Load Reduction in Ad Hoc Networks Using Mobile Servers

by
Viswanath Tolety

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science (Mathematical and Computer Sciences).

Golden, Colorado

Date _____

Signed: _____
Viswanath Tolety

Approved: _____
Dr. Tracy Camp
Associate Professor

Golden, Colorado

Date _____

_____
Dr. Graeme Fairweather
Professor and Head
Department of Mathematical and Computer Sciences

ABSTRACT

In this thesis, we propose two algorithms for network load reduction in ad hoc networks. With the increasing popularity of mobile computing and Internet based client-server applications, such algorithms should become popular.

Our algorithms are based on allowing the server to be mobile using mobile agent technology. Some of the mobile ad hoc network based applications that we have identified are distributed games, distributed databases, and chat rooms.

To illustrate our algorithms, and to evaluate their performance with respect to various parameters, we simulate a chat room server application. Lastly, we propose modifications to our algorithms which may lead to better performance in specific situations.

ACKNOWLEDGMENTS

Working with Dr. Tracy Camp has been a great experience for me. Her suggestions have proved to be invaluable in carrying out the thesis work. I would like to thank Dr. Raghu Krishnapuram for his continued support. I would also like to thank him for his suggestions in writing the thesis. I would like to thank Dr. Hugh King for his suggestions towards improving the thesis. I would like to thank the Mathematics and Computer Sciences Department for providing me with financial assistance during the course of the program. Lastly, I would like to thank members of the Toilers research group for their suggestions.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Introduction

Compared to the computers in the early years of computerization, today's computers are much smaller with more memory and computational power. Laptop computers and notebook computers are a common sight and have become very popular in a relatively short period of time. The miniaturization of computers made it possible for people to carry their computers with them.

The popularity of mobile devices can be gauged from the following two figures: there are about 40 million mobile cellular subscribers in the United States; in North America, there is expected to be 80 million cellular and personal communication subscribers by the year 2000. With the increasing popularity of the Internet, due to applications like multi-media and e-commerce, a proportional growth in demand for such facilities on mobile devices is expected.

In Section 1.2, we discuss issues in mobile computing and the solutions proposed in the literature. In Section 1.3, we discuss the issues and applications of a mobile ad hoc network. In Sections 1.4 and 1.5, we describe mobile agent and CORBA technologies, which we use in implementing our algorithms. Lastly, in Section 1.6, we discuss the rationale behind our algorithms and describe applications for our algorithms.

1.2 Mobile Computing

One solution to the problem of network connectivity of a mobile computer is having it connect to different points in a fixed network as it moves along. One problem that arises in such an implementation is the fact that in traditional fixed networks, like the Internet, the addressing and consequently the routing is based on the location of the end system. Obviously such protocols do not apply to a network where the nodes are moving and thus do not have a fixed location. This problem has been addressed, by the Internet Engineering Task Force, and a new protocol has been formulated and introduced into our Internet standards. The protocol is called Mobile IP [2, 3], an extension to the Internet Protocol (IP) used in the Internet today.

In Mobile IP every mobile node (MN) has a home agent (HA). An HA acts as a home agent for all the MNs in its network and handles all their mobility concerns. An MN is said to be "away from home" if its

current link-level point of attachment is not its home network. A wireless computer network consists of a set of base stations (bases) which provide link-level attachment points to the MNs in their cells. The cells are connected, adjacent or overlapping.

As an MN moves around in the network it may decide to switch its link level point of attachment to a new base. The new base assigns the MN a temporary care-of-address and a lifetime, and transmits these values to the MN's HA. This process is called registration. All the packets for an MN are transmitted to the MN's home address due to the location dependency of IP addresses; the HA intercepts the packets and forwards them to the MN's care-of-address. When the MN is no longer away from home it asks its HA to remove its care-of-address.

The above Mobile IP scheme has deficiencies. First, the host trying to communicate with the MN may be very near to the MN but far away from the MN's HA, thus leading to inefficient routing. This problem is termed the triangle routing problem, as a triangle is formed when the MN responds to the host's message. Extensions to Mobile IP for coping with the triangle routing problem have been proposed [4]. A second problem arises when the MN moves far from home, i.e., the cost of updating the HA may be quite high, even for small movements of the MN. Much research is being done to try and reduce this problem.

Algorithms have also been developed which determine the location of the mobile node (location management techniques) [15] and for determining when a mobile node should change its point of connection to the fixed network (cell switching).

1.3 Mobile Ad hoc Networks (MANET)

A mobile ad hoc network (MANET) [17] consists of a collection of autonomous systems connected through wireless links. The network is formed in an on-demand fashion and no fixed configuration exists. There are various scenarios where the creation of an ad hoc network is useful:

- emergency operations: a natural or man made disaster requires rapid installation of a new communication medium;
- law enforcement operations: rapid installation of a communication infrastructure is needed during special operations;
- tactical missions: rapid installation of a communication medium is required in an unknown territory and/or a difficult terrain;

- commercial projects: simple installation of a communication infrastructure at conferences, workshops, etc. is desired;
- educational classrooms: creation of an interactive classroom may need to be formed on demand.

Some of the challenges involved in the creation of an ad hoc network are:

- routing challenges: routing in a dynamically changing environment;
- wireless medium challenges: lower bandwidth, higher error rates, frequent disconnections and less security compared to fiber carriers;
- portability challenges: lower power and smaller storage capacity compared to desktop computers.

Various protocols [12] have been proposed for routing in an ad hoc network. The protocols can be classified into two categories:

1. Pro-active protocols: These table driven routing protocols maintain consistent up-to-date information from each MN to every other MN in the network. Examples of pro-active protocols include Destination Sequenced Distance Vector Routing (DSDV) [6], Clusterhead Gateway Switch Routing (CGSR) [1] and Wireless Routing Protocol (WRP) [23].
2. Reactive Protocols: These source initiated on demand protocols create a route only when desired by the source MN. Examples of reactive protocols include Ad hoc On-Demand Distance Vector Routing (AODV) [5], Dynamic Source Routing (DSR) [8], Temporally-Ordered Routing Algorithm (TORA) [25], Associativity-Based Routing (ABR) [7] and Signal Stability Routing (SSR) [21].

Most of the above protocols keep track of their immediate neighbors as part of the routing algorithm; we assume this neighborhood knowledge exists in our work.

Many of the above algorithms initiate a broadcast search for an MN when a communication with the MN is needed. Two recent protocols have been proposed which use location information on the MNs to restrict the search area. The Location Aided Routing (LAR) protocol [26] determines a request zone based on the location and velocity of the destination MN. An extension of LAR has also been proposed for location based multicasting, i.e., geocasting [27]. The Distance Routing Effect Algorithm for Mobility (DREAM) [22] is based on the change in angle of the destination MN with respect to the source MN. Since the change in the angle is inversely proportional to the distance between the MNs, DREAM updates the routing tables as

a function of distance, thus restricting the number of updates due to MN movement. Location information in the above schemes is obtained using the Global Positioning System (GPS).

GPS allows precise determination of location, velocity, direction, and time. GPS receivers are available for notebook personal computers and, with increasing usage and improvement in the technology, the prices are decreasing rapidly. Apart from general applications such as vehicle navigation and aviation systems, GPS can be used for various applications in the area of distributed systems and networks [13], e.g., circuit switching using synchronized clocks, transmission in synchronous slotted systems, clock synchronization in distributed systems, database synchronization, directional antennas, distributed robot control and navigation, and equipment location marking for maintenance crews.

Client-Server applications have recently become very popular, particularly on the Internet. They are widely used for applications such as e-commerce and information retrieval. Some work has been done addressing the issues involved in client-server applications for mobile nodes in a Mobile IP type of environment over wireless links. Issues addressed are mobility of the server MN, limited bandwidth, and asymmetrical communication between client and server MNs. Client-Server applications in a MANET environment also use wireless channels and hence have some of the above problems. In fact, compared to a Mobile IP type of environment, the problems of limited bandwidth that come with wireless communication is compounded in a MANET.

In the following chapters, we develop an algorithm for the movement of a server in a client-server application such that it reduces the amount of client-server communication. Such an algorithm could yield substantial savings in bandwidth, particularly when the server has multiple mobile clients and dynamic changes in the client list. We also describe a few possible applications for our mobile server techniques and we simulate one of them to illustrate our algorithm. We use the Common Object Request Broker Architecture (CORBA) [19] and mobile agent technology [16] in developing the application. Sections 1.4 and 1.5 give an overview of these two technologies.

1.4 Mobile Agents

An agent can be defined as *"software that assist people on their behalf and are delegated to perform task(s), and given constraints under which they can operate"* [20]. Different types of agents exist based on the environment in which they operate. Many different classifications have been proposed; one such classification identifies three classes of agents:

1. goal-oriented agents: agents that do not simply act in response to the environment;

2. communicative agents: agents able to communicate with others; and

3. mobile agents: agents able to transport themselves from one host another.

Mobile agents are programs, written in a scripting language or some other platform independent language such as Java, able to migrate from one host machine to another host machine in a network. Work has been done in developing software packages that support mobile agents and in researching the advantages and disadvantages of using mobile agents in a particular application [9, 10]. Some of the technical hurdles that have been identified in using mobile agents are the following.

- Performance and Scalability: Since the agents are written in platform independent interpreted languages, they are relatively slow. When sufficient bandwidth is available these agents yield worse performance than traditional client-server applications such as remote procedure call (RPC) [9].

- Portability and Standardization: To have a mobile agent move from any machine to any other machine, a standard mobile code system must be established. The Object Management Group's (OMG) Mobile Agent System Interoperability Facility (MASIF) [11] defines part of the problem by addressing cross-system communication and administration.

- Security: Security has been the biggest concern for the proponents of mobile agent technology. The problem is partly solved by having access rights to protect against malicious agents [14].

Many applications for mobile agent technology have been identified. The relative advantages and disadvantages of using mobile agents for such applications over traditional alternatives like RPC have been studied [9]. Some of the advantages offered by mobile agents over traditional technologies are as follows.

- Support for mobile clients:

    1. since mobile clients have frequent disconnections from the network, one way of processing is to launch an agent when the client and server are connected and retrieve the information at a later stage;

    2. since mobile clients are connected through a low bandwidth wireless medium, launching an agent may be cheaper than sending messages back and forth between the client and the server.

- Real-time interaction with the server: When the latency in the network is high compared to the real-time constraints, it might be desirable to use a mobile agent (e.g., space probes).

Other advantages include robust queries/transactions, support for electronic commerce, less security overhead, better scalability over traditional RPC and intelligent mail handling.

1.5 CORBA

The Common Object Request Broker Architecture (CORBA) is a standard specified by the OMG for developing distributed object oriented applications [19]. The specifications are independent of operating systems and programming languages.

A CORBA Object Request Broker (ORB) connects a client application with the objects it wishes to use. CORBA provides location and language transparency, i.e., the clients of an object do not need to know the object's location or language in which it has been implemented. ORB takes care of locating the object, routing the request, and returning the result. Some of the features and benefits of CORBA are as follows [24].

- Choice: CORBA is an open published specification implemented and supported by numerous hardware and operating system platforms.
- Interoperability: CORBA uses Internet Inter-ORB Protocol (IIOP) for communication between various objects; hence, CORBA objects are fully interoperable.
- Modularity: CORBA objects communicate via interfaces and thus have all the advantages that come with interfaces.
- Security: CORBA provides security features such as encryption, authentication, and authorization to protect data and to control user access to objects and their services.

Extensions to the CORBA standard for a mobile environment have been proposed [18]. The proposal involves extending the IIOP to communicate with a mobile host.

1.6 Present Work

Many client-server applications exist in our traditional fixed network. A lot of work has been done in various areas related to client-server applications. Aspects such as security and standard interfaces for application development are a few related areas that have been studied.

Client-Server architectures also have a lot of applications in an ad hoc environment. These applications inherit many of the issues from their traditional fixed network counterpart; thus, many solutions developed in the case of fixed networks could be used in the case of ad hoc networks with minor modifications.

Since ad hoc networks are totally based on wireless links with dynamically changing topographies, applications developed in this environment have some unique issues. The limited bandwidth provided by wireless links is one issue which is compounded by the fact that the MNs (and hence the clients and server) are mobile. Because the MNs are mobile, there is not a unique position for the server for optimal usage of the limited bandwidth. Thus in our work, we allow the server to change its position dynamically as the location of the client MNs change. In this thesis we develop two algorithms for server movement with the goal of reducing the network load.

The problem of reducing the network load boils down to finding the appropriate server position based on the position of the clients and the expected amount of communication from each client. In Section 1.3, we presented two algorithms that use location information on the destination MN to find a route efficiently. Location information is assumed to be available from a GPS type of system. In our work, we assume that such a system is available and that any MN can obtain the location of any other MN in the network. In other words, we assume the positions of the clients and the server are known. We estimate the expected amount of communication from each of the clients by using the client's communication history.

The client's location information and its communication pattern could be used to construct a graph from which one can obtain the absolute optimum position for the server. The problem with a graph theory solution is three-fold.

1. In a dynamic environment, the absolute optimum position for the server could change rapidly. Thus, by the time the server reaches the "optimum" position, it may no longer be optimal.

2. Graph theory solutions are NP complete. Thus, as the number of MNs increase, the number of computations required to calculate the server's optimum position increases exponentially. Such computations prohibit the use of a graph theory based solution in practice.

3. A graph theory solution must be calculated at a single location. Such a centralized solution is not fault tolerant.

To avoid the above problems with a graph theory solution, we propose a distributed algorithm which moves the server one step at a time by estimating the savings obtained for the move. This approach is both

computationally inexpensive and also scalable compared to a graph theory approach.

There are many varied client-server applications in a MANET which could use our algorithm. In fact, distributed collaborative computing applications are expected to be very popular due to the notion of teams in a MANET.

- Chat Room Server: A chat room server implements the communication between a group of clients. A message sent by a client is echoed by the server to all the clients in the chat group. The responsibilities of the server involve registering a client, unregistering a client and sending each message received from a client to all the other clients in that particular chat group. The server may also offer private chat rooms; thus client authentication and other security related issues are also the server's responsibilities.

- Distributed Game Server: A distributed game server allows a group of people, working on different machines, to play a particular game with each other. The server's responsibilities involve deciding the winner, detecting and reporting illegal moves and echoing the move of a player to all the other players participating in the game.

- Distributed Databases: Each client in a distributed database has his/her own image of the database. Whenever a client needs to make a modification to the database the server receives the request and either approves or disapproves it. Once a change has been accepted by the server, the server must ensure that all the clients with an image of the database update their databases appropriately.

- Multi-Purpose High Computational Servers: An ad hoc environment may consist of MNs with varying computational power and resources. In such a heterogeneous environment an MN with more resources could act as a centralized server, providing various services to the MNs with fewer resources. For example, in a battlefield situation tanks would be resource rich and soldiers carrying hand-held devices would be resource poor. The clients of the server could also be involved in a related task which requires combining the data from various clients. For example, in the battlefield scenario, groups of soldiers at different locations may need to act together in a tactical attack according to some strategy calculated by the server.

In Chapter 2, we describe the two algorithms we developed for reducing network load in an ad hoc network using mobile servers. In Chapter 3, we discuss the details of our simulation. In Chapter 4, we present and discuss the simulation results. In Chapter 5, we present conclusions and our thoughts on future work.

Chapter 2

OUR ALGORITHMS

2.1 Introduction

In this chapter we propose two algorithms which decide when the server should move, based on a goal of reducing the network load. The first algorithm makes a detailed calculation of savings before selecting the best possible location for the server. For implementing this algorithm, the server needs the following information.

1. *Identification of the server's neighbors:* Many of the routing algorithms described in Section 1.3 keep track of neighboring MNs for routing purposes. Thus, such information should be readily available to the server.

2. *The location of any MN in the network:* To reduce routing overhead, location information obtained through GPS can be used in conjunction with a routing algorithm (see Section 1.3). Hence, requiring this information does not put any additional constraints or additional costs on our system.

3. *The number of hops a message takes from one of the neighboring MNs to one of the client MNs:* In a few of the routing algorithms (e.g., AODV and DSR), this information is available. Thus, we can assume this information exists.

4. *The number of messages that will be sent between the server and a particular client MN:* An exact value for this information is usually not available in a dynamic environment. Hence, the server in our algorithm estimates the future number of messages based on the number of messages sent by the client in the past.

The second algorithm we propose is computationally inexpensive and requires less information than algorithm one. The server position is calculated solely upon the average angle the client MNs make with the server MN and the estimated communication from each of the clients. The actual cost of communication (e.g., the number of hops) between the client MNs and the server is not considered. We also investigate the

use of the second algorithm as a *screening test* in the first algorithm, thus reducing the amount of calculation necessary in the first algorithm.

2.2 Terminology and Definitions

In this section we define terminology that will be used to describe the algorithms. As mentioned in Section 2.1, the server predicts the amount of future communication the client will have with the server based on the past messages sent by the client.

Definition: The length of time the server tracks the number of messages sent by the clients is referred to as *history*.

The server maintains the history of each of the clients via an array whose length is equal to the history value. Each cell in the array contains the number of messages received by the server from the client at the corresponding time instant, i.e., the k^{th}-cell contains the number of messages received from the client in the $(N-k+1)^{th}$ time instant, where N is the present time instant and $0 < k \leq N$. History is an important parameter of both the first and second algorithms. As an example, suppose the present time instant is 10, history is 5 and the server received $1,2,\ldots,10$ messages from a client in the $1^{st},2^{nd},\ldots,10^{th}$ instant, respectively. Thus at the present time instant, $10,9,8,7$ and 6 are stored in the history array. When time moves forward, the history array is updated by moving the values in the cells to the left by one cell causing the value in the left most cell, which contains the number of messages received from the client at $(\textit{new time} - \textit{history})$ time instant, to be removed from the array. The number of messages received at the new time instant is then placed at the top of the array.

Definition: The value of savings below which the server doesn't move is referred to as the *threshold value*.

Once the neighboring MN with the most estimated savings is identified in algorithm one, the server is moved to that MN only if the savings are more than a certain fixed value. Such a framework is necessary because the calculation is based on the most recently known location of MNs (which are on the move) and an *estimate* of the amount of communication between the server and each of the clients.

A threshold value is used to avoid having the server make an expensive move when the savings are minimal. In other words, the threshold value helps maintain stability in the implementation. A similar value is used in algorithm two.

Definition: The neighboring MN whose angle is the least deviation from the average client direction is determined and the server is moved to that location only if this angle is below a certain fixed angle. This fixed angle is referred to as the *threshold angle*.

In algorithm two, the average client angle represents the average direction the server is sending messages. It is calculated via the position of the clients.

Definition: The *average client angle* is defined as α_{av}:

$$\alpha_{av} = \frac{\sum_{i=1}^{n} \alpha_{C_iS}}{n} \tag{2.1}$$

where n is the number of clients and α_{C_iS} is the angle made by the i^{th} client C_i with the server. The angle α_{C_iS} is calculated with respect to a fixed reference direction. We assume the fixed reference direction is the direction of the positive x-axis.

Definition: Neighboring MNs which make an angle, α, such that the difference $(\alpha - \alpha_{av})$ is greater than a certain maximum value are not considered as a potential location for the server. This maximum angle is referred to as the screening angle.

Figure 2.1 illustrates the concept of screening angle and average client angle. The figure shows a server S with two neighboring MNs, $N1$ and $N2$, and the average client angle, $a1$, calculated via Definition 2.2. The screening angle is $a2$. Since $N1$ lies within the screening angle, it will be considered as a potential server location. $N2$, on the other hand, will be discarded as an unsuitable location.

2.3 Algorithm Outlines

In this section, we give a brief overview of the two algorithms. In Section 2.4, we present the algorithms in detail.

2.3.1 Algorithm 1

The following is an outline of the steps performed by the server to implement algorithm one:

- identify neighbors as potential server positions;
- obtain the location of each client;

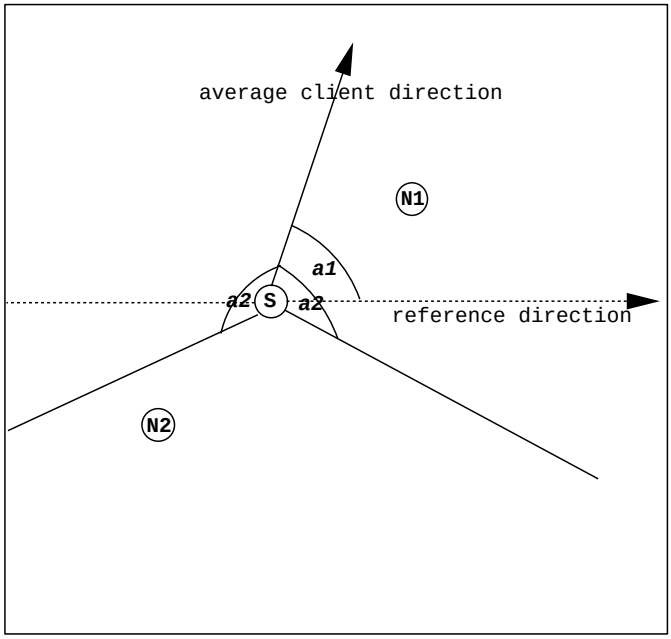

FIG. 2.1. Screening Angle and Average Client Angle Example

- based on the client's locations, eliminate unsuitable MNs from the list of potential server positions;
- based on the location of each client, the expected communication with each client, and the present location of the server, calculate the estimated savings if the server is moved to each MN in the potential server positions list;
- move to the neighboring MN with the highest savings over the threshold value.

2.3.2 Algorithm 2

The following is an outline of the steps performed by the server to implement algorithm two:

- identify neighbors as potential server positions;
- obtain the location of each client;
- estimate the average client angle based on the client's locations and expected communication with each client;
- for each of the neighboring MNs, calculate the angle they make with the reference direction and subtract the average client angle;

- identify the neighbor which makes the minimum angle and select it as the next server position, if the angle is below the threshold angle.

2.4 Algorithm Details

2.4.1 Algorithm 1

The following steps are completed by the server at each time instant.

1. Update the message history of each client:
   (a) Record the number of messages received from each client at the present time instant.
   (b) Move the values in the message history array forward one element (see Section 2.2).
   (c) Place the number of messages received at the present time instant in array element *h*, where *h* is the history value.
2. For each client, estimate the future number of messages that are expected between the server and the client using the client's history.
3. Calculate the average client angle α_{av}:
   (a) Find the location of each of the clients.
   (b) Calculate the angle of each client using a reference direction.
   (c) Calculate the average client angle via Definition 2.2 (see Section 2.2).
4. Calculate the list of MNs which are potential server positions:
   (a) Identify neighbor MNs and their locations.
   (b) Apply the screening test for each of the neighboring MNs:
      i. Calculate the angle each neighboring MN makes with the reference direction.
      ii. If the difference between this angle and the average client angle, α_{av}, is less than the screening angle, insert the MN into the list of potential server positions.
5. For each of the MNs in the list of potential server positions, calculate the estimated savings if the server is moved to the MN using the following equation:

$$\sum_{i=1}^{n}[C_{C_iS} - C_{C_iN}] - C_{SN} \tag{2.2}$$

where C_{C_iS} is the cost of communicating with the i^{th} client when the server is at the present location, C_{C_iN} is the cost of communicating with the i^{th} client when the server is at the neighboring MN in consideration, and C_{SN} is the cost of moving the server from the present location to the neighboring MN in consideration. Recall that the cost of communication between two MNs is the number of hops between the two MNs. Hence, the savings is in terms of $message - hops$.

6. Identify the MN with the highest savings calculated in step 5. Move the server to this neighboring MN provided the savings is above the threshold value (see Definition 2.2 in Section 2.2).

2.4.2 Algorithm 2

The following steps are completed by the server at each time instant.

1. Update the message history of each client:
   (a) Record the number of messages received from each client at the present time instant.
   (b) Move the values in the message history array forward one element (see Section 2.2).
   (c) Place the number of messages received at the present time instant in array element h, where h is the history value.
2. For each client, estimate the future number of messages that are expected between the server and the client using the client's history.
3. Calculate the average client angle α_{av}:
   (a) Find the location of each of the clients.
   (b) Calculate the angle of each client using a reference direction.
   (c) Calculate the average client angle via Definition 2.2 (see Section 2.2).
4. Identify neighbor MNs and their locations.
5. For each neighboring MN, calculate the difference between the angle they make with the reference direction and the average client angle, α_{av}.

6. Identify the MN which has the minimum difference calculated in step 5. Move the server to this neighboring MN provided the difference is below the threshold angle (see Definition 2.2 in Section 2.2).

Chapter 3

SIMULATION

3.1 Introduction

To illustrate our algorithms and to evaluate their performance with respect to various parameters, we have implemented a chat room server application in a MANET environment. The application consists of a server MN, a set of client MNs and a set of MNs not involved in the application. Recall from Section 1.6, when a chat room server receives a message from a client, it echoes the message to all the clients. The MNs not participating in the application together with the server and the client MNs form an ad hoc network. In our simulation, the MNs are initially placed in a 2-d grid and move according to a mobility model (see Section 3.3). At each clock tick, a potential server position is calculated and the server is moved accordingly. The number of hops between any two MNs is taken as the minimum number of hops between the MNs. The communication distance between two neighboring MNs consists of two parts: a constant part and a variable part. The variable part is chosen randomly from a uniform distribution (see Section 3.4.1 for details). Every client's message history is stored over a fixed length of time.

3.1.1 Calculation of Number of Hops

In our simulation, the cost of communication between two MNs is taken as the minimum number of hops between the two MNs. We use the following algorithm to discover the minimum number of hops between any two MNs in our simulation.

1. Initialize number of hops to zero.
2. If origin MN is the same as the destination MN, return number of hops.
3. Otherwise:
   (a) increment the number of hops by one;
   (b) identify MNs that are number of hops from the origin MN.
4. If the set of MNs identified contains the destination MN, return number of hops.

5. Otherwise, repeat steps 3 and 4.

3.1.2 Cost Estimation

To implement algorithm one we need the cost of sending a message from a client to the server. We base this cost on the number of hops the message traverses. Since we are implementing a chat room server, each message sent by a client to the server is echoed to all the clients. (Note that the message is echoed by the server to all the clients including the client that transmitted the message.) Hence, the cost of sending a message from the i^{th} client to the server is:

$$C_{iS} = N_{C_iS} + \sum_{k=1}^{N} N_{C_kS} \tag{3.1}$$

where C_{iS} is the cost of sending a message from the i^{th} client in the chat room application, N_{C_iS} is the number of hops from the i^{th} client to the server, and N is the total number of clients.

3.2 Parameters

The length of time the server tracks the number of messages sent by each client is one of the parameters of the simulation. In general, different applications should have different history values. For example, the history could be based upon the consistency with which the clients send messages. If the clients are sending messages at a fixed rate over a long period of time, using a large history would yield better performance. On the other hand, when the clients are changing their behavior rapidly, a low history value should be used.

The threshold value can be viewed as a safety zone. The server movement is based on *estimated savings*, which is calculated with the previously known locations of the MNs and the number of messages we expect each client to send. To ensure that a server movement does not lead to an *increase* in network load, due to subsequent MN movement or changes in client's behavior, algorithm one uses a threshold value of estimated savings below which the server doesn't move. In an environment where the MNs are fixed and the clients send messages at a fixed rate, the server should move for any positive estimated savings. In an ad hoc network, the threshold value should be based on the validity of our estimated savings. One of the environmental variables which affects the threshold value is the number of clients. For example, consider the following two scenarios:

1. A server with one client which sends two messages per second.

2. A server with two clients, each of which sends one message per second.

In the first scenario, the deviation in estimated savings from the actual savings is only due to the movement of a single client. In the second scenario, the deviation is due to the movement of two clients. Thus, we expect a higher deviation between the actual savings and the estimated savings in the second scenario, and hence a higher threshold value should be used. In general, as the number of MNs increases the deviation of the estimated savings from the actual savings increases. Thus, the threshold value should be proportional to the number of clients.

The screening angle is another parameter that can be used to tune our first algorithm. As the screening angle increases, the number of neighboring MNs that will be considered as potential locations increases. If the screening angle is 180-degrees, then all of the neighboring MNs are considered as potential server locations. A larger screening angle gives the server more options for movement and hence may lead to more savings. On the other hand, a smaller screening angle reduces the amount of calculations required in algorithm one and hence might be a better choice.

3.3 Mobility Model

The mobility model used in the simulation is based on the Random Gauss Model. The model is designed to adapt to different levels of randomness via one tuning parameter. The value at the n^{th} instance, v_n, is calculated based upon the value at the $(n-1)^{th}$ instance, v_{n-1}, and a random variable, x_{n-1}, using the following equation:

$$v_n = a v_{n-1} + (1-a)\mu + \sqrt{(1-a^2)}x_{n-1} \tag{3.2}$$

where a is the tuning parameter for setting the extent of randomness, μ is a constant, and x_{n-1} is a random variable selected from a gaussian distribution. For $a = 0$ the equation yields totally random values, equivalent to Brownian motion. For $a = 1$ the equation yields fixed values, equivalent to linear motion. The value of a can be adjusted between these two extremities to obtain different levels of random movement.

Initially each MN in our simulation is assigned a current speed and direction. At fixed intervals of time, movement occurs by updating the speed and direction of each MN. In our simulation, we update the speed and direction of each MN every 20 time intervals based on the following two equations. Specifically, at time interval n, the speed and direction of each MN are given by the equations:

$$s_n = as_{n-1} + (1-a)\mu + \sqrt{(1-a^2)}s_{x_{n-1}} \tag{3.3}$$

$$\alpha_n = a\alpha_{n-1} + (1-a)\mu + \sqrt{(1-a^2)}\alpha_{x_{n-1}} \tag{3.4}$$

where s_n and α_n are the new speed and direction of the MN at time interval n. In our simulation, the parameter a is chosen from a uniform distribution in the range $0-1$, and $s_{x_{n-1}}$ and $\alpha_{x_{n-1}}$ are chosen from a random gaussian distribution with *mean* $= 0$ and *standard deviation* $= 1$. The value of μ is fixed at 1.0.

At each time interval the next location is calculated based on the current location, speed, and direction of movement. Specifically, at time interval n, an MN's position is given by the equations:

$$x_n = x_{n-1} + s_{n-1}\cos\alpha_{n-1} \tag{3.5}$$

$$y_n = y_{n-1} + s_{n-1}\sin\alpha_{n-1} \tag{3.6}$$

where (x_n, y_n) and (x_{n-1}, y_{n-1}) are the x and y coordinates of the MN's position at the n^{th} and $(n-1)^{th}$ time intervals, respectively, and s_{n-1} and α_{n-1} are the speed and direction of the MN, respectively, at the $(n-1)^{th}$ time interval.

To ensure that an MN does not remain near an edge of the grid for a long period of time, the MNs are forced away from an edge when they move within a certain distance of the edge. For example, when an MN is near the right edge of the simulation grid, the value $\alpha_{x_{n-1}}$ is chosen from a random gaussian process whose mean is π. Thus, the MN's new direction is away from the right edge of the simulation grid. The values of the mean for different locations in the simulation grid are shown in Figure 3.1.

3.4 Simulation Details

3.4.1 Communication Distance

Our simulation initially places the MNs on a 300×300-grid. To simulate a realistic communication model, we use a variable communication distance between any two MNs. Specifically, the communication distance between two MNs is $d = 65 + d_x$, where d_x is picked from a random uniform distribution in the range $0-5$. Thus, the maximum communication distance is 70 and the minimum communication distance is

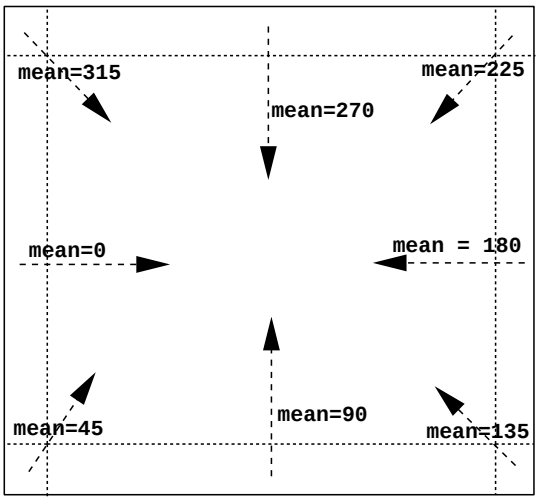

FIG. 3.1. Change of Mean Angle Near the Edges

65. Each of our simulation trials is with 10 clients, one server, and a variable number of MNs not involved in the client-server application. In other words, the total number of MNs in each simulation trial is equal to the number of clients plus the server MN plus the number of MNs not involved in the application. For example, in a simulation trial with 40 MNs not involved in the application, we have a total of 51 MNs in the ad hoc network.

3.4.2 Estimation of Savings

Each trial of the simulation executes for a total of 100 time intervals. During the simulation, each of the clients send messages at a constant rate; however, any two clients may send messages at different rates. The rate at which a client sends a message is chosen from a uniform distribution in the range $1-10$. Specifically, a client sending messages at rate n would send one message every n time intervals. Thus, each client will transmit between 10 and 100 messages during the simulation. The estimated savings for a particular simulation trial is calculated from the cost of sending the messages with the server at the dynamic locations, the cost of sending the messages with the server at the initial location, and the total number of movements the server makes during the course of the simulation trial. Specifically,

$$S = \sum_{i=1}^{N} [C^i - C_I^i] - N_H \tag{3.7}$$

where S is the total estimated savings, N is the total number of messages received by the server from all

the clients, C^i is the cost of sending the i^{th} message with the server at its current location, C_I^i is the cost of sending the i^{th} message with the server at its initial location, and N_H is the total number of movements the server makes during the simulation trial. In estimating the cost of the server movement, we assume that the server exists, in a passive state, on all the MNs in the ad hoc network; thus, moving the server to another MN is equivalent to activating the server on that MN by sending a single message. Thus, the cost of server movement is equal to the number of server movements.

To illustrate that our algorithms for server movement are possible, we placed the MNs on different machines and used CORBA interfaces for communication. Thus, the developed server is a mobile agent and moves based on one of our algorithms from one machine to the other. For our performance investigation, however, we only used a single machine. Due to the uncertain delays involved in sending messages using TCP, we could not replicate our performance results unless we controlled the communication delays in our simulation.

Chapter 4

SIMULATION RESULTS

4.1 Introduction

In this chapter, we describe the results from simulating the chat room server described in Section 3.1. We have identified the following parameters that affect the performance of algorithm one:

- threshold value: the value of savings below which the server doesn't move;
- history: the length of time the server tracks the number of messages sent by the clients;
- number of MNs: the total number of MNs includes the client MNs, the server MN, and the MNs that don't participate in the application;
- screening angle: the angle used to eliminate some obviously unsuitable neighbors from the list of potential server locations (see Definition 2.2 in Section 2.2).

The only parameter that affects algorithm two is the threshold angle, i.e., the maximum angle an MN can make with the average client direction in order for the server to move to that MN (see Definition 2.2 in Section 2.2). In Section 4.2, we evaluate the performance of algorithm one. In Section 4.3, we evaluate the performance of algorithm two. Lastly, in Section 4.4 we summarize the results of both algorithms, and we discuss their relative advantages and disadvantages.

4.2 Algorithm 1: Performance Evaluation

4.2.1 Effect of Threshold and History

The effect of the threshold value on the performance of the algorithm is evaluated by plotting Savings vs. Threshold (Figure 4.1) and Number of Server Movements vs. Threshold (Figure 4.2). We also vary history in order to evaluate the effect of history on the algorithm. Specifically, we evaluate the following parameter values in Figures 4.1 and 4.2:

- threshold value: 25 to 800 in steps of 25;

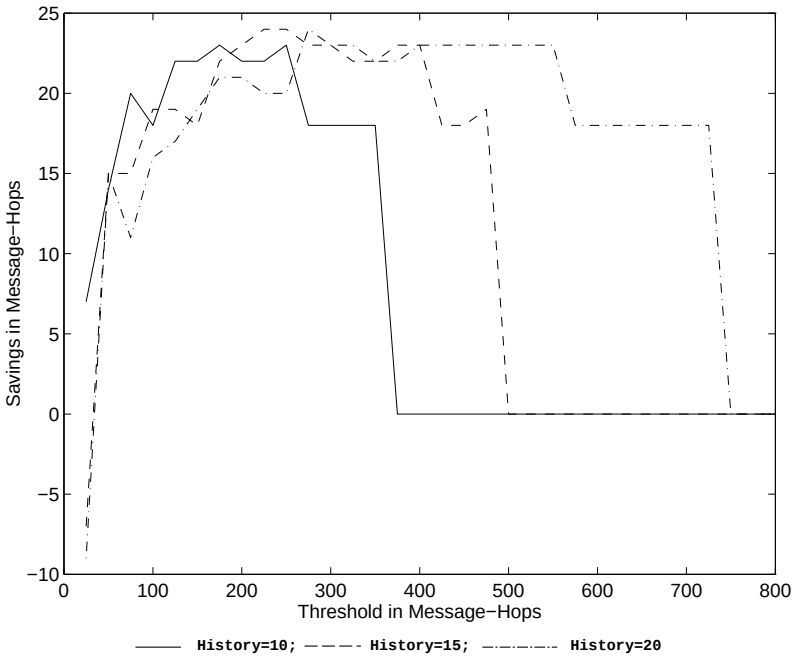

FIG. 4.1. Savings vs. Threshold at Different History Values

- history: 10, 15, 20;
- number of MNs: 51 (10 clients + 40 MNs not involved in application + 1 server);
- screening angle: 180 degrees.

Figure 4.1 plots Savings vs. Threshold at three different history values. In each history case, there is initially little savings. In fact, there is negative savings when history is equal to 15 or 20 and threshold is equal to 25 or 50. Positive savings begin, however, when the threshold value becomes larger than 50. All three history curves reach a maximum savings point; increases in the threshold value past this maximum savings point eventually leads to a decrease in savings. Each of the history curves exhibit a "stable" region around the maximum savings point, i.e., the change in savings is not significant for a range of threshold values. A low threshold value does not account for the subsequent movement of the MNs, and hence the savings obtained is low. Similarly, at a high threshold value the algorithm becomes too conservative, restricting the server movement and hence reducing the savings obtained; eventually the savings becomes zero as the threshold increases.

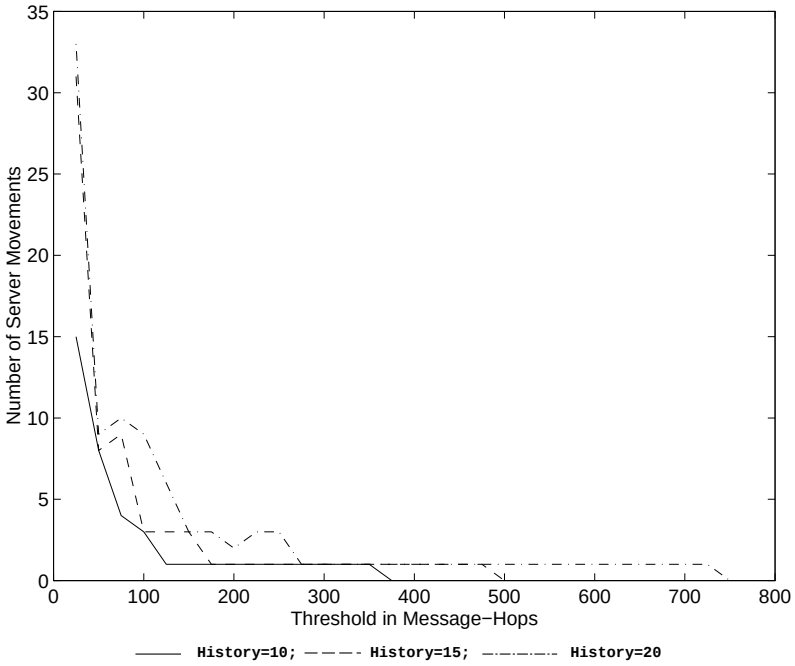

FIG. 4.2. Number of Server Movements vs. Threshold at Different History Values

As history is increased, the curves "move" to the right and become wider. Specifically the following trends are observed:

1. *The curve with a larger history reaches the same maximum savings as the curve with a smaller history, but the maximum savings point is reached at a larger threshold value.* The server predicts that each client will send as many messages as has been stored in its history. Since, with a larger history, the server keeps track of messages over a larger length of time, it predicts more messages will be sent from each client and hence more savings will exist if the server is moved. The threshold value is a minimum bound required on estimated savings to move the server. Thus, for the same system, the same performance (i.e., savings) is obtained with a small history and small threshold value as with a large history and large threshold value. In other words, the curves with a larger history reach the maximum savings point when the threshold value is larger.

2. *The region over which the savings is stable is wider for a larger history.* Since the server estimates more savings with a larger history, a larger threshold value is required to change the performance of the algorithm, i.e., to reduce the amount of savings. Thus, the region over which the savings is stable

becomes wider with an increase in history.

Figure 4.2 shows the total number of movements the server makes in the simulation as the threshold value increases. Each of the history curves show a decrease in the number of server movements with an increasing threshold value. The point where the number of server movements becomes zero is the same as the point where the savings in Figure 4.1 becomes zero (e.g., threshold equal to 375 for history equal to 10). Furthermore, the stable savings regions in Figure 4.1 correspond to the regions where the number of server movements is stable in Figure 4.2.

As history increases, the curves take longer to reach the stable regions in Figure 4.2. This fact can be compared to Figure 4.1, where it takes a larger threshold value for a larger history to reach the stable savings region. Similarly, the larger history curves have a larger stable region in Figure 4.2, which can be compared to the corresponding regions in Figure 4.1.

Conclusions: At a fixed history, there is a range of threshold values around which the savings are stable. Any fluctuations in the environment in this stable range has little effect on the algorithm's performance. Since using a larger history offers a wider stable region, a large history is preferred; however a large history is only feasible when the clients send messages at a semi-constant rate (see Section 3.2).

4.2.2 Effect of Number of MNs

In this section we present the results obtained from our simulation as the number of MNs increase. We vary the number of MNs by varying the number of MNs not involved in the application. The number of client MNs is fixed at 10. We evaluate the following parameter values in Figures 4.3 and 4.4:

- threshold value: 25 to 500 in steps of 25;
- history: 10;
- number of MNs: 31, 51, 71, 91;
- screening angle: 180 degrees.

Figure 4.3 plots Savings vs. Threshold as the number of MNs increase. The nature of the curves in Figure 4.3 is similar to the nature of the curves in Figure 4.1: each curve begins with a small savings, increases to a maximum savings point and then decreases to zero.

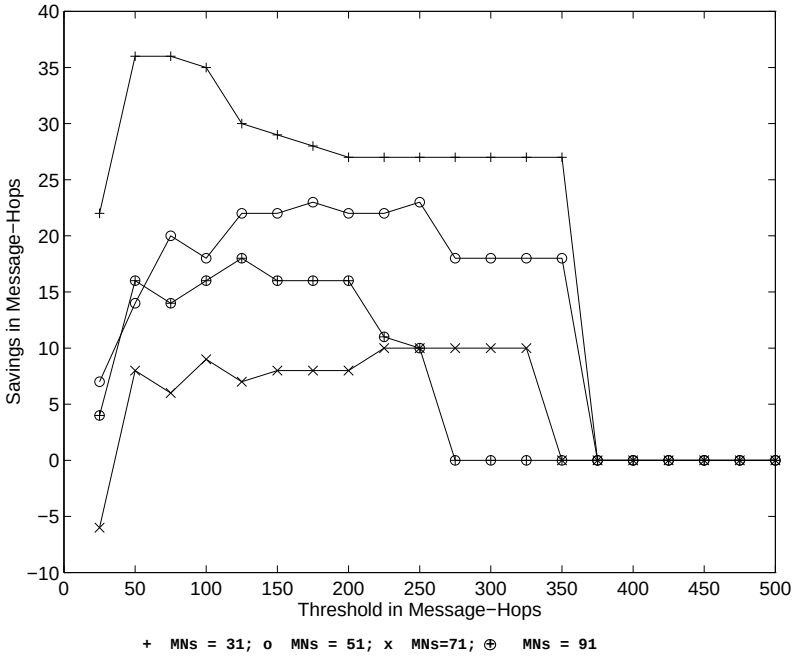

FIG. 4.3. Savings vs. Threshold at Different Number of MNs

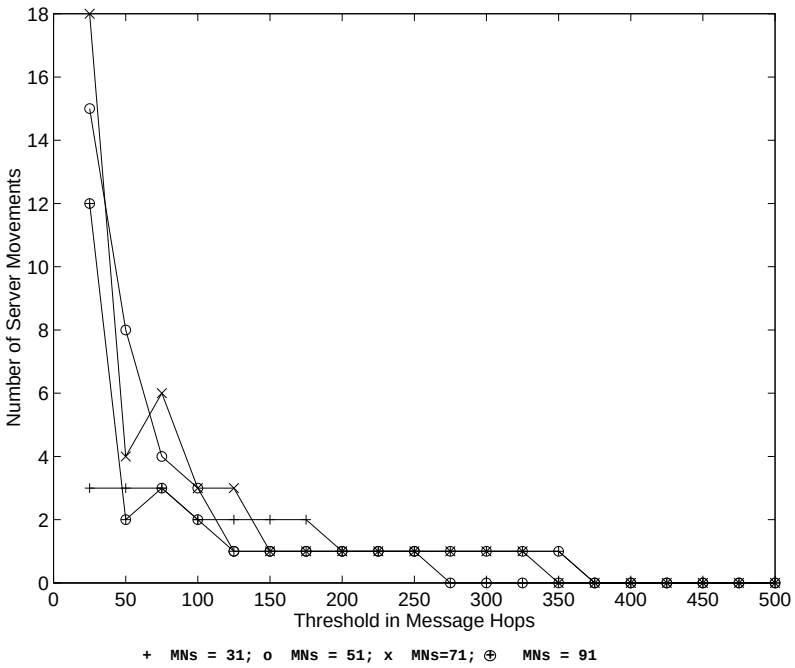

FIG. 4.4. Number of Server Movements vs. Threshold at Different Number of MNs

As the number of MNs increase, the savings sometimes decrease and sometimes increase. Specifically, the savings in the curves decrease from 31 MNs to 71 MNs, decrease from 71 MNs to 91 MNs when threshold is greater than 250, and increase from 71 MNs to 91 MNs when the threshold is less than 250. We understand this trend exists due to the following reason. As the number of MNs increase, two conflicting trends occur. On one hand, an increase in the number of MNs leads to a decrease in average number of hops for communication and hence to lower estimated savings. On the other hand, with an increase in the number of MNs, a server has more options for movement (i.e., more neighbors for potential server movement) and hence may find a better position and increase savings. The rate of decrease in the average number of hops corresponds to the rate of increase in the number of MNs, i.e., the difference between the average number of hops for 51 MNs and 71 MNs is greater than the difference between the average number of hops for 71 MNs and 91 MNs. Initially this decrease in the average number of hops prevails over the possibility of a better position. However, as the number of MNs increase, the effect diminishes and an increase in savings due to a better position outweigh the decrease in savings due to a decrease in the average number of hops.

Figure 4.4 plots Number of Server Movements vs. Threshold, illustrating the effect of the number of MNs on the number of server movements. The nature of the curves in Figure 4.4 and their relationships to the corresponding curves in Figure 4.3 is the same as in the case of Figure 4.2 and Figure 4.1. Specifically, the number of server movements for a low threshold is initially very large, but decreases rapidly to zero as the threshold value is increased. Similar to Figures 4.1 and 4.2, stable regions in Figure 4.4 correspond to stable regions in Figure 4.3.

As the number of MNs increase, the server has more options for movement and, hence, may move a larger number of times. Conversely, an increase in the number of MNs means a decrease in the estimated savings and, hence, a decrease in the number of movements the server makes. Thus, there is no obvious trend in the number of server movements as the number of MNs and the threshold value increase.

Conclusions: Since the number of MNs is an environmental variable it cannot be controlled. However, these results offer an estimate of the amount of savings based on the number of MNs. One can use such estimates in setting the frequency at which the algorithm is performed. In other words, when one does not expect a lot of savings due to a large number of MNs involved in the application, the algorithm can be performed less frequently. On the other hand, when one expects a lot of savings, a more frequent calculation of algorithm one might prove beneficial.

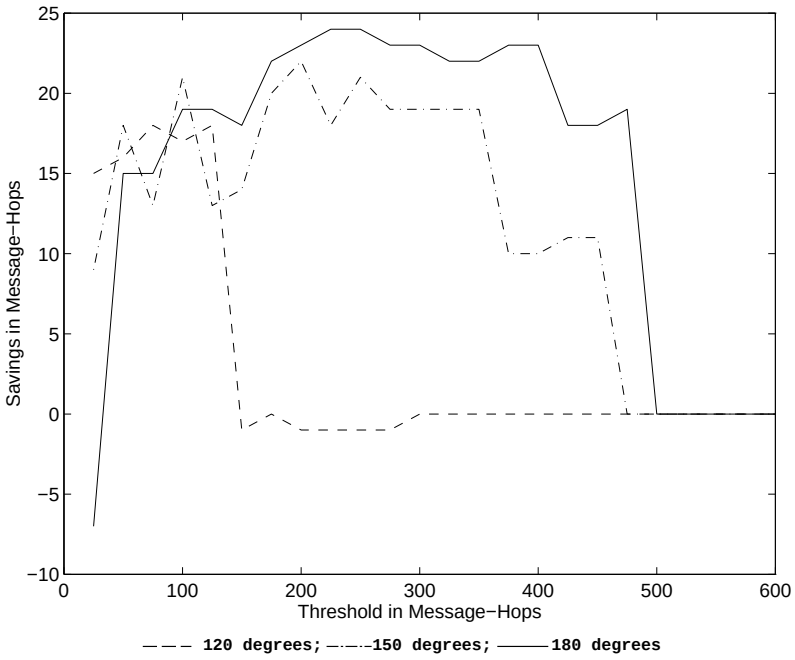

FIG. 4.5. Savings vs. Threshold at Different Screening Angles

4.2.3 Effect of Screening Angle

In this section we present the results obtained from our simulation when we vary the screening angle. We evaluate the following parameter values in Figures 4.5, 4.6 and 4.7:

- threshold value: 25 to 600 in steps of 25;
- history: 15;
- number of MNs: 51;
- screening angle: 120, 150, 180 degrees.

A 180-degree screening angle is equivalent to considering all the neighboring MNs as potential server locations (see Figure 2.1 in Section 2.2). The results presented in Sections 4.2.1 and 4.2.2 are with a 180-degree screening angle and hence consider all the neighboring MNs as potential server locations. Figure 4.5 plots Savings vs. Threshold. The curves exhibit the same nature as the corresponding curves in Figures 4.1 and 4.3. We analyze the reasons for the illustrated behavior in Section 4.2.1.

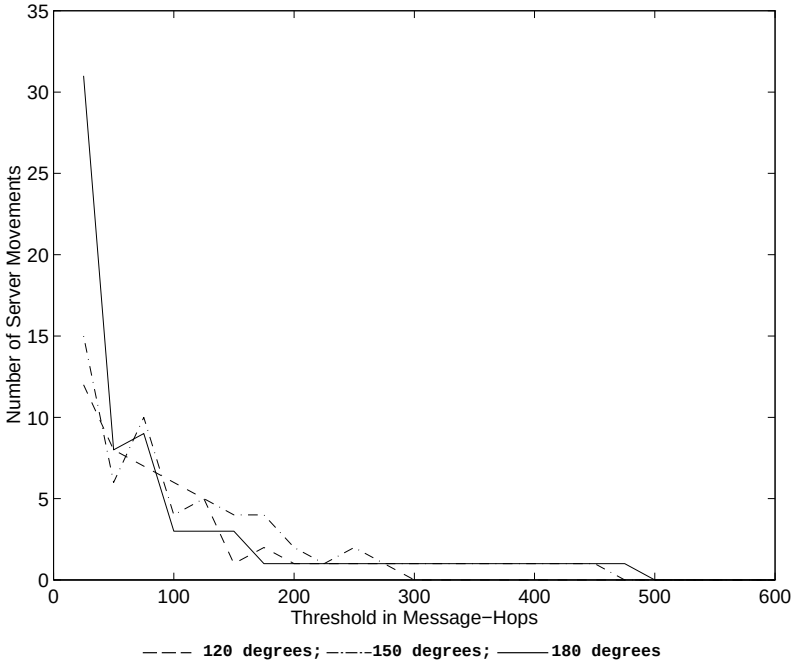

FIG. 4.6. Number of Server Movements vs. Threshold at Different Screening Angles

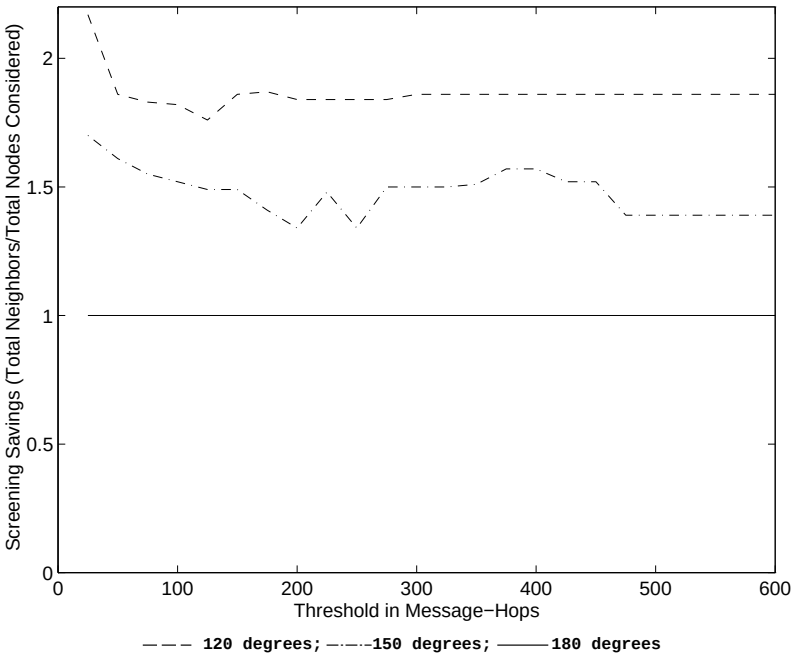

FIG. 4.7. Screening Savings vs. Threshold at Different Screening Angles

The effect of the screening angle on savings varies with the threshold value. At low threshold values (e.g., 25), a lower screening angle yields larger savings. On the other hand, at high threshold values (e.g., 200 to 400), a higher screening angle yields larger savings. We believe the reasons for the behavior are the following. At low threshold values, algorithm one, without the screening test, may move the server to an MN which offers low estimated savings. When the screening test is included with a small screening angle, the direction in which the server can move is restricted, which restricts excessive server movement and improves the savings. At high thresholds, server movement is overly restricted when a small screening angle exists and thus less savings is obtained with a small screening angle.

Figure 4.6 shows the effect of the screening angle on the number of server movements. At low threshold values, the number of server movements increases with increasing screening angle; at high threshold values, the number of server movements decreases with increasing screening angle. We attribute such behavior to the following reasons. At small screening angles, the number of neighbors considered as potential server locations is small; hence, at low threshold values, the number of server movements increases as the screening angle increases. Furthermore, a small screening angle may move the server to a neighboring MN whose savings is not the maximum. Since there exists an MN with more savings than the new location of the server, the server may move again thus leading to a larger number of server movements.

Figure 4.7 plots Screening Savings vs. Threshold at various screening angles. The screening savings represents the amount of computation reduction due to reducing the neighboring MNs considered as potential locations for server movement. The amount of screening savings is calculated by dividing the number of neighbors by the number of MNs considered after the screening test is applied for each time instance.

At 180-degree screening angle, i.e., all the neighbors are considered as potential locations for server movement, the screening savings is equal to one. At smaller screening angles, some of the neighboring MNs are rejected during screening and hence the screening savings is greater than one. Reducing the screening angle reduces the number of MNs considered and hence increases the screening savings.

Conclusions: A small screening angle is desired in order to reduce the amount of computations involved. When the number of MNs is large, a small screening angle could lead to substantial savings. Furthermore, a small screening angle is desired when accurate information on the environment variables, such as the client’s movement/calling pattern and the number of clients, is not available.

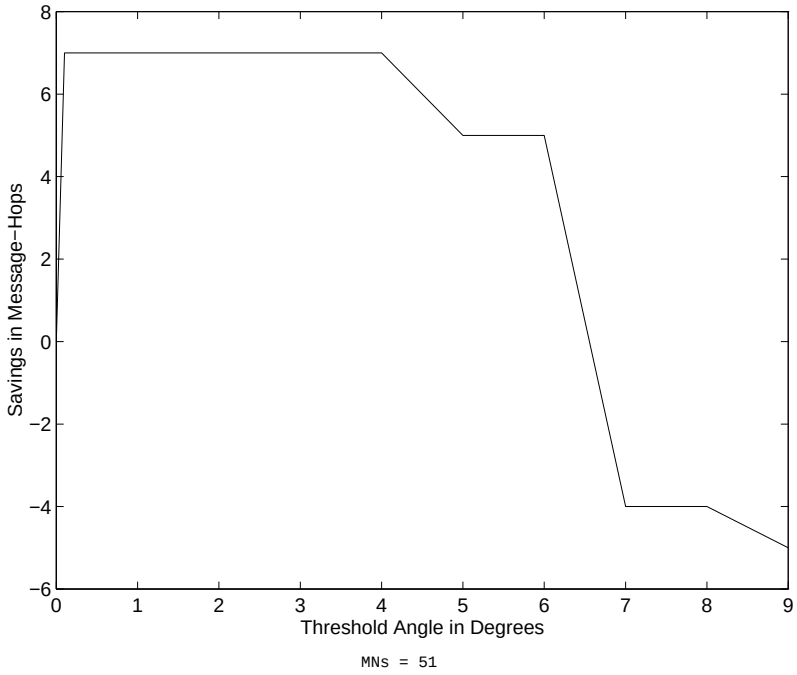

FIG. 4.8. Savings vs. Threshold Angle

4.3 Algorithm 2: Performance Evaluation

In this section we present the performance of algorithm two which is described in Section 2.4.2. This algorithm involves fewer calculations compared to algorithm one. Figure 4.8 plots Savings vs. Threshold Angle for 51 MNs. The plot shows positive savings at very low threshold angles. As the threshold angle increases, there is initially a stable region for savings and then the savings decreases rapidly. A large threshold means the server may move to an MN which is not in the general direction of the clients, thus a decrease in savings occurs. In other words, a large threshold angle means the server may move away from the clients; instead, the server should remain where it is to maintain the savings accumulated.

Figure 4.9 plots the number of server movements at various threshold angles when 51 MNs are in the system. The figure illustrates that the number of server movements increases with increasing threshold angle. As the threshold angle increases, there are more possible neighboring MNs where the server can move; thus the server moves a larger number of times. In fact, the server moves whenever there is at least one neighboring MN that satisfies the threshold angle condition.

Conclusions: Algorithm two is a lightweight algorithm and could be used on MNs that are resource

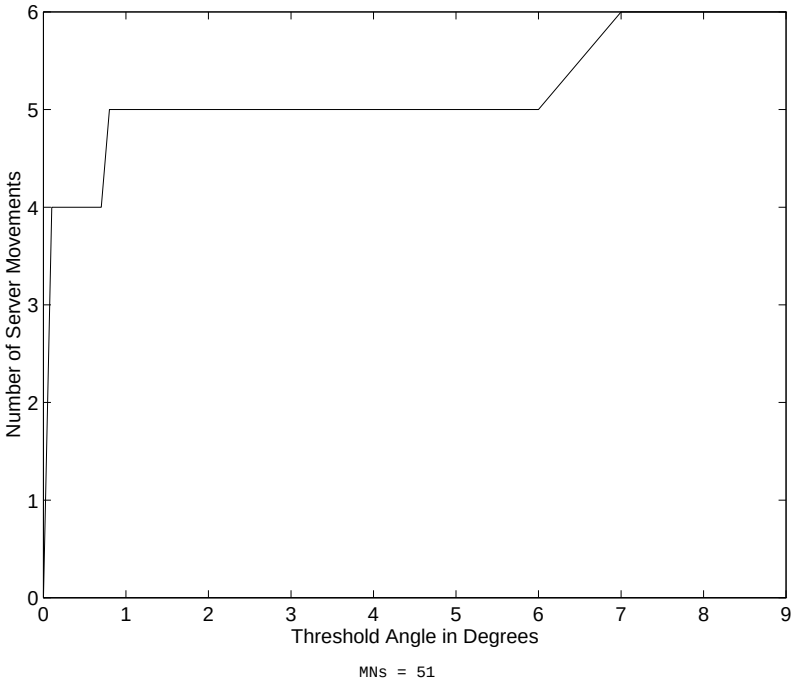

FIG. 4.9. Number of Server Movements vs. Threshold Angle

poor. In addition, algorithm two could be used in cases where the routing information, e.g., number of hops from one MN to another, is not freely available. Figures 4.8 and 4.9 illustrate that savings in network load can be obtained with this inexpensive algorithm. Since the algorithm does not consider the cost of communication between MNs, the change in the communication cost due to a change in the number of MNs will not have any particular effect on the performance of the algorithm.

4.4 Conclusions

Both algorithm one and algorithm two show substantial positive savings. The parameters of the algorithms could be fixed based on results that optimize their performance. The savings in algorithm one is considerably higher than the savings in algorithm two; however, algorithm one is computationally more complex and requires more information than algorithm two. Using the screening test in algorithm one, we reduce the amount of computation required, at the cost of a reduction in savings.

Chapter 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

With improvements in technology and a reduction in costs, mobile computing is rapidly becoming popular. Various Internet based applications are widely being used and with an increasing popularity of mobile computing, the demand for such applications on mobile devices is expected to grow.

A system consisting of a collection of autonomous systems communicating through wireless links is called a MANET. Recently, routing algorithms for a MANET which use location information on the MNs, obtained from GPS, have been proposed. The use of GPS offers a reduction in routing overhead, which is worthwhile since MANET suffers from a narrow bandwidth due to the wireless medium.

In order to reduce the network load in MANET based client-server applications, we proposed two algorithms that determine when to move a server process. The first algorithm estimates the savings possible when the server is moved to a particular neighboring MN. The server is moved to the neighboring MN with the highest estimated savings. The savings is calculated via the current location of the client MNs and an estimate on the number of messages that will be sent by the clients in the future. A screening test, which reduces the number of computations in algorithm one, is also proposed. The second algorithm is a computationally inexpensive algorithm which bases the decision to move purely on the location of the MNs.

To illustrate our algorithms and to evaluate them with respect to various parameters, we simulated a chat room server application. The performance of the algorithms is evaluated on the basis of savings in network load and the number of server movements. The following conclusions were drawn from our performance evaluation of algorithm one:

- a large history value should be used whenever the clients show a consistent behavior;
- a small screening angle should be used when the MNs are resource poor in order to reduce the number of computations;
- the frequency at which the algorithm is performed should be based on the number of MNs in the network.

Algorithm two has lower savings than algorithm one and should be used only when sufficient resources or information is not available.

The complexity of our algorithms is proportional to the average number of neighboring nodes an MN has in the network. The average number of neighboring nodes, for a fixed area, is proportional to the square root of the number of nodes. Thus, the complexity of our algorithms is of order $\sqrt{n}$, where n is the number of nodes.

In algorithm two we use the average client angle to estimate the direction of the client's location. When the clients are uniformly distributed around the server, however, the average client angle is not a good estimate for the direction of the client's locations. Hence, algorithm two is only valid when the clients are located in a particular direction with respect to the server.

5.2 Future Work

In this section we discuss possible extensions to our two algorithms. We qualitatively assess such modifications and mention the possible advantages and disadvantages they might offer over algorithm one and algorithm two.

5.2.1 Multiple Servers

Both algorithms yield substantial savings when the server receives a large fraction of the total messages from a particular direction. In a scenario where the clients are localized at two different areas in the MANET, which are far away from each other, our algorithms may not yield much savings. In such a scenario, both of our algorithms would move the server to an MN which is located in between the two areas where the clients are located; however, since both the regions are far from the server, substantial savings cannot be realized. In such a scenario, providing a server for each of the localized areas should improve savings. Within the localized areas, our algorithms could be used to further reduce the network load.

Implementing multiple servers would require an algorithm for identification of the localized areas. Similarly, an algorithm for "merging" two localized areas would also be required. We note, however, that such a solution may not apply to all client-server applications. For example, ensuring consistency in a distributed database would be more complex if multiple servers are allowed.

Figure 5.1 depicts a chat room server application with two servers $S1$ and $S2$. Server $S1$ has three clients, $C1$, $C2$, and $C3$ and server $S2$ has three clients, $C4$, $C5$, and $C6$. When client $C6$ sends a message,

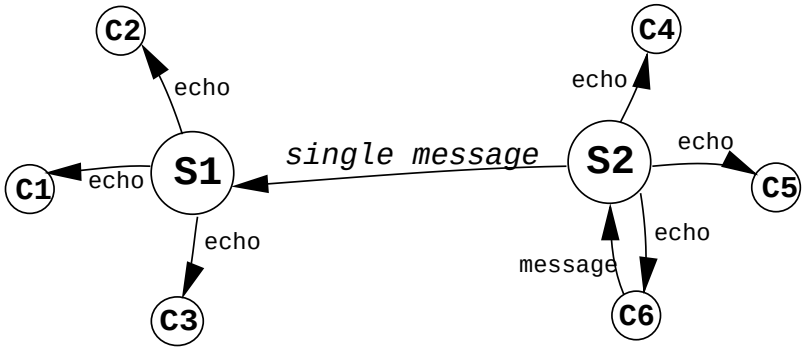

FIG. 5.1. Multiple Chat Room Servers

server *S*2 echoes the message to its clients, *C*4, *C*5, and *C*6, and sends a single message to the server *S*1. Server *S*1 then echoes the message to its clients, *C*1, *C*2, and *C*3. Since only a single message is sent from *S*2 to *S*1, substantial reduction in network load over a single server system would be achieved.

5.2.2 Variable Parameters

In the simulation of our two algorithms, we have used fixed values for all the parameters, i.e., threshold value, history and screening angle. It may be possible to improve the performance of our algorithms, by varying the parameter values during run time, at the cost of additional calculations.

In Section 3.1, we stated that history should be chosen based on the client's consistency in sending messages. In some applications, a subset of the clients may be more consistent than others. Furthermore, one client may be consistent for a period of time and then inconsistent at another time. In such a situation, the server can dynamically assign an appropriate value for history based on the client's current behavior.

Similarly, instead of having a fixed screening angle, the server could vary the screening angle based on the location of the client MNs. As the MNs move, all client MNs may be in a particular direction with respect to the server MN. In this situation, using a small screening angle could lead to a reduction in computation with a minor reduction in savings. On the other hand, when the client MNs are spread around the server MN, a large screening angle could be used. In other words, the server can choose an appropriate screening angle based on the current location of the client MNs.

The effect of the number of clients on the threshold value has been discussed in Section 3.1. In an environment where there is a wide variation in the number of clients, the server can dynamically change the threshold value based on the current number of clients.

5.2.3 Using Node Movement Information

In addition to the current location, GPS provides the velocity and direction of movement of the MNs (see Section 1.3). LAR algorithms use this information in routing by identifying an expected zone based on the location and velocity of the MN. A similar improvement could be made in our algorithms. Based on the movement of the client MNs, the future location of the MNs could be estimated. These estimated locations could then be used in making a better estimate of the savings in algorithm one.

5.2.4 Realistic Mobility Model

The mobility model (see Section 3.3) used in our simulation is not based on realistic situations. Since research in ad hoc networks is just beginning, movement data for the MNs from which a realistic model can be formulated, is not available. A realistic movement model would help in relating our results to real world situations and may also help in tuning our algorithms for better performance.

5.2.5 Statistical Techniques

In our algorithms we use history to predict future client behavior. We could use statistical techniques on past client behavior to improve our prediction of the future. Similarly, an approximation for the threshold value could be obtained using statistical techniques. Since the threshold value is a measure of the error in estimated savings, a methodology for estimating the error would help in determining an appropriate threshold value. We plan to consider these issues in future work.

REFERENCES

[1] C.-C. Chiang, M. Gerla, and L. Zhang. Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel. *Proceedings of IEEE SICON '97*, pages 197–211, April 1997.

[2] C. E. Perkins. IP Mobility Support. *Request for Comments 2002*, October 1996.

[3] C. E. Perkins. Mobile IP. *IEEE Communications Magazine*, 35(5):84–99, 1997.

[4] C. E. Perkins and D. Johnson. Route Optimization in Mobile IP. *Internet Draft, draft-ietf-mobileip-optim-08.txt*, February 1999.

[5] C. E. Perkins and E. M. Royer. Ad Hoc On-Demand Distance Vector Routing. *Proceedings of 2nd IEEE Workshop on Mobile Computing Systems and Applications*, February 1999.

[6] C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. *Computer Communications Review*, 24(4):234–244, October 1994.

[7] C-K. Toh. A Novel Distributed Routing Protocol to Support Ad-Hoc Mobile Computing. *Proceedings of the 1996 IEEE Fifteenth Annual International Phoenix Conference on Computers and Communications*, pages 480–486, March 1996.

[8] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *Mobile Computing, ed. T. Imielinski and H. Korth, Kluwer Academic Publishers*, pages 153–181, 1996.

[9] D. Chess, C. Harrison, and A. Kershenbaum. Mobile Agents: Are They a Good Idea? Technical report, IBM, 1994.

[10] D. Kotz and R. Gray. Mobile Agents and the Future of the Internet. *ACM Operating Systems Review*, 33(3):7–13, 1999.

[11] D. Milojicic, M. Breugst, I. Busse, J. Cambell, S. Covaci, K. Kosaka, D. Lange, K. Ono, M. Oshima, C. Tham, S. Virdhagriswaran, and J. White. MASIF: The OMG Mobile Agent System Interoperability Facility. *Proceedings of the Second International Workshop on Mobile Agents*, pages 50–67, September 1998.

[12] E. Royer and C-K. Toh. A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks. *IEEE Personal Communications Magazine*, 6(2):46–55, 1999.

[13] G. Dommety and R. Jain. Potential Networking Applications of Global Positioning Systems (GPS). Technical report, CS Dept., The Ohio State University, 1996.

[14] G. Vigna. *Lecture Notes in Computer Science*, volume 1419. Springer-Verlag, 1998.

[15] I. Terekhov and T. Camp. An Efficient Location Management Technique for Mobile Computing. *IEEE Transactions on Networking*, submitted May 1998.

[16] J. E. White. Telescript Technology: The Foundation for the Electronic Marketplace. Technical report, General Magic Inc., Mountain View, CA, 1994.

[17] J. Macker and M. Corson. Mobile Ad Hoc Networking and the IETF. *Mobile Computing and Communications Review*, 2(1):9–14, February 1998.

[18] M. Haar, R. Cunningham, and V. Cahill. Supporting CORBA Applications in a Mobile Environment. *Proceedings of the International Conference on Mobile Computing and Networking (MOBICOM)*, pages 36–47, 1999.

[19] Object Management Group. The Common Object Request Broker: Architecture and Specification V2.2. Technical report, Object Management Group, 1998.

[20] P. Clements, T. Papaioannou and J. Edwards. Aglets: Enabling the Virtual Enterprise. In *Managing Enterprises - Stakeholders, Engineering, Logistics and Achievement*, 1997.

[21] R. Dube, C. D. Rais, K.-Y. Wang, and S. K. Tripathi. Signal Stability Based Adaptive Routing (SSA) for Ad Hoc Mobile Networks. *IEEE Personal Communications*, 4(1):36–45, February 1997.

[22] S. Basagni, I. Chlamtac, V. Syrotiuk, and B. Woodward. A Distance Routing Effect Algorithm for Mobility (DREAM). *Proceedings of the International Conference on Mobile Computing and Networking (MOBICOM)*, pages 76–84, 1998.

[23] S. Murthy and J. J. Garcia-Luna-Aceves. An Efficient Routing Protocol for Wireless Networks. *ACM Mobile Networks and Applications Journal*, 1(2):183–197, October 1996.

[24] S. Vinoski. CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments. *IEEE Communications Magazine*, 35(2), February 1997.

[25] V. D. Park and M. S. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. *Proceedings of IEEE INFOCOM '97*, pages 1405–1413, April 1997.

[26] Y. Ko and N. Vaidya. Location Aided Routing (LAR) in Mobile Ad Hoc Networks. *Proceedings of the International Conference on Mobile Computing and Networking (MOBICOM)*, pages 66–75, 1998.

[27] Y. Ko and N. Vaidya. Geocasting in Mobile Ad Hoc Networks: Location-Based Multicast Algorithms. *Proceedings of the* 2^{nd} *IEEE workshop on Mobile Computing Systems and Applications*, 1999.