery process fails to find a path to the destination, then the originator node increments the value of the TTL field and repeats the process again. This continues until either the originator node finds a path to the destination or the whole network has been searched without finding a path (i.e., an RREQ message with IP TTL field set to the preconfigured TTL threshold value was sent out, but a route to the destination was not found). This search technique prevents unnecessary network-wide dissemination of RREQs.

Despite the expanding ring search technique, the route discovery process in AODV often results in a large number of control packets traveling through the network. This consumes already scarce network resources (e.g., bandwidth, processing power, battery power). Furthermore, anytime there is a demand for a route that is either marked for deletion or does not exist in the routing table, the routing protocol must rediscover the path. This can be costly in volatile MANETs, as the constant change in the topology of the network frequently causes routes to become unavailable. Continuously re-computing routes creates substantial overhead in the network, which will eventually lead to performance degradation. Many enhancements have been suggested to cut down on the number of control packets. Among these are approaches that use the geographical coordinates of each node to predict which paths are most promising. Using geographical coordinates to optimize routing is known as location–aided or geo-assisted routing.

This study examines AODV optimizations that mitigate the aforementioned issues by reducing the scope of the control message broadcast during the route discovery phase. Chapter 2 provides an overview of LAR and other related protocols. Chapter 3 discusses in detail our proposed location-aided improvements to AODV protocol called GeoAODV. The implementation of GeoAODV is covered in Chapter 4. Chapter 5 describes the comparison study of GeoAODV and other location-aided routing protocols using OPNET Modeler Simulation package. The study is concluded in Chapter 6.

Chapter 2

Related Works

Introduction

Geo-assisted routing utilizes the geographical location of the nodes to decrease the amount of overhead in the network, which is created by control packets used during the route discovery process. There are numerous geographically-aided routing protocols for MANETs including Location Aided routing [3, 4], Modified Location Aided Routing [5], Prediction Based Location Update Algorithm [6], AODV Directional Forward Routing [7, 8, 9], Geo-Assisted Multicast Inter-Domain routing [10], and Geographical-AODV [11, 12, 13, 14, 15].

Location Aided Routing (LAR)

Location Aided Routing (LAR) is an extension of the AODV protocol for reducing the control message overhead by using geographical location information [3, 4]. LAR relies on knowing the destination node's coordinates and velocity. There are two main variations of LAR known as *LAR Zone* and *LAR Distance*. These approaches are described below.

LAR Zone

LAR Zone uses geographical location information to identify the area that is likely to contain the path to the destination. The LAR Zone approach defines two areas: the

expected zone and the *request zone*. The expected zone is the area where the destination node is most likely to be located at the time of route discovery. It is defined as a circular area centered at the destination's last known location. The radius of the expected zone is the maximum distance that the destination node could have traveled since the time its location coordinates were obtained, as shown in Equation (2.1):

$$r = v \times (t_1 - t_0) \tag{2.1}$$

where,

- **r** is the radius of the expected zone
- **v** is the average traveling speed of the destination node
- $\mathbf{t_0}$ is the time when the destination coordinates were obtained
- $\mathbf{t_1}$ is the current time

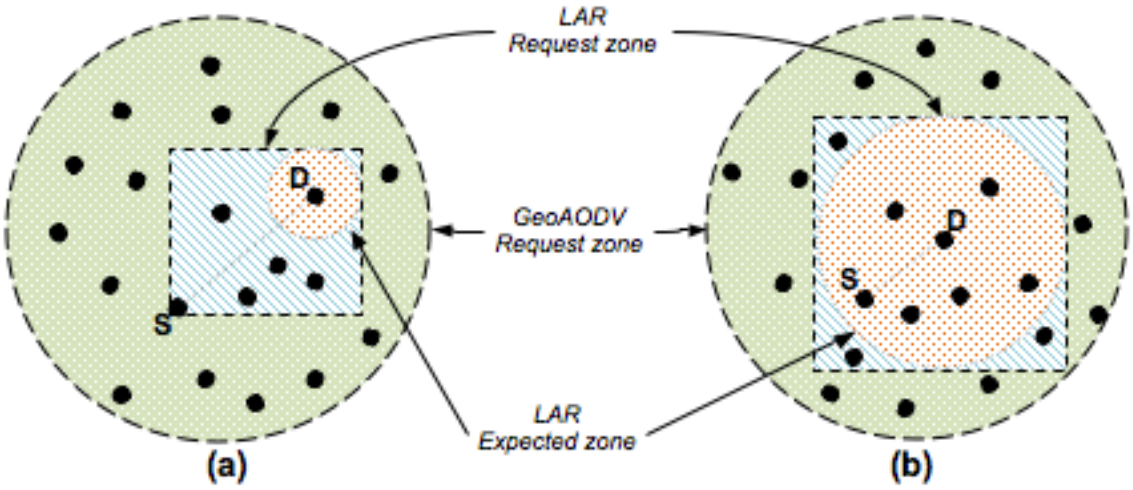

Figure 2.1 Request Zone for AODV and LAR Zone protocols

The request zone is an area that is likely to contain the path to destination. It is defined as the smallest rectangle that contains the expected zone and has its sides parallel to the X- and Y-axes. A possible arrangement of the expected and request zones is shown in Figure 2.1. In Figure 2.1 (a), the source node S is located outside the request zone. In Figure 2.1 (b), the source node S is located within the expected zone for destination node D.

Only the nodes within the request zone participate in the route discovery process and rebroadcast RREQ messages. Specifically, when an RREQ packet arrives at an intermediate node, the node first determines whether or not it belongs to the request zone using the data carried in the incoming RREQ message. This determines whether the regular AODV route discovery processing should then be performed. If the node is part of that search area (i.e., is in the request zone), then the RREQ packet is processed and possibly rebroadcast; otherwise, the packet is discarded.

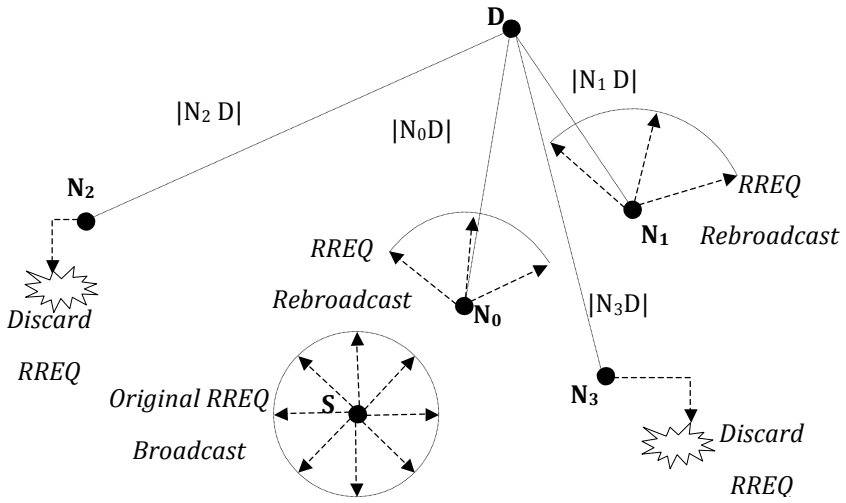

Figure 2.2 Summary of LAR Distance Protocol

LAR Distance

LAR Distance is another variation of location-aided routing protocol based on AODV. LAR Distance relies on the distance between the current node and the destination to determine if the RREQ should be rebroadcast. Consider the scenario illustrated in Figure 2.2. When node N_1 receives an RREQ from node N_0, it checks if it is located closer to the destination than the node from which it received an RREQ (i.e., N_0). If the distance between N_0 and destination D is greater than the distance between N_1 and D, then N_1 rebroadcasts the RREQ; otherwise, the message is discarded. Specifically, the RREQ message is forwarded only if inequality (2.2) holds true:

$$\alpha \times \text{Dist}(N_0\ D) + \beta \leq \text{Dist}(N_1\ D) \quad (2.2)$$

where,

- α and β are configuration parameters
- N_1 is the node which received the RREQ
- N_0 is the node which forwards RREQ to N_1
- D is the destination node

Modified Location Aided Routing

Modified Location Aided Routing (MLAR) is a routing algorithm that optimizes LAR via several enhancements [5]. The first enhancement that MLAR makes is redefining the request zone area. While LAR calculates the request and expected zones relative to the X- and Y-axes, MLAR defines the request zone as the rectangle that is independent of the axes and relative to a line connecting the source and destination nodes. Figure 2.3 (a) shows an example of the MLAR rectangular shaped request zone. The source node determines the area of the MLAR request zone by computing the coordinates of the vertices in the MLAR request zone rectangle. These coordinates are computed relative to the line between source and destination and thus need to be translated into Cartesian coordinates before being used. MLAR relies on equations (2.3) and (2.4) to perform this translation, where (x_1, y_1) denotes the vertex coordinates and l denotes the distance between source and destination.

$$x = x_1 \times (y_D - y_S)/l + y_1 \times (x_D - x_S)/l + x_S \tag{2.3}$$

$$y = x_1 \times (x_S - x_D)/l + y_1 \times (y_D - y_S)/l + y_S \tag{2.4}$$

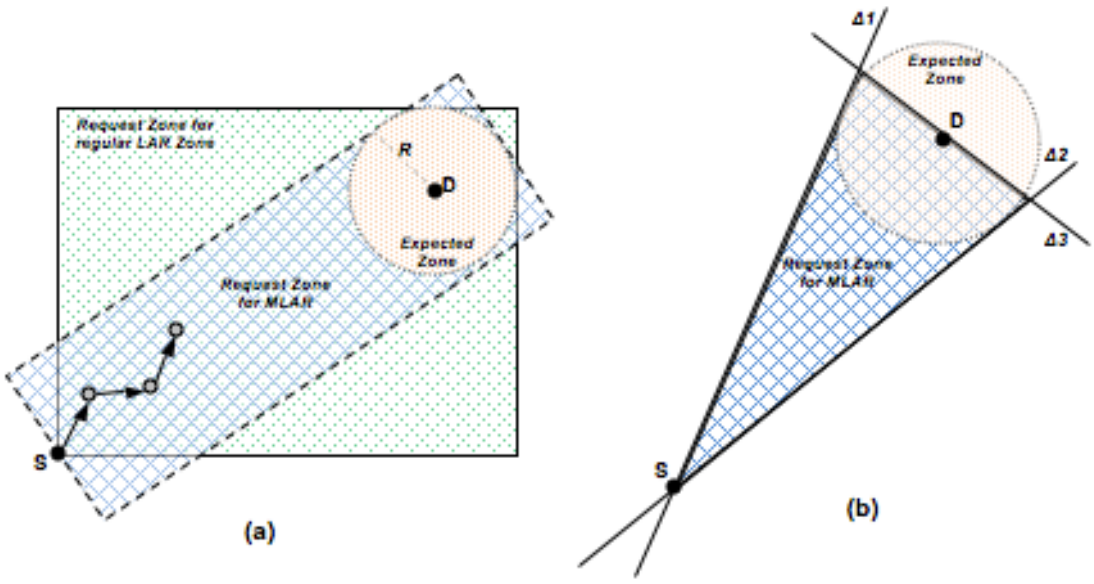

Figure 2.3: Alternative Definition of the Request Zone: (a) Rectangular shaped (b) Cone-shaped

MLAR also makes use of fixed nodes, or posts, which are nodes that move very little or not at all. If one of these nodes exists in the request zone, then the route discovery process is performed in two steps. First, find a route to the post. Then, the post will finish the route discovery process from there. The hope is that the post already has a route to the destination. In the event of failure, MLAR will enlarge the request zone instead of immediately reverting to flooding.

MLAR also examined an alternative cone-shaped definition of the request zone, shown in Figure 2.3 (b). The cone-shaped request zone is defined via three lines $\Delta 1$, $\Delta 2$, and $\Delta 3$. Lines $\Delta 1$ and $\Delta 2$ originate at the source and are tangent to the expected zone. Line $\Delta 3$ is perpendicular to the line between source and destination and travels through the destination node. In this variation of MLAR, only nodes that belong to the triangle defined by lines $\Delta 1$, $\Delta 2$, and $\Delta 3$ will participate in route discovery. The MLAR approach was shown to further reduce the overhead created in the network. Furthermore, the MLAR with a cone-shaped request area performed better than MLAR with a rectangular-shaped area. That is why the GeoAODV approach adopts the idea of a cone-shaped request zone area.

Geographical AODV (GeoAODV)

Geographical AODV (GeoAODV) is an AODV-based routing protocol that utilizes the knowledge of a node's location to reduce the route discovery overhead. GeoAODV employs an idea similar to that of MLAR with the cone-shaped request zone. However, unlike LAR protocols, GeoAODV relies on a distributed process to share location coordinates among the nodes in the network, instead of assuming that these coordinates are readily available. GeoAODV also limits the search area to a portion of the network that is likely to contain a route to the destination [4, 11]. By limiting the search area, GeoAODV decreases the amount of control traffic introduced into the network and thus improves the network's overall performance. GeoAODV is described in detail in subsequent chapters.

Prediction-Based Location Update

The Prediction-Based Location Update (PLU) algorithm [6] aims to improve upon current location update algorithms, which are used to distribute location data across a MANET and keep it up-to-date. Location update algorithms are also known as location update schemes or Location Information Services (LIS). The primary goal of any LIS is to reduce the number of Location Update Packets (LUP) sent into the network while still keeping a high level of accuracy with respect to location information. Therefore, these schemes are usually benchmarked on the accuracy of the location data distributed across the MANET and by the overhead introduced into the network. There are three main approaches for providing location information services: Location Information Flooding (LIF), DREAM Location Service, and Simple Location Service [6].

Table 2.1 A Sample Location Table

Node Address	Location Information (Cartesian coordinate)
150.250.191.218	(15.6, 10.0)
150.250.190.144	(30.0, 90.7)
150.250.64.69	(-8.0, 109.6)

In the LIF approach, all the nodes in the network keep a location table, which maps node addresses to location information. An exemplative mapping is shown in Table 2.1. This table is updated based on periodic broadcasts of location information by individual nodes.

This solution is simple and easily implemented. However, it relies on flooding, which consumes too many of the available resources in the network.

DREAM Location Service (DLS) is an extension built on top of the Distance Routing Effect Algorithm for Mobility (DREAM) [16]. DLS uses the distance between two nodes to determine how frequently the nodes should send LUPs to one another. DLS classifies all nodes in the network into one of two distinct groups: nearby or faraway. It updates the location information of the nearby nodes more frequently than that of the faraway nodes. Faraway nodes move slower in relation to the node sending the updates and thus do not require frequent updates.

Simple Location Service (SLS) performs the location information update identically to DLS, except for one variation [17]. Instead of a node sending only its own coordinates, each node broadcasts its entire location table to neighboring nodes. All downstream nodes merge the table received with their own table and also periodically share their location tables with their own neighboring nodes. By reducing the frequency with which LUPs are sent, DLS and SLS reduce the amount of overhead in the network.

The PLU algorithm relies on location prediction and one-hop broadcasting of location updates to decrease the control-packet overhead introduced by the routing protocol [6]. The algorithm uses a timer to trigger PLU to send updates using one of the

aforementioned LIF schemes. This algorithm is broken into two stages. In the first stage, the timer is calculated based on the range and velocity of the node. In the second stage, the timer is calculated based on how much the node is expected to move.

The goal of the first stage is to carry out normal location updates. These updates are generated on a periodic basis. The amount of time between updates is a function of the transmission range and average velocity of a node. The idea is that the nodes with a long transmission range or low velocity generate updates less frequently, while the nodes with a short transmission range or high velocity are updated more frequently [6].

$$Interval_1 = \frac{Range_{trans}}{\alpha \times v_{avg}} \tag{2.6}$$

where,

- $Interval_1$ is the interval of the first stage
- $Range_{trans}$ is the node's transmission range
- v_{avg} is the average speed of the node
- α is a scaling factor.

While a longer update interval introduces fewer packets into the network, it also may lead to a situation where substantial changes in a node's location are not propagated quickly enough. For this reason, PLU uses a second stage to distribute LUPs.

The goal of the second stage is to trigger an update whenever a node's location changes substantially. Using equations (2.7) and (2.8), PLU calculates a node's predicted location and compares it against the previously predicted values. If the difference between the newly and previously calculated values is outside of a predefined threshold, then the node will broadcast a LUP containing its current location. The study showed that PLU was reasonably accurate; it had the lowest average location error for speeds greater than 6 m/s as compared to DLS, SLS, or flooding. Also, when PLU is used, 95% of location data in the network is less than 10 meters off from the actual location of a node.

$$x_2 = x_1 + v \times (t_2 - t_2) \times cos\theta \tag{2.7}$$

$$y_2 = y_1 + v \times (t_2 - t_2) \times sin\theta \tag{2.8}$$

AODV-Directional Forward Routing

AODV Directional Forward Routing (AODV-DFR) is another routing algorithm that takes advantages of location information to reduce the control traffic overhead. AODV-

DFR is a hybrid routing algorithm that combines the ideas of proactive and reactive protocols with forward routing. Forward routing refers to the concept of storing the location of other nodes in the network in a node's routing table. Each node keeps a direction cache, in addition to a routing table. The direction cache stores the last known traveling direction of the nodes in the network, along with an expiry time. In the AODV-DFR approach, the nodes compute routes on-demand (i.e., reactively or as needed), while the routing updates are propagated in a proactive manner. This approach allows AODV-DFR to generate less control traffic overhead than conventional MANET routing protocols.

Similarly to AODV, AODV-DFR starts the route discovery process only when there is data to transmit to a destination for which a path is currently unknown. However, AODV-DFR nodes also periodically advertise their locations to their one-hop neighbors. During the route discovery process, AODV-DFR nodes maintain the reverse path to the source node by recording the information about the nodes from which the RREQ arrived (i.e., this node is the next hop on the path to the source). The recorded information also includes the traveling direction of the node, which is to be stored in the direction cache.

The evaluation study showed that, by only exploring promising paths to the destination, AODV-DFR drastically reduces the amount of overhead introduced into a network in an environment with mobile nodes. When the nodes in the network are mobile, AODV-DFR generates about half as much traffic as AODV.

Multicast and Geocast Routing in MANETs

Multicast routing has emerged as an efficient means of communicating between large domains in MANETs. This is important when a message needs to reach a broad audience. For example, a search and rescue team may want to alert all of the volunteers that a missing person was found and that the search is over. However, finding a route to every node in a multicast group tree can introduce a larger amount of overhead than finding a route to a single node. This will ultimately degrade the performance of the network. In an ad-hoc network, multicast trees can become very complex. This makes it difficult for the multicast network to scale appropriately [10]. Several approaches have been explored to improve the performance in multicast environments such as Geo-Assisted Multicast Inter-Domain Routing (GMIDR), which relies on geo-routing and clustering to facilitate communication between the network domains. GMIDR employs elected group cluster heads (GCH) to provide communication to the entire multicast group. A multicast tree is built from the source node to each GCH, instead of to individual multicast group members. A GCH is responsible for receiving information from the source and delivering it to all the members of the multicast group. This approach was shown to significantly reduce the overhead associated with the multicast group management.

Geocasting is a mechanism for sending messages to a specific geographical region, also known as a geocast region [4]. A simple method for achieving geocasting in MANET is to flood the geocast network region with data packets. The solution proposed in [4] relies

on the idea of LAR and employs geographical information to reduce the amount of overhead introduced into a network by the geocast flooding algorithms. Similar to LAR, this geocast solution defines a forwarding zone such that only the nodes within that area rebroadcast geocast packets to their neighbors. The geocast solution that employs a forwarding zone significantly reduces control traffic overhead and was able to provide accuracy of data delivery comparable to that of the regular geocast flooding.

Chapter 3

GeoAODV

Introduction to Geographical AODV

Geographical AODV (GeoAODV) [11, 12, 13, 14, 15] is an extension of the AODV protocol, which, like the LAR protocol, attempts to reduce the amount of control message overhead during the route discovery process. GeoAODV is also a reactive protocol, which initiates the route discovery process only when the node has data to be transmitted to a destination for which a path is currently unknown. However, unlike AODV, GeoAODV conducts the route discovery only within a limited area. GeoAODV defines its search area in the same way as does the MLAR cone-shaped approach shown in Figure 2.3 [3].

GeoAODV also defines the route discovery search area as a cone-shaped request zone, where the apex point is located in the position of the source node. However, GeoAODV determines the size of the cone-shaped area via a flooding angle. The source node dynamically controls the value of the flooding angle. MLAR assumes that the destination node coordinates and traveling speed, which are used to compute the expected zone and the cone-shaped request zone area, are readily available to everyone in the network. GeoAODV does not make this assumption. Instead, GeoAODV piggybacks location information onto control messages during the route discovery process. Effectively, GeoAODV dynamically distributes the node location information through the network during the route discovery process. Thus, GeoAODV initially performs the same way as

regular AODV. GeoAODV begins to search for a route to the destination in the cone-shaped request zone only after the destination node's location information has been distributed and the source node has GPS coordinates of the destination node.

Another key difference between the MLAR protocol and GeoAODV is the dynamic adaptability of the cone-shaped area. If the path is not completely located within the request zone (i.e., if there is some constituent hop that is outside the zone), then MLAR fails to find a route to the destination. The authors of LAR and MLAR do not specify how the protocol should handle such an event. GoeAODV, on the other hand, dynamically increases the cone-shaped request zone angle, known as the flooding angle, until the protocol either finds a route to the destination or searches the whole network and determines that there is no such route. Note that the GeoAODV protocol operates the same way as does regular AODV once the flooding angle reaches 360 degrees.

Overview of GeoAODV Operation

The shape of the GeoAODV request zone can also be described as an isosceles triangle, where the source node is the vertex of the triangle and is located in the top corner opposite to the base (i.e., the source node is the origin point of the equal sides of the triangle). The destination node is located on the line that originates at the source node and is perpendicular to the base of the triangle. The width of the GeoAODV request zone (i.e., the isosceles triangle) is controlled via the angle between the equal sides. This protocol configuration parameter is denoted as α. Only the nodes located within the confines of

the GeoAODV request zone participate in route discovery (i.e., only the nodes within the request zone rebroadcast arriving RREQ packets). All the other nodes discard arriving RREQs. For example, Figure 3.1 illustrates an instance of GeoAODV where a RREQ from source node S arrives at an intermediate node N. Node N tries to determine if it belongs to the GeoAODV request zone for node S. In order to make this determination, an intermediate node N computes the angle θ that is formed between the source node, itself, and the destination. Since the source-destination vector always divides the flooding angle evenly, node N belongs to the request zone if angle θ is not larger than one half of the flooding angle α.

$$\theta \leq {}^{1}/_{2} \times \alpha \tag{3.1}$$

Thus, if inequality (3.1) holds, then node N is located within the request zone for source node S and will rebroadcast the RREQ packet. Otherwise, N is outside of the request zone and the RREQ will be discarded. The value of angle θ is computed according to equation (3.2), where $\overrightarrow{SD}$ denotes a vector between source node S and destination node D, $\overrightarrow{SN}$ denotes the vector between source node S and node N, and $|SD|$ and $|SN|$ are the absolute values of vectors $\overrightarrow{SD}$ and $\overrightarrow{SN}$, respectively.

$$\theta = \cos^{-1}\left(\frac{\overrightarrow{SD} \cdot \overrightarrow{SN}}{|SD| \times |SN|}\right) \tag{3.2}$$

At the start of the route discovery process, GeoAODV sets the flooding angle α to some initial value. This initial value could be determined by the freshness of the destination's GPS coordinates (i.e., the value of α increases proportionally to t_d, the time passed since the last update of the destination's location information). Once t_d crosses a certain threshold, the location information is considered stale. As a result α is set to 360 degrees and GeoAODV performs the same way as does regular AODV. Alternatively, the initial value of the flooding angle could be a function of the expected zone radius defined in equation (2.1).

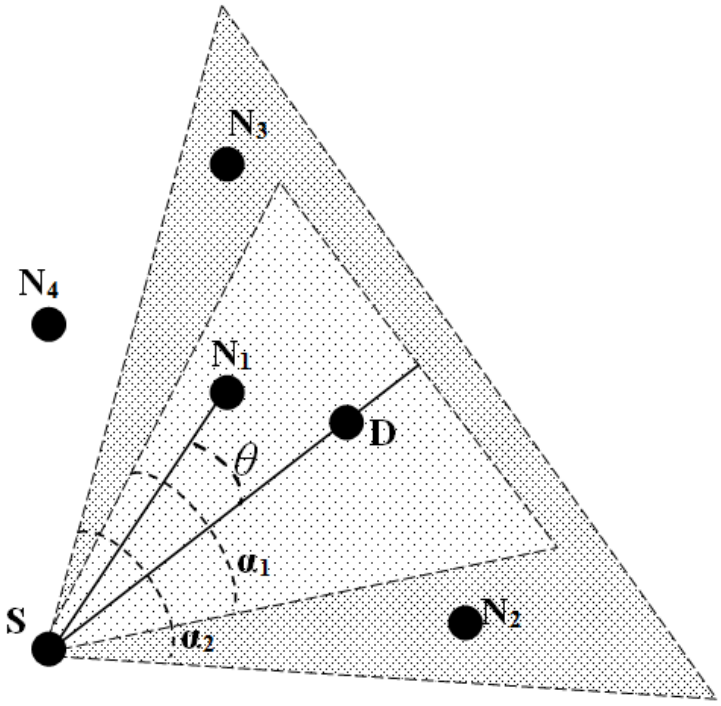

Figure 3.1 GeoAODV Protocol Operation

Figure 3.1 illustrates an example of the GeoAODV protocol operation where source node S initiates the route discovery process in an attempt to find a path to destination node D.

Initially S uses the flooding angle with the value α_1. The request zone defined by α_1 is shown in Figure 3.1 as an isosceles triangle of a lighter grey color. During this round of route discovery, only intermediate node N_1 rebroadcasts the RREQ packets. The remaining nodes are outside of the request zone defined by α_1 and do not participate in route discovery. These nodes (i.e., N2, N3, and N4) discard all arriving RREQs during this initial round. If the first round of route discovery fails, then the source node increases the flooding angle to some new value α_2 and repeats the process again. During the second round of route discovery, the request zone is extended (shown in Figure 3.1 as a darker color isosceles triangle) and intermediate nodes N1, N2, and N3 rebroadcast RREQs. Intermediate node N4 discards all arriving RREQ packets, since it is located outside of the request zone defined by the flooding angle α_2.

Distributing Location Information in GeoAODV

GeoAODV assumes that each node in the network knows its own position with precision through the use of a GPS-like device. However, GeoAODV nodes do not explicitly possess knowledge about the destination's coordinates or traveling speed, and therefore must learn about other nodes' positions during the lifetime of the network. The location information about other nodes in the network is obtained during the route discovery process. To store this information, each node in a GeoAODV network maintains a supplementary geographical location table, known as a GeoTable. An entry in the GeoTable consists of: the location information (e.g., GPS coordinates), the freshness timer, the AODV sequence number, the identity of the destination node (e.g., IP address), and status. The freshness timer keeps track of when the node coordinates were last

updated, while the AODV sequence number allows an intermediate node to identify if the arriving control packet (e.g., RREQ, RREP) carries new location information. This process is similar to the manner in which the AODV protocol differentiates between new and old control packets. However, the GeoTable entries remain valid for longer periods of time than do the entries in the AODV routing table. This is because the location information can help determine the general direction in which the destination node may be located, even if a route to that destination has changed. Table 3.1 illustrates a possible GeoTable stored in a GeoAODV node.

Table 3.1 An example of a GeoTable

IP Address	Coordinates	Timer	Sequence Number	Status
192.168.0.8	(104.7, -365.7)	10	1234567220	Fresh
192.168.0.77	(134.0, -59.1)	55	9446543201	Deleted
192.168.0.90	(4.0, 256.8)	38	9446543201	Fresh
192.168.0.234	(-47.2, 56)	13	7357907642	Stale

The formats of RREQ and RREP packets were modified to carry additional information (e.g., the locations of the source and destination nodes, the flooding angle). This information is used to populate the GeoTable, as well as to determine if an intermediate

node should participate in route discovery. At the start of the route discovery process, the source node consults its GeoTable and generates an RREQ packet that will carry: the node's own location information, the initial value of the flooding angle, and the last known location of the destination node. If the source node does not contain a GeoTable entry for the destination node, then the flooding angle is set to 360 degrees and GeoAODV shall operate in the same way as does regular AODV.

Upon the arrival of an RREQ message, all nodes (even those that will discard the RREQ packet) update their GeoTables with the source node's location information. An intermediate node only updates its GeoTable with the destination's location information if the destination sequence number carried in the RREQ is larger than that stored in the node's GeoTable. Otherwise, an intermediate node discards the destination coordinates carried in the RREQ packet. Similar processing occurs when an RREP is sent back. Each intermediate node updates its GeoTable with the source and destination location information carried in the packet. GeoAODV also utilizes periodic AODV Hello messages (which have the same header format as RREQ packets) to distribute location information among the neighboring nodes.

Stale GeoTable entries are identified using the sequence number and freshness timer. The nodes periodically purge stale entries from the table, similarly to the manner in which AODV updates its routing table. This is done using two timers. The first timer is known as the freshness timer. It is used to identify when an entry becomes stale. When an entry becomes stale, the node changes the entry's status to stale and starts the second

timer. Stale entries are eligible for deletion. When the second timer expires, the entry is marked for deletion and can be removed from the GeoTable. Each node periodically purges all the entries that are marked for deletion from the table.

Generally, the GeoTable retains entries longer than the AODV routing table. This is because the routes can become unavailable quickly in a highly dynamic MANET environment where the nodes are moving around. However, even incorrect geographical location information can still be helpful in limiting the route discovery search area by providing a general direction in which the destination node is likely to be located. Typically, this is sufficient to determine the request zone area where the route discovery process should be conducted.

GeoAODV

The GeoAODV protocol operates as follows. When the source node or an originator receives a request to transmit data, it checks its forwarding table. If a route to the destination is known, then the node transmits the data. If a route to destination is unknown, then the node queues the data and initiates the route discovery process.

The first step of the route discovery process is to determine the value of the flooding angle. If the GeoTable contains the location information for the destination node, then the flooding angle is computed based on the freshness of the location coordinates, otherwise, the flooding angle is set to its maximum value of 360 degrees. Next, the originator node creates an RREQ packet and broadcasts it in the network. The generated RREQ carries the following values: the computed flooding angle value, the originator's location

coordinates, the originator sequence number, the last known destination coordinates (if available), and the destination sequence number (if available).

Upon RREQ arrival at an intermediate node, GeoAODV performs the following steps before the packet is rebroadcast. First, the node updates its GeoTable with the most up-to-date source and destination information. Next, the node performs the regular AODV validation to determine if the arriving RREQ is a duplicate. If so, then the packet is discarded. Otherwise, the node checks its routing table to see if it knows a route to the destination. If such a route is available, then the intermediate node discards the RREQ packet and sends an RREP message back to the source node. Note that the RREP message will contain fresher destination location and sequence number information, as well as the intermediate node's location coordinates and sequence number. This feature facilitates distribution of the node location information throughout the network.

If the RREQ packet was not discarded and if no RREP was generated, then the intermediate node performs the GeoAODV validation to determine if it is within the originator's request zone. If the node is outside the request zone, then the packet is discarded. Otherwise, the node rebroadcasts the RREQ packet. Note that if the node's GeoTable contains destination information that is fresher than that which is carried in the RREQ packet, then the node may update the RREQ's destination location and sequence number fields.

When the destination node receives an RREQ packet, it updates its GeoTable and sends an RREP message back to the originator. Note that while the RREP message is unicast back the originator node, each packet transmission in the MANET environment is overheard by all the neighboring nodes. Thus, all the neighboring nodes along the path between the destination and the originator will overhear the RREP message and will each update their GeoTable with the most up-to-date location information for the destination and originator nodes. Once the RREP packet arrives at the originator, the route discovery process is complete and the originator begins sending data to the destination node.

If the originator does not receive an RREP within a certain amount of time during the first round of route discovery, then it is assumed that a route to the destination within the current request zone cannot be found. In this case, the originator node increases the value of the flooding angle and repeats the route discovery process, this time searching a wider area. Eventually, the GeoAODV request zone could be morphed into a network-wide search, as in regular AODV. Figure 3.2 illustrates how the GeoAODV request zone is expanded during each round of route discovery until it becomes the regular AODV request zone.

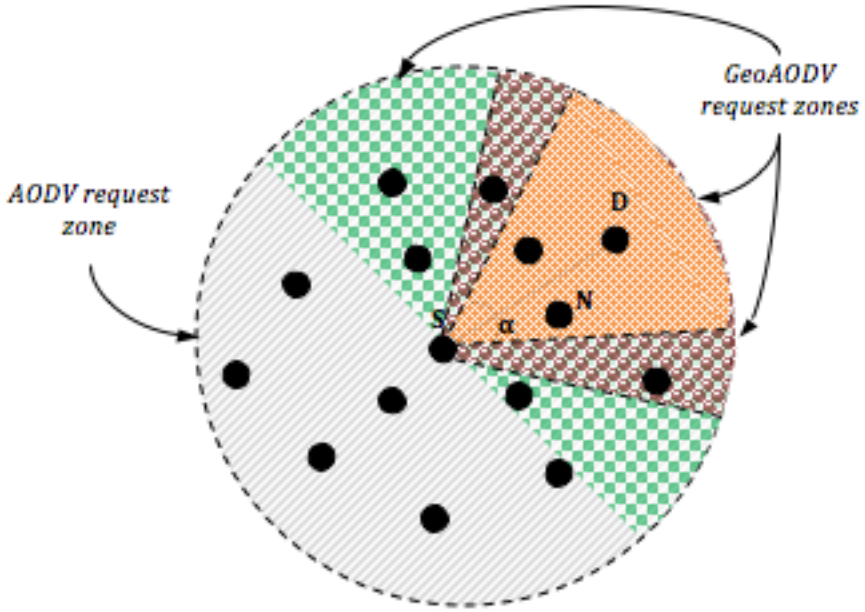

Figure 3.2 GeoAODV Request Zone

GeoAODV Rotate Protocol

This study considered two variations of the GeoAODV protocol: *GeoAODV Static* and *GeoAODV Rotate*. GeoAODV Static operates as described above. The GeoAODV Static request zone remains unchanged during each round of route discovery (i.e., the source node is always the vertex opposite to the base of the isosceles triangle). GeoAODV Rotate operates slightly differently. It re-orients the request zone towards the destination at each intermediate node by making the previous node a new vertex of the triangle (i.e., each intermediate node re-computes the request zone based on the location of the previous hop, rather than on the location of the source node).

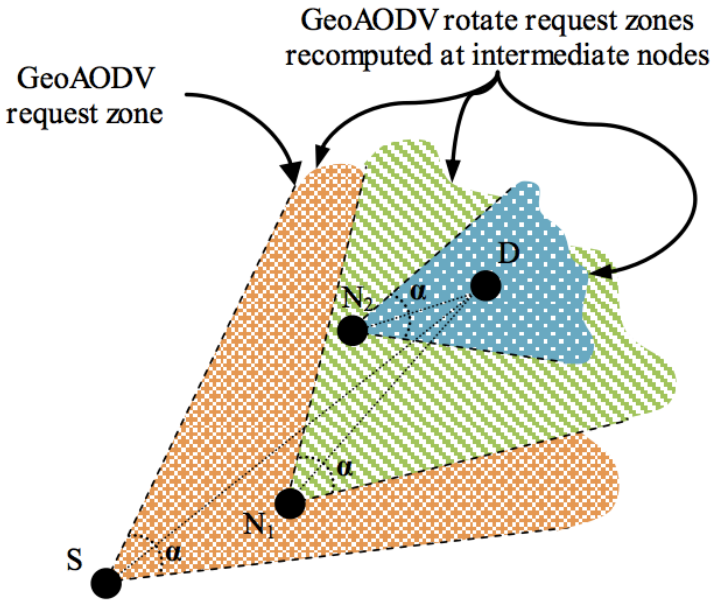

Figure 3.3 GeoAODV Rotate Protocol

Figure 3.3 illustrates the idea of GeoAODV Rotate. Node $\mathbf{N_1}$ belongs to the request zone that was computed based on the location of node **S.** Node $\mathbf{N_2}$ belongs to the new, re-oriented request zone that was computed based on the location of node $\mathbf{N_1}$. Although they belong to different request zones, both nodes $\mathbf{N_1}$ and $\mathbf{N_2}$ participate in route discovery. On the other hand, node $\mathbf{N_3}$, which receives an RREQ from $\mathbf{N_1}$, will not participate in route discovery. This is because node $\mathbf{N_3}$ is located outside the new request zone that was computed using the location of $\mathbf{N_1}$, its previous hop. Were GeoAODV Static to be used instead, all the nodes in the Figure 3.3 would participate in the route discovery process. This is because they all belong to the request zone that was computed based on the location of source node **S**.

Chapter 4

Implementation

Overview of Implementation

A network simulation was chosen as the means by which to compare the performance of GeoAODV Rotate against AODV, GeoAODV, LAR Zone, and LAR Distance. A simulation-based approach offers an effective way to accurately and inexpensively measure and analyze large-scale systems [18]. Implementing a network simulation using standard models is straightforward. However, modifying existing models can be a tedious and often time-consuming process. OPNET Modeler [19] provides a highly cohesive and flexible platform to modify existing networking protocols and is one of the most popular commercial network simulations in the world [18]. OPNET Modeler version 16.0 was used to implement GeoAODV and LAR routing protocols. OPNET's existing AODV process model was modified, along with several other process models and external files that are responsible for modeling MANET routing protocols. These implementations were verified and benchmarked. Then, the performances of the implemented protocols were compared.

AODV Implementation

In OPNET Modeler, a process model is used to represent a singular networking process such as a routing protocol, a load-balancing mechanism, or an upper-layer protocol [18]. Process models are implemented using the Proto-C programming language, which uses a mixture of state transition diagrams and the C programming language. Proto-C also

provides an expansive set of APIs to help model various systems. In particular, the OPNET Modeler implements AODV MANET routing protocol through the ***aodv_rte*** process model [20, 21, 22].

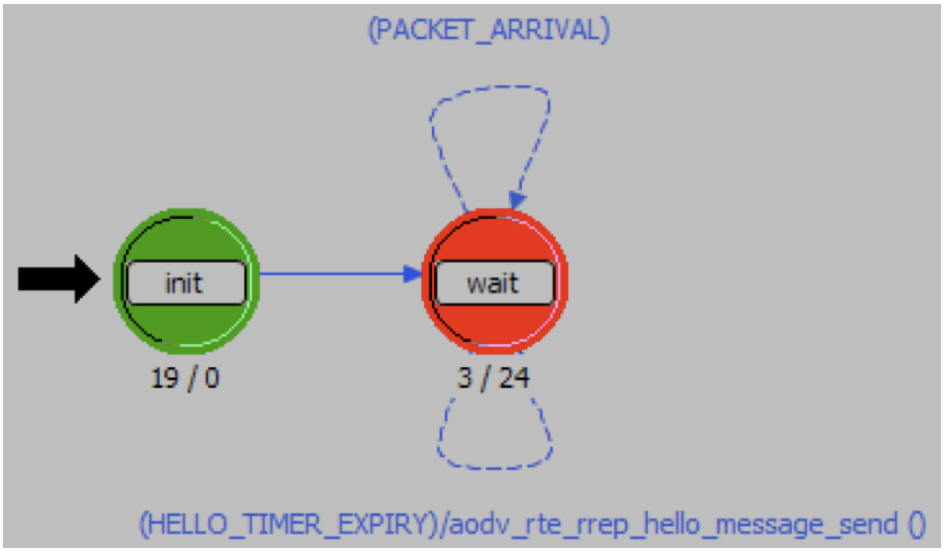

Figure 4.1 OPNET's aodv_rte Process Model State Diagram

Figure 4.1 provides a screenshot of the OPNET process model state diagram for the AODV MANET routing protocol. The AODV process model is made up of two states: ***init*** and ***wait***. The init state is denoted in Figure 4.1 by a green circle with the incoming arrow to the left of it. It is responsible for initializing any data structures and related processes required for AODV's operation. This includes the AODV packet queue, AODV request table, AODV route table, statistic collectors, etc. While in this state, the node parses the routing protocol configuration and is configured accordingly. Note that each MANET node that employs AODV during the simulation will have an instance of the AODV process model associated with it.

While in the init state, AODV also sets-up all user-specified configuration parameters for the routing protocol. During the simulation set-up, the OPNET user may configure each protocol by providing the values for protocol configuration parameters, which are also known as model attributes. For example, the AODV process model includes model attributes such as *Active Route Timeout*, *Hello Interval*, *Net Diameter*, and several others. To properly model GeoAODV and LAR protocol operation, several new model attributes were added to the AODV process model. The following model attributes were added:

- *Geo-Assisted Protocol Type* – specifies which AODV-based protocol will be used in the simulation: AODV, GeoAODV, GeoAODV Rotate, LAR Distance, or LAR Zone
- *GeoAODV Initial Flooding Angle* – specifies the value of the flooding angle used by a MANET node when starting the GeoAODV route discovery
- *GeoAODV Flooding Angle Increase* – specifies the amount by which the value of the flooding angle will be increased after each round route discovery failure.

Several other model attributes were added as well.

OPNET performs the actual parsing of the model attributes for all MANET routing protocols in a single process model called *manet_mgr*, which subsequently sends parsed attributes to the corresponding MANET routing protocol process model. The *aodv_rte* process model initializes corresponding internal data structures to user-specified configuration values in its *attributes_parse_buffers_create* function. The complete code for GeoAODV and LAR's init function can be found in Appendix A.

Once the protocol initialization is complete, the AODV process model moves into the *wait* state, which implements the actual operation of the AODV protocol. In this state, the AODV protocol waits for either a packet arrival or the Hello timer to expire. Upon packet arrival, AODV discovers the packet type (i.e., data packet or ADOV control packet), performs the corresponding packet processing (e.g., forward the data packet into the network if a route to destination is known or queue the data packet otherwise, perform AODV protocol operations upon the control packet arrival), and returns to the *wait* state. If the Hello timer expires, then the AODV process will broadcast a HELLO message to all of its immediate neighbors and return to the *wait* state.

To implement the LAR Distance, LAR Zone, GeoAODV, and GeoAODV Rotate protocols, we modified the *aodv_rte* process model by dividing the protocol processing into six logical modules [19]:

- *aodv_packet_queue* – module for managing the incoming and outgoing packet data
- *aodv_pkt_support* – module for creating AODV control packets and headers
- *aodv_request_table* – module that manages AODV's request table
- *aodv_route_table* – module that manages AODV's routing table
- *aodv_support* – module for various supporting functions, including collecting statistics and printing debugging information
- *manet_support* – module that exposes MANET functionality to other MANET protocols and neighboring layers.

Upon packet arrival, the AODV process model calls the *aodv_rte_pkt_arrival_handle* function. If the arriving packet is an application data packet, then AODV calls the *app_pkt_arrival_handle* function. If it is a control packet, then it is handled by the control message handler. Specifically, the AODV process model contains the following functions for processing incoming AODV control packets:

- *rreq_pkt_arrival_handle* – handles the arrival of the route request packets
- *rrep_pkt_arrival_handle* – handles the arrival of the route reply packets

Additionally, the aodv_rte process model includes the *route_request_send* and *route_reply_send* functions. These are responsible for generating and forwarding RREQ and RREP messages, respectively. The *rreq_pkt_arrival_handle* function was modified to disregard packets that should not be rebroadcasted.

Location Information Sharing

GeoAODV distributes location information in the network through the use of AODV control messages. For that purpose, internal OPNET data structures that represent control information carried by the RREQ and RREP packets were modified. Specifically, the *AodvT_Rreq* and *AodvT_Rrep* data structures were modified to include the following information: the value of the flooding angle (i.e., the request level), the GPS coordinates of the originator, and the GPS coordinates of the destination node. This code can be seen in Appendix B. The new structure of an RREQ packet is shown in Figure 4.2. To properly create and process the new RREP and RREQ packet formats, two new functions

were added in the *aodv_pkt_support.ex.c* OPNET external support file. These functions are known as *aodv_pkt_support_rreq_option_create_geo* and *aodv_pkt_support_rrep_option_create_geo*. Note that the AODV HELLO messages use the same header fields as the RREP messages and thus also carry location information. A HELLO message is simply an RREP packet with its IP TTL header field set to 1.

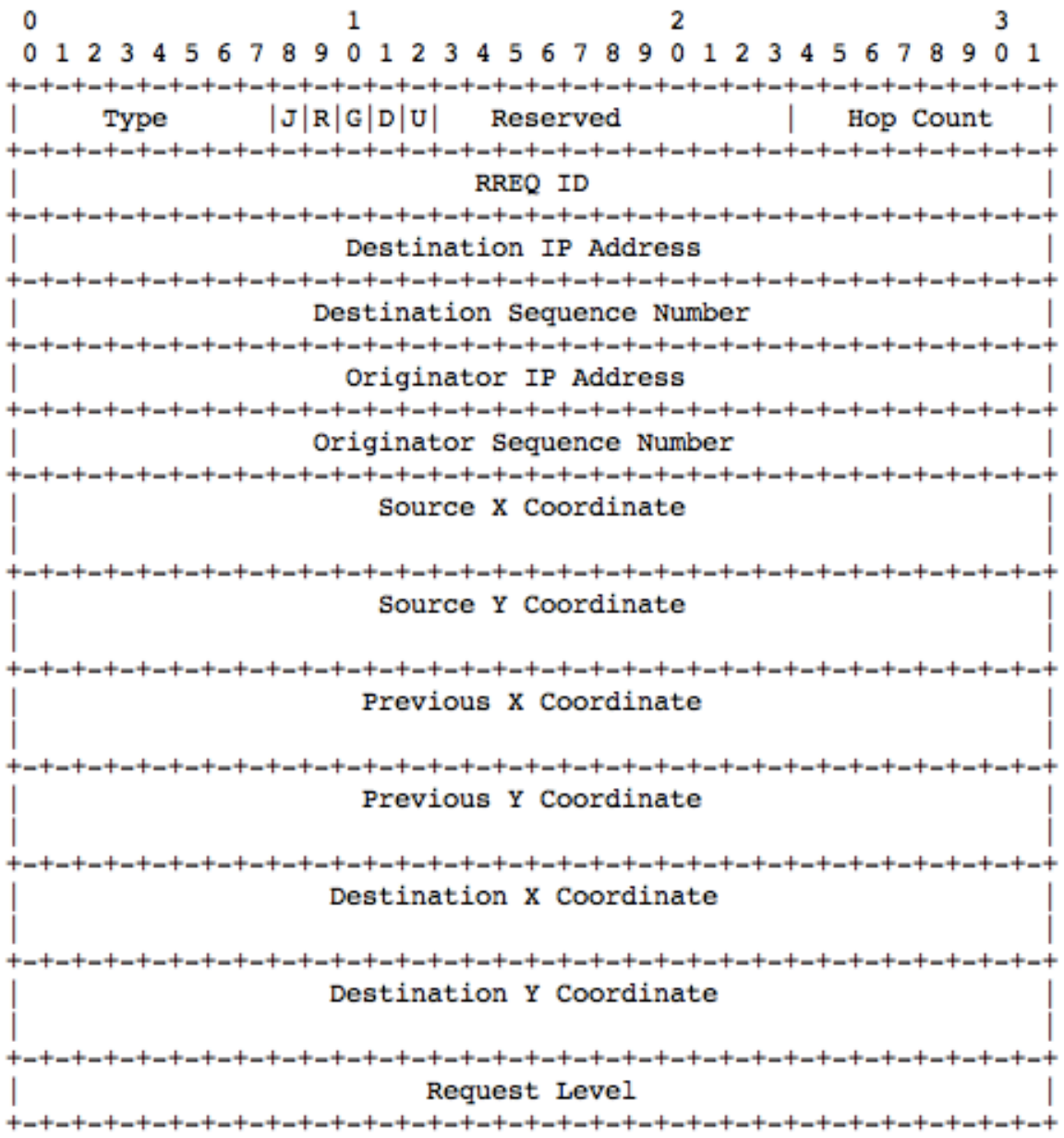

Figure 4.2 The RREQ Packet Structure used in the Simulation

LAR assumes that all nodes in the network have access to location information and to the traveling speed of all other nodes in the network. OPNET offers a data definition sub-package, known as *oms_data_def*, which was used to model this behavior. This sub-package can be used to manage a global database that is shared by all nodes in the simulation. To implement the ubiquitous availability of LAR's geographical information,

the AODV process model was modified to have each node record its coordinates and traveling speed in the global database during the process initialization. Similarly, during the simulation run, individual nodes periodically update this database with new values for their coordinates and traveling speed. The periodicity of this update is a user-configurable parameter and can be specified during simulation set-up.

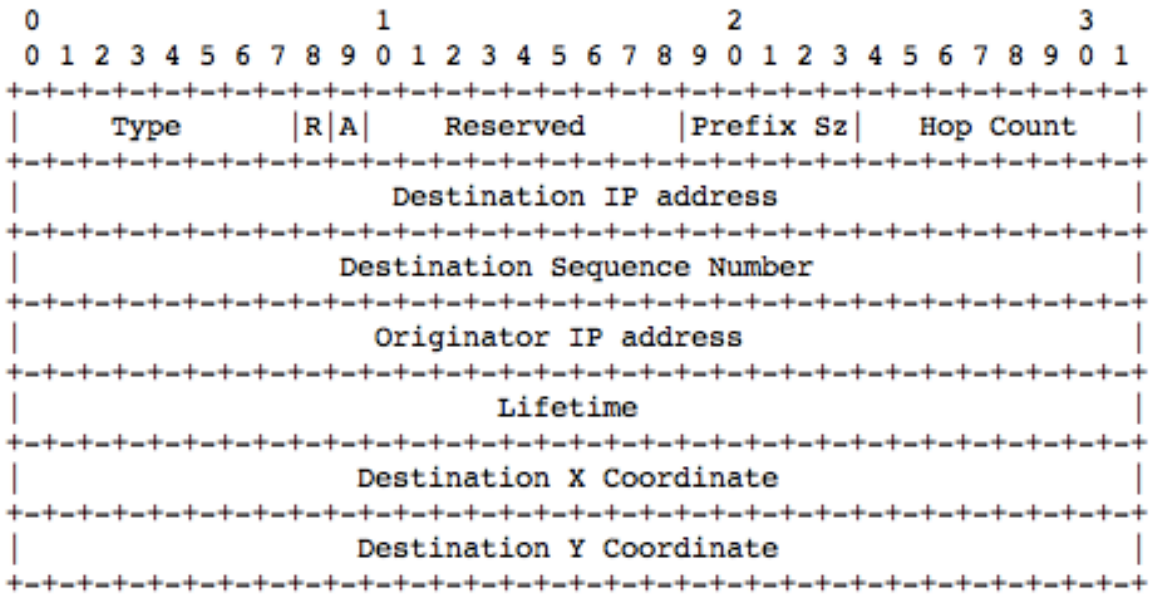

Figure 4.3 The RREP Packet structure used in the Simulation

OPNET automatically creates a global attribute database. The AODV process model can simply use this database as needed. Specifically, the *oms_data_def_entry_insert* function was used to add an entry into a network-wide database and the *oms_data_def_entry_access* function was used to retrieve specific information as needed. The entries in the global attribute database are keyed based on the node's IP address.

Unlike LAR, GeoAODV nodes maintain their own local database to store discovered location information. This database is called the GeoTable and it is populated during the route discovery phase and through periodic HELLO messages. The GeoTable is implemented as a hash map that stores node locations and is indexed by the node's IP address. This table is updated through the additional information (i.e., coordinates of the originator and destination nodes) carried in the modified RREQ and RREP packets. The GeoTable data structure is modeled after an AODV routing table and is defined in the *aodv.ex.h* header file. This file contains definitions of various data structures, including the AODV routing table. The AODV routing table is defined as a C structure called *AodvT_Route_Table* with the entries defined as *AodvT_Route_Entry*. The AODV routing table keeps track of all valid routes to potential destination nodes in the network. Each time a new route is discovered, AODV updates its routing table and the common IP forwarding table. This functionality was modified to additionally insert destination coordinates into the GeoTable upon route discovery. OPNET generally implements packet forwarding within the IP module, which relies on the common IP forwarding table to determine the next hop address. The common IP forwarding table is updated by active, possibly different routing protocols used in the simulation study.

AODV also maintains a separate hash map indexed by the destination node's IP address, which keeps track of all requests that are originated-by and forwarded-by this node. The AODV request table is implemented in the C structure called *AodvT_Request_Table*. This table keeps track of all requests that were initiated by a node. It also allows the node to determine if it should generate a new request when an application data packet arrives and

there is no known route to the destination. If a request for the packet's destination was already generated, then the packet is simply queued; otherwise, AODV generates a new request. Each entry in the AODV request table consists of such information as request id, insertion time, current TTL value, number of retries, etc. These table entries are implemented as the *AodvT_Orig_Request_Entry* C structure. This data structure was modified to also keep track of the flooding angle value used during the most recent round of the route discovery process. If the current route discovery round fails to find a route to the destination, then the flooding value stored in the AODV request table is increased and the node attempts another round of route discovery.

The AODV request table is used in a similar fashion to implement the AODV expanding ring technique, where the node increases the TTL value in the generated route request packet after each failed attempt to discover a route to the destination [18]. The AODV request table is also used to identify duplicate RREQs that arrive at the node. The information about each RREQ that arrived at the node is stored in the AODV Route Request Table. Each subsequent request with the same id is simply discarded. Note that in broadcast environments such as a MANET, it is very likely that after forwarding (i.e., broadcasting) an RREQ packet, the node will receive duplicates of the same packet as they are forwarded (i.e., rebroadcasted) by the node's neighbors.

Lastly, AODV stores a hash map that keeps track all of the neighboring nodes located one-hop away. This table is called the connectivity table and is implemented as a C structure named *AodvT_Conn_Info*. The connectivity table is populated using periodic

HELLO messages. It is indexed using the node's IP address and contains the time at which the most recent HELLO message from that node was received. This table was not modified during implementation.

Geo-Assisted Routing Implementation

Creating and maintaining multiple supporting data structures described above was necessary in order to implement Location-Aided routing protocols. The logic for dealing with RREQ and RREP packet arrivals also required modification. Specifically, the following function responsible for handling control packet arrivals were updated in the *aodv_rte* process model: *aodv_rte_rreq_pkt_arrival_handle* and *aodv_rte_rrep_pkt_arrival_handle*.

When an RREQ packet arrives, AODV checks if the packet should be rebroadcasted. The internals of the *aodv_geo_rebroadcast* function use the information retrieved from the message to make the appropriate decision about rebroadcasting the packet. This code can be found in Appendix C. A different mechanism is employed depending on the types of the routing protocols used in the simulation:

- *AODV* – the node conducts regular AODV processing and rebroadcasts the packet only if the packet is not a duplicate and the node does not know the route to the destination.
- *LAR Zone* – in addition to regular AODV processing, the node verifies that it is within the LAR request zone, before rebroadcasting the packet. This is accomplished in several steps. First, the location coordinates for the originator

and destination nodes, as well as the traveling speed of destination node, are retrieved from the global table. Next, the request zone is computed. Finally, a determination is made as to whether the current node is located within the computed request zone. This is all implemented in the function called *aodv_geo_LAR_within_request_zone*.

- *LAR Distance* – in addition to regular AODV processing, the node verifies that it is located closer to the destination than was the previous node, before rebroadcasting the packet. This is accomplished by calculating the distance from previous node to the destination node and the distance from the current node to the destination node and finally comparing these values. This logic is implemented in the function called *aodv_geo_LAR_distance.*
- *GeoAODV* – in addition to regular AODV processing, the node verifies that it belongs to the GeoAODV request zone, before rebroadcasting the packet. This is accomplished in several steps. First, the location coordinates of the originator and destination nodes are retrieved, as well as the flooding angle from the arriving RREQ packet. Next, the GeoAODV request zone is computed. Finally, a determination is made as to whether the current node is located within the request zone. This is all implemented in the function called *aodv_geo_compute_angle.*
- *GeoAODV Rotate* – in addition to regular AODV processing, the node verifies that it belongs to the GeoAODV Rotate request zone, before rebroadcasting the packet. This is accomplished in several steps. First, the location coordinates for the previous and destination nodes are retrieved, as well as the flooding angle from the arriving RREQ packet. Next,the GeoAODV request zone is computed.

Finally, a determination is made as to whether the current node is located within the request zone. This is all implemented in the function called *aodv_geo_compute_angle.*

Chapter 5

Simulation Study

In order to evaluate and compare the performance of GeoAODV with that of other MANET routing protocols, a series of simulation studies were executed. Each series was configured with a different set of model attribute values and simulated different network conditions. Specifically, in this study the performance of AODV, GeoAODV Static, GeoAODV Rotate, LAR Zone, and LAR Distance in the MANET network were studied with the number of communicating nodes and traveling velocities being varied.

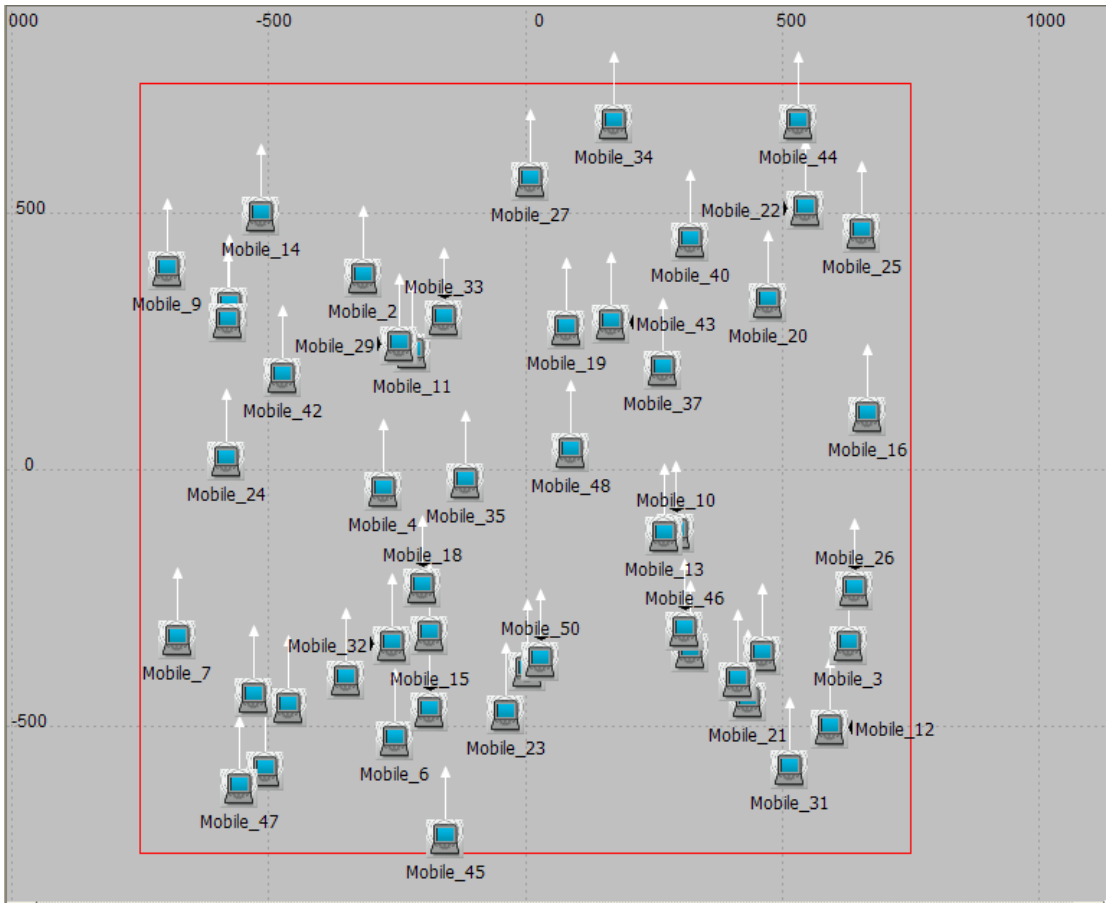

Figure 5.1 Network Topology used in the Simulation Study

Simulation Setup

The network topology used in this simulation study is depicted in Figure 5.1. It consisted of fifty WLAN nodes randomly placed within an area of 1,500 meters by 1,500 meters. The study was parameterized along two sets of model attributes: the number of communicating nodes and traveling velocity. Specifically, the number of communicating nodes was varied between 2, 5, 10, 20, and 30, while the node traveling velocity was set to: 0 meters/second (stationary nodes), 5 meters/second, 10 meters/second, and a random value which was obtained using a uniform distribution in the interval [0, 20]. Thus, 20 different network settings for each of five routing protocols were examined. This created a total of 100 unique scenarios. Furthermore, each of the unique scenarios was executed six times. The results for that scenario were averaged in order to acquire a single, more accurate figure. The upshot was a total of 600 simulation runs, which took close to 120 hours to complete on a Windows XP machine with 2.4 GHz dual-core CPU and 3 GB of RAM.

When configuring communicating nodes, each source-destination pair was randomly selected. Every communicating node was configured to wait 100 seconds before transmitting any data, in order to allow other network protocols to initialize and distribute any necessary protocol information throughout the network. The nodes were configured to move according to the Random Waypoint model. In this model, a node starts by pausing for a random amount of time. It then selects a random direction within the network domain and moves in that direction with a specified velocity. The node continues to repeat this process. In this simulation study, the pause time was determined

using an exponential distribution with a mean outcome of ten seconds. Once the node reaches the boundary of the network domain, it will redirect itself and continue traveling within the confines of the specified 1500 meters by 1500 meters network.

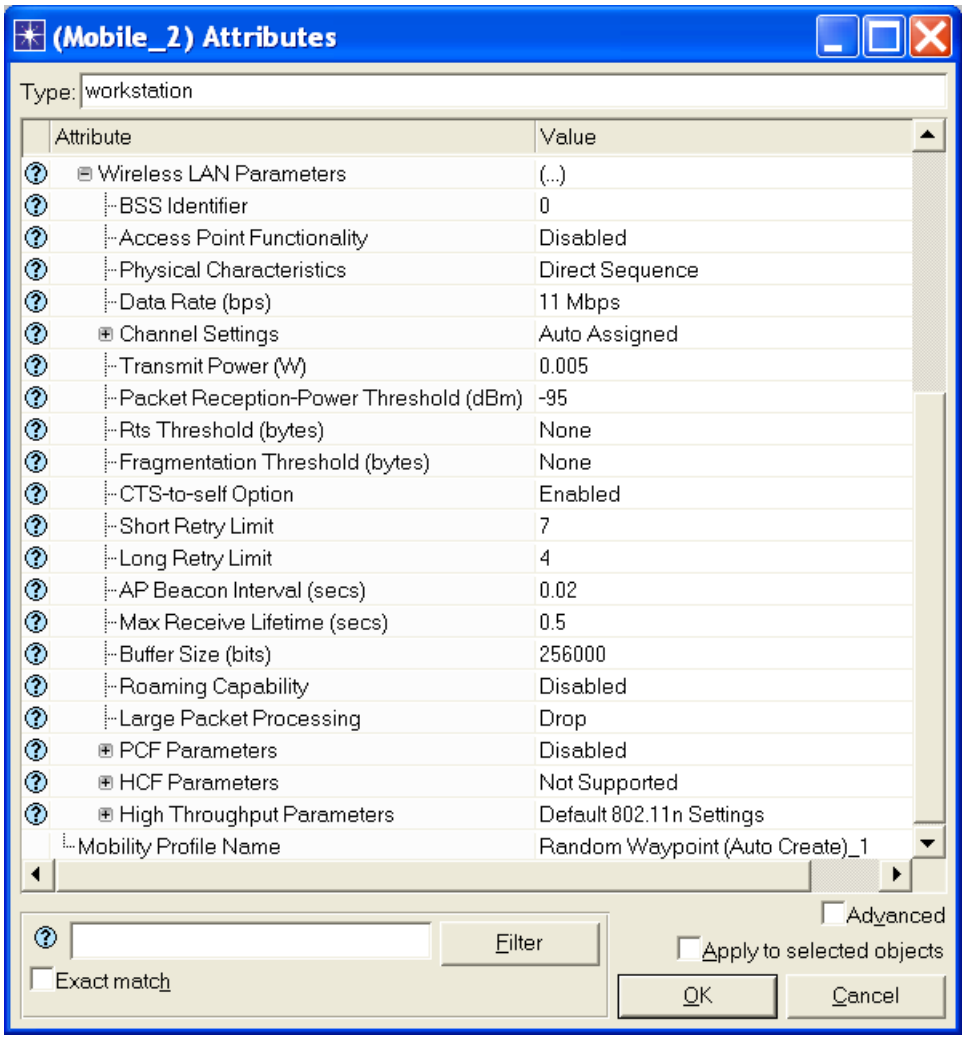

Figure 5.2 Summary of WLAN configuration

Each node in the simulation study was represented with the ***manet_station*** node model, which is typically used to simulate Wireless LAN (WLAN) nodes. The default values for

all of the WLAN configuration parameters of the nodes were used in this simulation study. A summary of WLAN configuration settings is provided in Figure 5.2.

Table 5.1 Summary of Node Configuration

Configuration Parameter	Value
Channel Data Rate	11 Mbps
Transmit Power	0.0005 Watts
Packet Reception Power Threshold	-95 dBm
Start of Data Transmission	Normal (100, 5) seconds
End of Data Transmission	End of simulation
Duration of Simulation	300 seconds
Packet Inter-Arrival Time	Exponential (1) second
Packet Size	Exponential (1024) bytes
Mobility Model	Random Waypoint
Pause Time	exponential(10)
Destination	Random

Communicating nodes were configured to start transmission after roughly 100 seconds had elapsed within the simulation. The actual time was computed using a normal distribution with a mean outcome of 100 seconds and a 5 second variance. Nodes continue sending data until the end of simulation. The source nodes generated a packet of approximately 1,024 bytes every second. The actual packet size and packet inter-arrival times were computed using an exponential distribution. A summary of key MANET configuration parameters for this simulation study is presented in Table 5.1.

Finally, both versions of the GeoAODV protocol were configured such that the initial flooding angle value was set to 90 degrees. The flooding angle value was incremented by 90 degrees after each unsuccessful route discovery attempt, until GeoAODV ultimately reverts to regular AODV (i.e., 360 degrees). If AODV fails to find a route, then GeoAODV fails the route discovery process. The LAR protocols were configured to have the MANET nodes publish their location and traveling velocities into a global database once every second. The configuration parameters for the LAR distance protocol, α and β, were set to 1 and 0, respectively. In this study, the default AODV configuration settings were used. A summary of these configuration settings is presented in Figure 5.3.

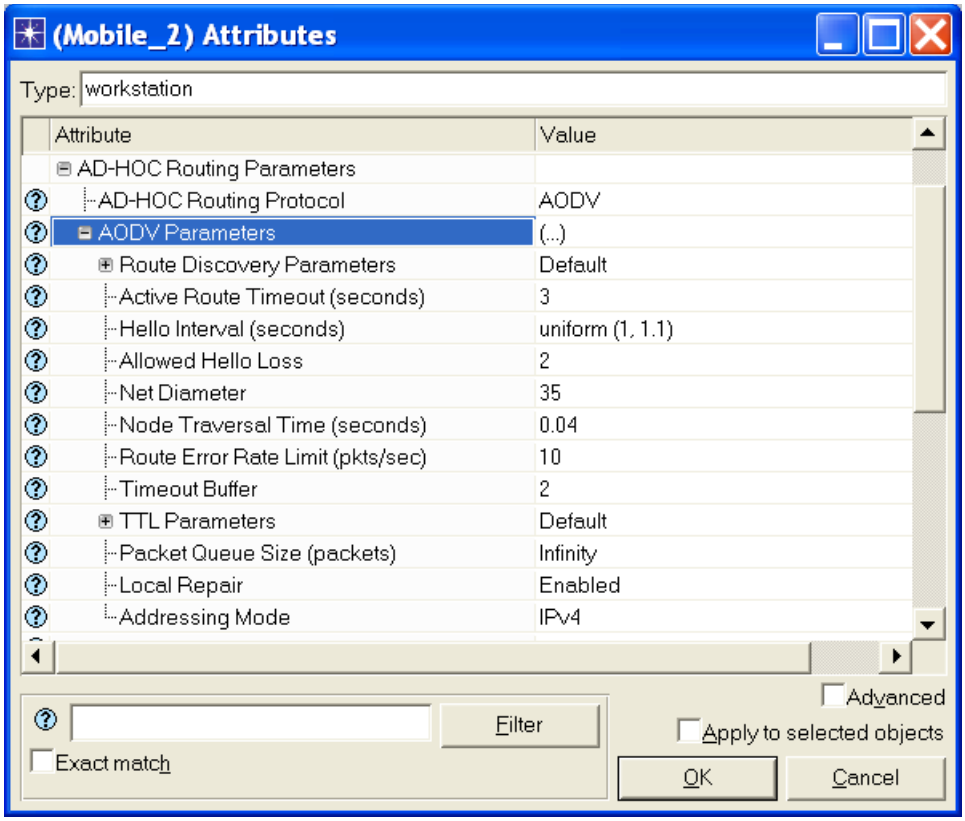

Figure 5.3 Summary of AODV Node Configuration

Analysis of the Results

A few simplifying assumptions were made in this simulation. For example, the end-to-end delay associated with the retrieval of GPS coordinates was not accounted for. In this evaluation of the GeoAODV protocols, the overhead introduced by the addition of new fields in the RREQ and RREP packet headers were similarly disregarded. With respect to the LAR protocols, the assumption was made that location information and traveling velocities are available everywhere in the network at no additional cost. This study primarily focused on the total amount of control traffic generated by each of the examined protocols. The raw results can be found in Appendix D. The results of this study suggest that all location-aided routing protocols outperform AODV by generating significantly fewer control packets during route discovery. A summary of the collected results is presented in Figures 5.4 – 5.8.

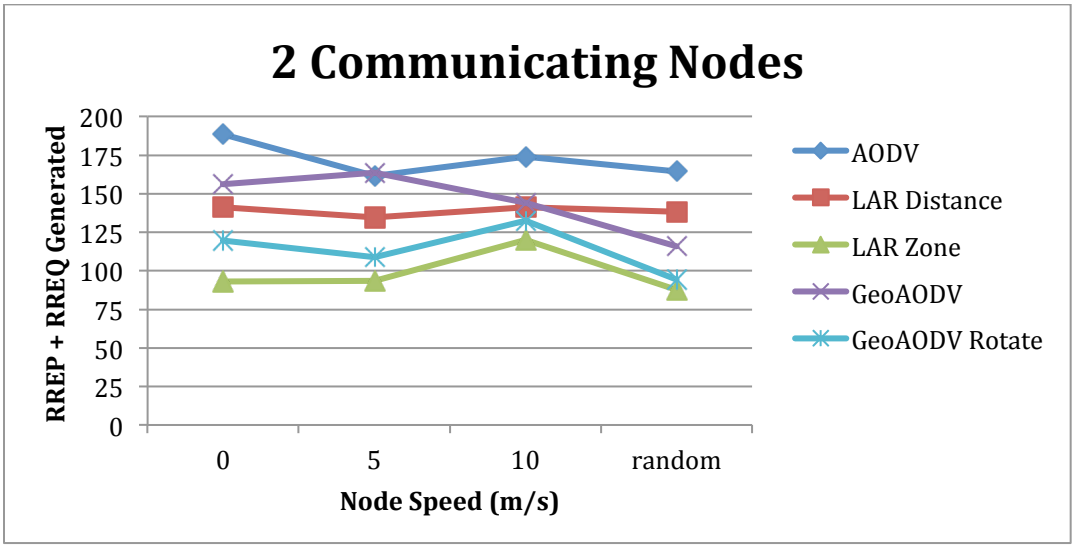

Figure 5.4 Number of Control Packets in Scenarios with 2 Communicating Nodes

Specifically, the results show that the LAR Zone protocol consistently generates the smallest number of control packets among all of the studied protocols, while GeoAODV Rotate is a close second. This can be attributed to the fact that the simulation does not account for the cost associated with retrieval of node coordinates and traveling speeds in LAR Zone (i.e., these values are assumed to be available as needed). On the other hand, GeoAODV makes no such assumption and dynamically distributes location information during the route discovery process.

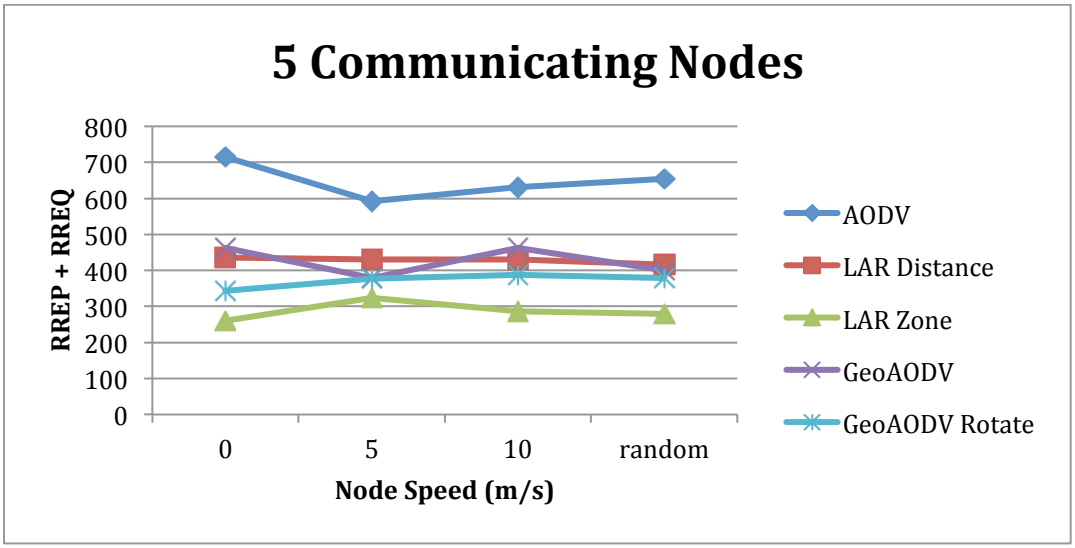

Figure 5.5 Number of Control Packets in Scenarios with 5 Communicating Nodes

Additionally, the nodes require some time to gather location information about the location of other nodes in the network before location-aided improvements of GeoAODV can begin being utilized. As a result, GeoAODV initially operates the same way as does regular AODV. GeoAODV can only take advantage of the limited flooding optimization after location information has been distributed in the network. In addition, it may take up to 3 rounds of route discovery using different values of the flooding angle before

GeoAODV realizes that limited broadcast optimizations do not help. In this case, route discovery is conducted using the AODV protocol. The LAR protocols, on the other hand, revert to AODV after a single failure.

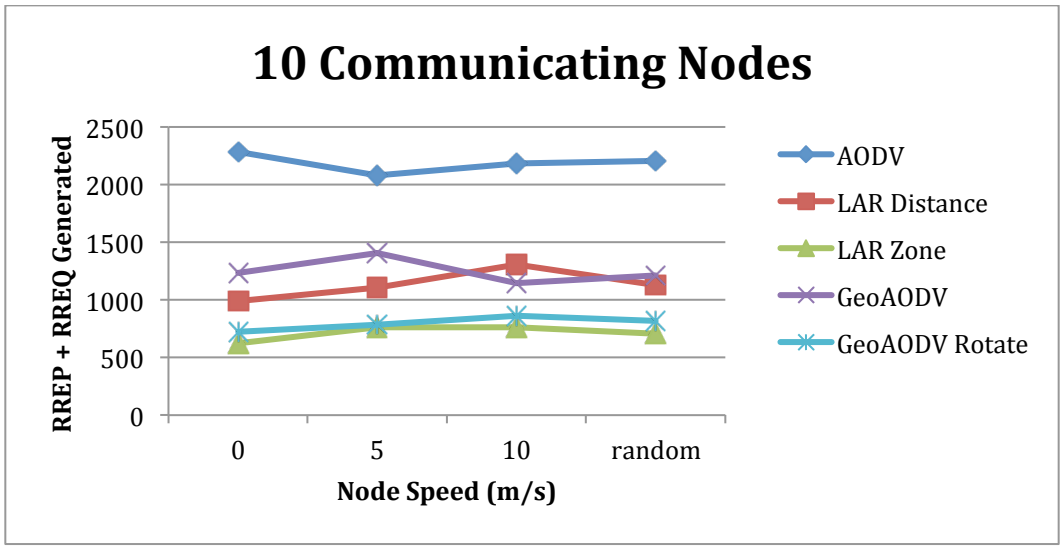

Figure 5.6 Number of Control Packets in Scenarios with 10 Communicating Nodes

The difference in performance between two LAR protocols can be attributed to how often the protocols do not find a route to destination and thus are forced to conduct route discovery using AODV. The search area of the LAR Distance protocol is very limited. The current node rebroadcasts the RREQ message only if it is closer to the destination than was the previous node. Thus, it is not surprising that LAR Distance fails to find a route to the destination more frequently than does LAR Zone. As a result, the LAR Distance protocol often behaves like AODV and generates a large number of control packets. LAR Zone, on the other hand, conducts route discovery over a wide area and thus is less likely to revert to AODV. This results in LAR Zone consistently outperforming the LAR Distance protocol.

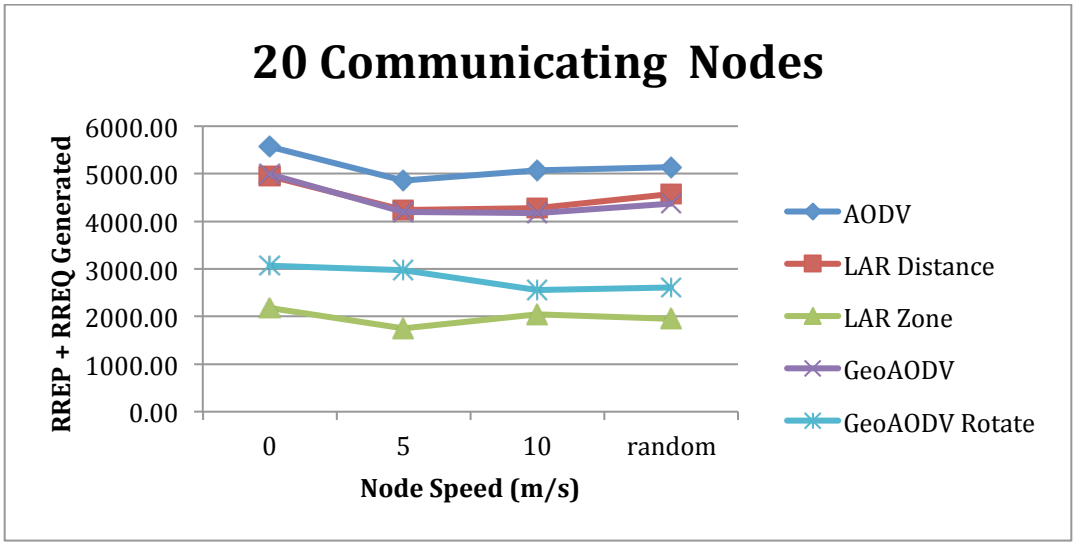

Figure 5.7 Number of Control Packets in Scenarios with 20 Communicating Nodes

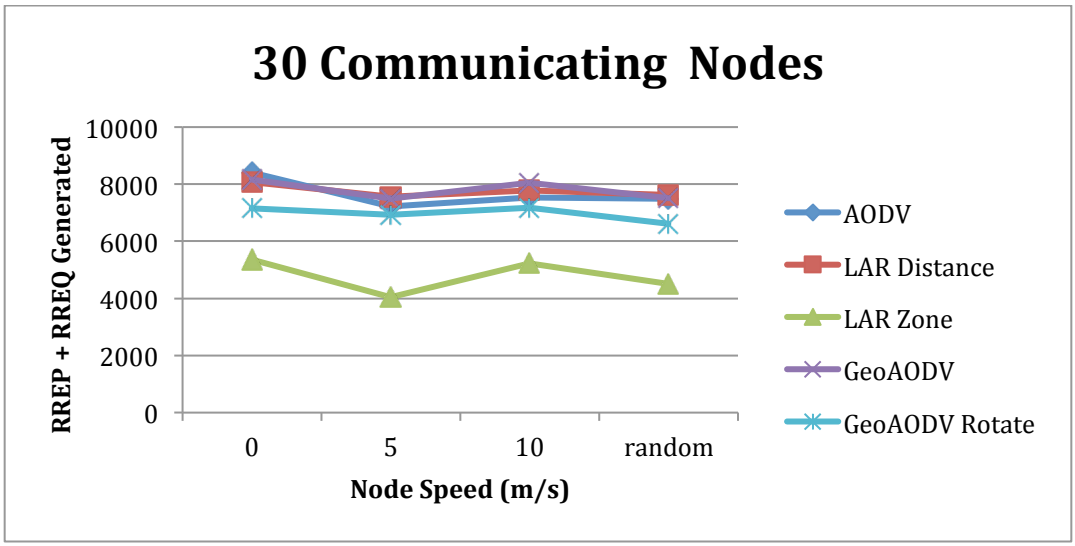

Figure 5.8 Number of Control Packets in Scenarios with 30 Communicating Nodes

Comparing the GeoAODV protocols showed that GeoAODV Rotate consistently outperforms GeoAODV Static. While both variations of GeoAODV exhibit roughly the same number of failures at finding a route to the destination, GeoAODV Rotate

dynamically reorients the direction of the request zone and thus excludes more nodes that are unlikely to be part of a route to the destination. This results in GeoAODV Rotate forwarding fewer RREQ packets through the network and thus introducing lower control traffic overhead than does the GeoAODV Static protocol.

Collected results show that the control traffic overhead increases correspondingly with the increase in the number of communicating nodes, as expected. What was surprising was how all the protocols, except for LAR Zone, performed in the simulation scenario with 30 communication nodes. As shown in Figure 5.8, LAR Distance, GeoAODV Static, and GeoAODV Rotate generated almost the same amount of control traffic as did AODV. Such behavior could be attributed to the fact that when there are many communicating nodes, the chance of failing to find a route using limited broadcast increases. This would cause these protocols to revert to regular AODV more frequently. As a result, all of the advantages gained by successfully employing limited flooding are lost when the protocols fail to find a route and have to conduct a network-wide flooding. LAR Zone appears to be less susceptible to this problem and exhibits the best performance, as shown in Figure 5.8. Nevertheless, the GeoAODV Rotate protocol consistently remains a second-best option, outperforming all of the other protocols, except for LAR zone, in all evaluated scenarios.

Chapter 6

Conclusions

This research indicates that the GeoAODV Rotate protocol could be a viable option for routing traffic in a MANET where the nodes are equipped with GPS devices. Specifically, the results of the simulation study indicate that GeoAODV Rotate outperforms the AODV, LAR Distance, and GeoAODV protocols. In many cases, its performance is comparable to LAR Zone. It should also be noted that, in this simulation study, it was assumed that LAR could retrieve location information instantaneously and that the delay introduced by retrieving such information was unaccounted for.

One of the possible reasons for the performance differences between GeoAODV and LAR Zone is the fact that LAR implementations assume location information to be globally available on-demand. GeoAODV makes no such assumptions and dynamically distributes location information during the route discovery phase. Also, GeoAODV must distribute geographical information throughout the network before the geo-assisted optimizations can be utilized. During the time that this information is being distributed, GeoAODV must operate the same way as AODV. After having achieved this distribution, GeoAODV will begin to take advantage of the limited flooding zone. Furthermore, GeoAODV may go through the route discovery phase up to three times (potentially more, if the flooding angle increment was set to a smaller value) using different values of the flooding angle (i.e., 90°, 180°, 270°) before it gives up and conducts route discovery using the AODV protocol (i.e., 360°). The LAR protocols, on the other hand, revert to AODV after a single failure.

The difference in the number of control packets generated between the LAR Distance and LAR Zone protocols can be accredited to the fact that the LAR Distance protocol reverts to the regular AODV route discovery process more often than does LAR Zone. The search area for LAR Distance is very limited, (i.e., the LAR Distance node only rebroadcasts an RREQ message if it is closer to the destination than was the previous node). Thus, it is no surprise that LAR Distance does not find a route to the destination more frequently than does LAR Zone. This ultimately causes LAR Distance to behave like AODV and generate a relatively large number of control packets. On the other hand, LAR Zone is not as restrictive and thus will revert to AODV less often. This results in LAR Zone consistently outperforming the LAR Distance protocol.

GeoAODV Rotate also consistently outperformed GeoAODV (GeoAODV Static). While both variations exhibit a similar number of failures at finding a route to the destination, GeoAODV Rotate dramatically reduces the number of forwarded control packets by dynamically adjusting the request zone during the route discovery process. Thus, GeoAODV Rotate introduces fewer control packets into the network than does GeoAODV Static.

Results show that there is a direct correlation between the number of communicating nodes and the number of control packets traveling through the network. As the number of communicating nodes increases, so too does the number of nodes that initiate the route discovery procedure and generate control traffic. As expected, an increase in the number

of communicating nodes corresponds to an increased presence of control traffic in the network. It was surprising to observe that, in a simulation study with 30 communicating nodes, the performances of GeoAODV Static and LAR Distance protocols were similar to that of AODV, rather than to GeoAODV Rotate and LAR Zone. One possible explanation for this behavior could be the increased likelihood for the GeoAODV Static and LAR Distance protocols to fail to find a route and revert to AODV in the high-traffic simulation. Overall, GeoAODV Rotate was consistently the second-most efficient routing choice and is the clear option in networks where geographical information is not available on-demand and must be distributed.

Future Work

While this study has been completed, there are numerous directions for further investigation of location-aided routing. Specifically, it would be interesting to see how GeoAODV Rotate performs in different environmental settings. Additionally, there is a need to develop mechanisms for more accurate incrementing of the flooding angle value after route discovery failures. Future studies could rerun created simulation models with a larger number of repetitions (each with a different seed value), and further analyze the collected results. A study examining the performance of the GeoAODV Rotate during the pre- and post-convergence periods and how fast GeoAODV Rotate converges to a stable state could shed light on how GeoAODV Rotate compares to LAR Zone once the network has converged. Studying how accurately GeoAODV Rotate distributes location information in the network could help improve the performance of the protocol.

Additionally, a new study of location-aided routing could focus on other aspects of protocol performance (e.g., the number of route discovery failures, the time to find a route to destination). It may focus on possible optimizations of the GeoAODV Rotate protocol, including a more intelligent selection of the initial value of the flooding angle and dynamically adjusting the flooding angle at intermediate nodes (i.e., increasing the flooding angle value when an intermediate node knows that there are no neighboring nodes within the request zone defined by the flooding angle). Another direction that research could take is exploring improvements to the LAR protocols. This would allow for increasing the search area after a route discovery failure, instead of immediately reverting back to AODV. One possibility would be for LAR Distance to adjust the values of configuration parameters α and β (seen in Figure 2.2) or for LAR Zone to expand the request zone by transmitting the source node coordinates as if the source node is located farther away from the destination node than it really is. Lastly, the study of location-aided routing could be expanded to other protocols (e.g., the Greedy Perimeter Stateless Routing (GPSR) protocol [23], the Geographical Routing Protocol (GRP) [24]).

References

[1] J Li, C Blake, D De Couto, H Imm ,L R Morris, "Capacity of Ad Hoc Wireless Networks", Proceeding MobiCom '01 Proceedings of the 7th annual international conference on Mobile computing and networking, pp. 61-69, 2001

[2] Perkins, C.; Belding-Royer, E.; Das, S. (July 2003). *Ad hoc On-Demand Distance Vector (AODV) Routing*. IETF. RFC 3561. Retrieved 2012-01-26.

[3] Y. Ko and N. H. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks," Wireless Networks, 6(4), July 2000, pp. 307-321.

[4] Young-Bae Ko and Nitin H. Vaidya, "Flooding-Based Geocasting Protocols for Mobile Ad Hoc Networks." Mobile Networks and Applications archive, Volume 7, Issue 6, December 2002, Pages 471 – 480

[5] Sidi-Mohammed Senouci and Tinku Mohamed Rasheed, "Modified Location-Aided Routing Protocols for Control Overhead Reduction in Mobile Ad Hoc Networks," Telecommunications, 2003. ICT 2003. 10th International Conference on, 23 Feb 23rd 2003 - Mar 1st 2003

[6] Jun Sen, Kun Yang, and Shaochun Zhong, "A Prediction Based Location Update Algorithm in Wireless Mobile Ad-Hoc Networks," International Conference on Computer Networks and Mobile Computing, 2005

[7] Jiwei Chen, He Zhou, Yeng-zhong Lee, Mario Gerla, and Yantai Shu, "AODV-DFR: Improving Ad Hoc Routing Scalability to Mobility and Load, "2006 IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS), , Oct. 2006

[8] Jiwei Chen, Yeng-Zhong Lee, He Zhou, Mario Gerla, and Yantai Shu, "Robust Ad Hoc Routing for Lossy Wireless Environment," Military Communications Conference, 2006. MILCOM 2006. IEEE. 23-25 Oct. 2006

[9] Jiwei Chen, Mario Gerla and Yeng Zhong Lee, “TCP Performance over Geo-routing for High Mobile Ad Hoc Networks,” Wireless Communications & Mobile Computing, Volume 4 Issue 2, March 2004. Pages 203 – 222

[10] Konglin Zhu, Biao Zhou, Xiaoming Fu and Mario Gerla, “Geo-assisted Multicast Inter-Domain Routing (GMIDR) Protocol for MANETs,” 2011 IEEE International Conference on Communications (ICC), 5-9 June 2011

[11] V. Hnatyshin, and H. Asenov, "Design and Implementation of an OPNET model for simulating GeoAODV MANET routing protocol", Proc. of the OPNETWORK 2010 International Conference, Session: Wireless Ad Hoc and Wireless Personal Area Networks, Washington DC, August 2010.

[12] H. Asenov, and V. Hnatyshin, "GPS-Enhanced AODV routing," in Proceedings of the 2009 International Conference on Wireless Networks (ICWN'09), Las Vegas, Nevada, USA (July 13-16, 2009)

[13] Vasil Hnatyshin, Remo Cocco, Malik Ahmed, Dan Urbano, "Improving Geographical AODV Protocol by Dynamically Adjusting the Request Zone," In Proc. of OPNETWORK 2012 International Conference, Washington, DC, August 2012

[14] Remo Cocco, Vasil Hnatyshin, Malik Ahmed, Dan Urbano, "Improving Geographical AODV Protocol by Dynamically Adjusting the Request Zone," presented at 35th IEEE Sarnoff Symposium, Newark, NJ 2012. Short paper

[15] V. Hnatyshin, M. Ahmed, R. Cocco, and D. Urbano, "A Comparative Study of Location Aided Routing Protocols for MANET", In Proceedings of 4th IEEE IFIP Wireless Days 2011 conference, Niagara Falls, Canada (PDF).

[16] S. Basagni, I. Chlamtac, V. R. Syrotiuk etal. “A distance routing effect algorithm for mobility (DREAM).” The ACM/IEEE Int' l Conf on Mobile Computing and Networking (MOBICOM), Dallas, 1998

[17] T. Camp, J. Boleng, and L. Wilcox. “Location information services in mobile ad hoc networks”. In Proceedings of ICC, 2001.

[18] Vasil Hnatyshin, Hristo Asenov, and John Robinson, “PRACTICAL METHODOLOGY FOR MODELING WIRELESS ROUTING PROTOCOLS USING OPNET MODELER”,

[19] OPNET Modeler ver. 16.0. OPNET Technologies, Inc®, www.opnet.com last visited 6/12/12.

[20] E. M. Royer and C. E. Perkins. An Implementation Study of the AODV Routing Protocol, Proc. of the IEEE Wireless Communications and Networking Conference, Chicago, IL, September 2000.

[21] C. E. Perkins and E. M. Royer. Ad hoc On-Demand Distance Vector Routing, Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, February 1999.

[22] D. Espes, Z. Mammeri. Adaptive expanding search methods to improve AODV Protocol, IST Mobile and Wireless Communications Summit, July 2005.

[23] B. Karp and H. T. Kung, “GPSR: greedy perimeter stateless routing for wireless networks,” in MobiCom ’00: Proceedings of the 6th annual international conference on Mobile computing and networking. New York, NY, USA: ACM Press, August 2000, pp. 243–254.

[24] OPNET Modeler 16.1 Documentation, OPNET Technologies, Inc., 2012.

Appendix A Initiation Functions

```
// Purpose: Initialize global GeoAODV parameters
// In/Out:  NONE
static void  aodv_rte_geo_init() {
  Objid aodv_parms_id;
  Objid aodv_parms_child_id;
  Objid parent_id;
  char name[128];
  double x, y;
  double LAR_update_start_time;

  FIN (aodv_rte_geo_init());

  parent_id = op_topo_parent(own_mod_objid);
  op_ima_obj_attr_get(parent_id, "name", name);
  op_ima_obj_attr_get(parent_id, "x position", &x);
  op_ima_obj_attr_get(parent_id, "y position", &y);
  parent_id = op_topo_parent(parent_id);
  op_ima_obj_attr_get(parent_id, "name", name);

  /* Read the AODV Parameters  */
  op_ima_obj_attr_get (own_mod_objid,
    "manet_mgr.AODV Parameters", &aodv_parms_id);
  aodv_parms_child_id = op_topo_child (aodv_parms_id,
    OPC_OBJTYPE_GENERIC, 0);

  // Read GeoType
  op_ima_obj_attr_get(aodv_parms_child_id, "Position-Based Routing",
    &geo_routing_type);

  op_ima_obj_attr_get(aodv_parms_child_id, "LAR Update Interval",
    &LAR_update_interval);
  op_ima_obj_attr_get(aodv_parms_child_id, "LAR Update Start Time",
    &LAR_update_start_time);

  // Attribute for GeoExpand.
  op_ima_obj_attr_get(aodv_parms_child_id, "GeoExpand Angle Padding",
    &angle_padding);

  op_ima_obj_attr_get(aodv_parms_child_id, "Node Location DB",
    &location_data_distributed);

  // Initialize location databases
  if ( geo_routing_type != AODV_TYPE_REGULAR ){
    op_intrpt_schedule_self (op_sim_time() + LAR_update_start_time,
      AODVC_LAR_UPDATE);

    aodv_geo_LAR_init( module_data_ptr, aodv_addressing_mode, x, y );
  }

  // The geo table is always created, whether we're using
  //distributed or centralized.
  geo_table_ptr = aodv_geo_table_create(aodv_addressing_mode);

  FOUT;
}
```

```
// Purpose:  This function performs initializations for LAR, including
//           inserting the initial LAR_Data entries for the global
//           database.
//
// In:       module_data_ptr - a pointer to the node's module
//           data (used to retrieve the IP. (usually retrieved
//           like this: (IpT_Rte_Module_Data*) op_pro_modmem_access ())
//           address_mode - the address mode we're using (this
//           format will be used for the IPs used as keys for storing
//           LAR_Data).
//           x, y - the node's initial position.
void aodv_geo_LAR_init( IpT_Rte_Module_Data* module_data_ptr,
InetT_Addr_Family address_mode, double x, double y ) {

  // The following variables are used to initialize
  // data for LAR updates.
  LAR_Data *lar_data;

  // Added to put a false entry in the central hello messages.
  // This is because nodes should insert its entry into the table
  // when it is about to send a hello messsage.
  LAR_Data *hello_message_database_invalid_data;
  int num_interfaces;
  int ifnum;
  cha address[INETC_ADDR_STR_LEN];

  FIN (aodv_geo_LAR_init( <args> ));

  aodv_addressing_mode = address_mode;

  // Store initial position
  // You want to store the following information:
  // 1. x, y coordinates
  // 2. time when they were recorded
  // 3. velocity

  // Create the initial entry in the global database,
  // which will be updated at each LAR interrupt.
  lar_data = new_LAR_Data(x, y);
  hello_message_database_invalid_data =
    new_LAR_Data(DEFAULT_X, DEFAULT_Y);
  num_interfaces = inet_rte_num_interfaces_get (module_data_ptr);

  for (ifnum = 0; ifnum < num_interfaces; ifnum++) {
    // In case there are multiple interfaces at this node,
    // create an entry for each one referencing the same data
    // so that no matter which IP the data is pulled from,
    // the data will be the same.
    get_node_ip(address, module_data_ptr, ifnum);
    oms_data_def_entry_insert(LAR_OMS_CATEGORY, address, lar_data);
    oms_data_def_entry_insert(HELLO_OMS_CATEGORY, address,
      hello_message_database_invalid_data);
  }

  FOUT;
}
```

Appendix B Modified AODV Packet Strucutres

```
/* Route Request Option */
typedef struct {
  Boolean join_flag;
  Boolean repair_flag;
  Boolean grat_rrep_flag;
  Boolean dest_only;
  Boolean unknown_seq_num_flag;
  int hop_count;
  int rreq_id;
  InetT_Address dest_addr;
  int dest_seq_num;
  InetT_Address src_addr;
  int src_seq_num;
  AodvT_LAR_Info geo_lar_options;
} AodvT_Rreq;

/* Route Reply Option */
typedef struct {
  Boolean repair_flag;
  Boolean ack_required_flag;
  int hop_count;
  InetT_Address dest_addr;
  int dest_seq_num;
  InetT_Address src_addr;
  double lifetime;
  double dst_x;
  double dst_y;
} AodvT_Rrep;

/* Encapsulate all GEO/LAR options */
typedef struct {
  Point2D src;
  Point2D prev;
  Point2D dst;
  int request_level;
  double velocity;
} AodvT_LAR_Info;
```

Appendix C RREQ Rebroadcast Logic and Functions

```
static void aodv_rte_rreq_pkt_arrival_handle (Packet* ip_pkptr,
  Packet* aodv_pkptr, IpT_Dgram_Fields* ip_dgram_fd_ptr,
  IpT_Rte_Ind_Ici_Fields* intf_ici_fdstruct_ptr,
  AodvT_Packet_Option* tlv_options_ptr) {

  /* Code excluded for brevity */

  // Destroy the packet if we shouldn't rebroadcast.
  if (aodv_geo_rebroadcast(
      geo_lar_options->src.x, geo_lar_options->src.y,
      prev_x,  prev_y,
      curr_x,  curr_y,
      geo_lar_options->dst.x, geo_lar_options->dst.y,
      (double) ((geo_lar_options->request_level+1) * 90),
      angle_padding,
      geo_routing_type,
      geo_lar_options->velocity
    ) == OPC_FALSE) {
    op_pk_destroy (aodv_pkptr);
    manet_rte_ip_pkt_destroy (ip_pkptr);
    FOUT;
  }

  /* Code excluded for brevity */
}

// Purpose: Given positions of the nodes, flooding angle, and aodv
//          type determine if the current node should rebroadcast
//          RREQ or not
//
// In:      orig_x, orig_y - position of the node that originated
//                           the RREQ
//          prev_x, prev_y - position of the node where the RREQ was
//                           received from
//          curr_x, curr_y - position of the node that received the
//                           RREQ
//          dest_x, dest_y - position of the destination node
//          flooding_angle - acceptable angle to forward the RREQ
//          aodv_type      - type of aodv being used
//
// Out:     TRUE if the current node should rebroadcast the RREQ
//          FALSE if the RREQ should be destroyed
Boolean aodv_geo_rebroadcast(
  // Coordinates of the node that originated RREQ
  double orig_x, double orig_y,
  // Coordinates of the node that send RREQ
  double prev_x, double prev_y,
  // Coordinates of the node that received RREQ
  double curr_x, double curr_y,
  // Coordinates of the destination node
  double dest_x, double dest_y,
  // Angle in degrees of the flooding angle
  double flooding_angle,
  // The maximum value by which the flooding angle
  // can expand (for Geo_Expand only)
  double angle_padding,
```

```
// Type of AODV being used
int aodv_type,
// The calculated velocity of the destination node (LAR)
double dest_velocity) {

double angle;

FIN (aodv_geo_rebroadcast( <args> ));

if (flooding_angle >= MAX_ANGLE) {
  // We're flooding, so you have to rebroadcast.
  // This takes care of regular AODV too since flooding_angle will
  // always be 360 for regular AODV when it is computed in
  // aodv_geo_compute_expand_flooding_angle.
  FRET (OPC_TRUE);
}

// If we're not in broadcast mode, we can do what each type of AODV
// would normally do.

switch(aodv_type) {
  case (AODV_TYPE_LAR_DISTANCE):
    // if current node is at least as close as the previous node
    // from destination then rebroadcast RREQ (return true), else
    // drop (return false)
    FRET(aodv_geo_LAR_distance(prev_x, prev_y,
      curr_x, curr_y,
      dest_x, dest_y)
    );

  case AODV_TYPE_GEO_STATIC:
    // GeoAODV implementation:
    // Compute the angle formed by the destination, source and
    // current nodes. If computed angle is not larger than flooding
    // angle (e.g. the value is carried via request level) then
    // forward RREQ, else drop RREQ

    // Check if this is not a broadcast
    if(flooding_angle < 360) {

      // Compute the angle formed by the destination node,
      // originating node, and current node. Since the angle may be
      // located on either side of the vector formed by the
      // source-destination nodes we need to multiply the computed
      // value of angle by 2 before comparing it to the value of the
      // flooding angle, so that flooding angle is evenly devided by
      // the line formed via source-destination nodes
      angle = 2 * aodv_geo_compute_angle(dest_x, dest_y,
        orig_x, orig_y,
        curr_x, curr_y
      );

      if (angle > flooding_angle) {
        FRET(angle <= angle_padding)
      }
    }

    // This is NOT a broadcast or the angle formed by orig, curr,
```

```
      // and dest node is less than flooding angle
      FRET(OPC_TRUE);

    case AODV_TYPE_GEO_ROTATE:
      // GeoAODV Rotate implementation:

      // Set flooding angle to initial value degrees, forward to all
      // neighbors in the search area formed by the flooding angle
      // if fails to find the route then increment flooding angle
      // until it reaches 360 degrees and morphs into regular AODV

      // NOTE: angle at the intermediate node is computed based on the
      // previous node location

      // Check if this is not a broadcast
      if(flooding_angle < MAX_ANGLE) {
        // Compute the angle formed by the destination node,
        // originating node, and previous node. Since the angle may be
        // located on either side of the vector formed by the
        // previous-destination nodes we need to multiple the computed
        // value of angle by 2 before comparing it to the value of the
        // flooding angle, e.g. flooding angle is evenly divided by
        // the line formed via prev-destination nodes
        angle = 2 * aodv_geo_compute_angle(dest_x, dest_y,
          prev_x, prev_y,
          curr_x, curr_y
        );

        if (angle > flooding_angle) {
          FRET(OPC_FALSE);
        }
      }

      // This is NOT a broadcast or the angle formed by orig, curr, and
      // dest node is less than flooding angle
      FRET(OPC_TRUE);

    case AODV_TYPE_LAR_ZONE:
      FRET(aodv_geo_LAR_within_request_zone(orig_x, orig_y,
        curr_x, curr_y,
        dest_x, dest_y,
        dest_velocity
      ));
    case AODV_TYPE_REGULAR:
      // Always rebroadcast in AODV
      FRET(OPC_TRUE);
  }

  FRET (OPC_TRUE);
}


// Purpose: Determine if the length of the vector formed by
//          start-end  points(vector SE) is
//          greater than the length of the vector formed by middle-end
//          points(vector ME)
//
// In:      start_x, start_y - position of node where the RREQ was
```

```
//                              generated (e.g. previous node, not
//                              an originator)
//          mid_x, mid_y - position of node that receives the RREQ
//          end_x, end_y - position of the destination node
//
// Out:     True,  if length(SE) >= length (ME)
//          False, otherwise
Boolean aodv_geo_LAR_distance(double start_x, double start_y,
  double mid_x,   double mid_y,
  double end_x,   double end_y) {

  FIN (aodv_rte_rreq_within_distance( <args> ));


  if (aodv_geo_vector_length(start_x, start_y, end_x, end_y) >=
      aodv_geo_vector_length(mid_x, mid_y, end_x, end_y)) {
    FRET(OPC_TRUE);
  }

  FRET(OPC_FALSE);
}

// Purpose: Compute the angle SME formed by three points:
//          start (S), middle (M), end (E)
//
// In:      start_x, start_y -- position of starting point S
//          mid_x, mid_y -- position of the middle point M
//          end_x, end_y -- position of ending point E
//
// Out:     A value of the angle formed by the points S, M, E in units
//          of degrees
double aodv_geo_compute_angle(double start_x, double start_y,
  double mid_x, double mid_y,
  double end_x, double end_y) {

  double vector_MS_x;
  double vector_MS_y;

  double vector_ME_x;
  double vector_ME_y;

  double angle_form_numer;
  double angle_form_denom;

  double angle;

  FIN (aodv_geo_compute_angle( <args> ));

  vector_MS_x = mid_x - start_x;
  vector_MS_y = mid_y - start_y;

  vector_ME_x = mid_x - end_x;
  vector_ME_y = mid_y - end_y;

  angle_form_numer = (vector_MS_x * vector_ME_x) +
    (vector_MS_y * vector_ME_y);

  angle_form_denom =
```

```
    aodv_geo_vector_length(mid_x, mid_y, start_x, start_y) *
    aodv_geo_vector_length(mid_x, mid_y, end_x, end_y);

  angle = acos(angle_form_numer / angle_form_denom) * (180 / PI);

  FRET(angle);
}

// Purpose: This method returns whether or not the current node is
//          within the request zone. The request zone will be as
//          specified by LAR Zone
//
// In:      src_x, src_y - the coordinates of the originating source
//                         node
//          curr_x, curr_y - the coordinates of the node to test
//          dest_x, dest_y - the coordinates of the destination node.
//          radius - the velocity of the destination, or the radius of
//                   the expected zone per LAR1.
//
// Out:     OPC_TRUE if the node is within the request zone and
//          OPC_FALSE otherwise.
Boolean aodv_geo_LAR_within_request_zone(double src_x, double src_y,
  double curr_x, double curr_y,
  double dest_x, double dest_y,
  double radius) {

  // The corners of the rectangular request zone: ll = lower-left,
  // ul = upper-left, ur = upper-right, lr = lower-right.
  Point2D ll, ul, ur, lr;

  // The location of the current node.
  Point2D currentLocation;

  // This is the request zone rectangle made up of the four points
  // above.
  Rectangle requestZone;

  // The return value for this method (whether or not the current node
  // is contained within the request zone.
  Boolean contained;

  FIN (aodv_geo_LAR_within_request_zone( <args> ));

  currentLocation.x = curr_x;
  currentLocation.y = curr_y;

  // The lower-left corner of the request zone is as far left and as
  // far down as possible.
  ll.x = min(src_x, dest_x - radius);
  ll.y = min(src_y, dest_y - radius);

  // The upper-left corner of the request zone must be as far left and
  // as far up as possible.
  ul.x = ll.x;
  ul.y = max(src_y, dest_y + radius);

  // The upper-right corner of the request zone must be as far right
  // and as far up as possible.
```

```
  ur.x = max(src_x, dest_x + radius);
  ur.y = ul.y;

  // The lower-right corner of the request zone must be as far right
  // and as far down as possible.
  lr.x = ur.x;
  lr.y = ll.y;

  //Encapsulate these four points into a rectangle.
  requestZone.lower_left = ll;
  requestZone.upper_left = ul;
  requestZone.upper_right = ur;
  requestZone.lower_right = lr;

  contained =
    aodv_geo_LAR_is_point_contained(&currentLocation, &requestZone);

  FRET ( contained );
}

// Purpose: Simple helper function that determines whether or not the
//          given Point is within the bounds of the provided Rectangle.
//
// In:      location - the point to check.
//          zone - the bounds to check against.
//
// Out:     OPC_TRUE if the given location is contained within the zone,
//          and OPC_FALSE otherwise.
Boolean aodv_geo_LAR_is_point_contained(Point2D *location,
  Rectangle *zone) {

  FIN (aodv_geo_LAR_is_point_contained( <args> ));

  // assumes that all sides of the
  // rectangle are parallel to their respective axes

  // left of rectangle
  if (location->x < zone->upper_left.x)
    FRET ( OPC_FALSE );

  // above the rectangle
  if (location->y > zone->upper_left.y)
    FRET ( OPC_FALSE );

  // right of the rectangle
  if (location->x > zone->upper_right.x)
    FRET ( OPC_FALSE );

  // below rectangle
  if (location->y < zone->lower_right.y)
    FRET ( OPC_FALSE );

  // Otherwise, it's in the rectangle.
  FRET (OPC_TRUE);

}

// Purpose: Compute the length of the vector
```

```
//
// In:      start_x, start_y - starting point of the vector
//          end_x, end_y - ending point of the vector
//
// Out:     length of the vector
double aodv_geo_vector_length(double start_x, double start_y,
  double end_x,   double end_y) {

  double x, y;

  FIN (aodv_geo_vector_length( <args> ));

  x = end_x - start_x;
  y = end_y - start_y;

  FRET (sqrt(pow(x, 2.0) + pow(y, 2.0)));
}
```

Appendix D Raw Results

2 Communicating Nodes

	Speed (m/s)			
Protocol	0	5	10	Random
AODV	188.67	161.50	174.17	164.33
LAR Distance	141.33	134.67	141.17	138.17
LAR Zone	93.17	93.50	120.17	87.50
GeoAODV	156.33	163.50	144.33	115.83
GeoAODV Rotate	119.67	108.83	132.67	94.33

5 Communicating Nodes

	Speed (m/s)			
Protocol	0	5	10	Random
AODV	714.17	592.17	632.00	654.17
LAR Distance	436.00	431.17	429.67	416.00
LAR Zone	260.33	324.00	286.67	279.00
GeoAODV	462.83	379.00	461.67	399.83
GeoAODV Rotate	342.67	377.50	388.67	378.50

10 Communicating Nodes

	Speed (m/s)			
Protocol	0	5	10	Random
AODV	2283.67	2080.00	2185.83	2208.50
LAR Distance	989.83	1105.17	1305.00	1129.17
LAR Zone	623.33	759.50	764.17	704.83
GeoAODV	1235.67	1404.83	1144.83	1214.33
GeoAODV Rotate	720.67	786.00	862.67	816.00

20 Communicating Nodes

	Speed (m/s)			
Protocol	0	5	10	Random
AODV	5570.33	4850.83	5066.67	5139.17
LAR Distance	4944.83	4232.83	4272.17	4577.17
LAR Zone	2177.67	1751.00	2043.33	1946.67
GeoAODV	4994.17	4198.00	4173.67	4367.50
GeoAODV Rotate	3069.00	2969.83	2555.83	2617.17

30 Communicating Nodes

	Speed (m/s)			
Protocol	0	5	10	Random
AODV	8411.00	7213.67	7537.00	7498.67
LAR Distance	8071.67	7563.00	7789.17	7628.50
LAR Zone	5359.50	4042.50	5218.17	4519.17
GeoAODV	8169.17	7496.33	8050.00	7508.83
GeoAODV Rotate	7153.00	6924.17	7183.33	6606.00