Intelligent Agent Routing for Mobile Ad-hoc Networks

by

Yan Zhou

Submitted in partial fulfillment of the requirements
For the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
April 2003

Table of Contents

List of Figures

List of Tables

List of Abbreviations and Symbols Used

RIP	Routing Information Protocol
OSPF	Open Shortest Path First
DSDV	Destination Sequenced Distance Vector
AODV	Ad Hoc On-Demand Distance Vector
DSR	Dynamic Source Routing
MAR	Mobile Agent Routing
PDA	Personal Digital Assistant
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
CBR	Constant Bit Rate

Abstract

This thesis introduces a new agent-based ad-hoc network routing protocol: Mobile Agent Routing protocol (MAR). Mobile agents are distributed, intelligent packets that carry data and explore the network with their intelligence. They communicate with one another to exchange the routing information. By mobile agent exploration, we put the intelligence across the network and make the routing distributed and adaptive. We compare our protocol with another state-of-the-art ad-hoc routing protocol: Destination Sequenced Distance Vector (DSDV). We perform extensive experiments to evaluate the performance of MAR. The results of the t-Test analysis show that MAR performs significantly better than DSDV within 95% confidence interval at speeds greater than 0.1 m/s.

Acknowledgements

This thesis was supervised by Dr. Nur A. Zincir-Heywood. I am very grateful for her constructive suggestions, and for her valuable time. Many thanks also go to Dr. Allan G. Jost and Dr. Ernst W. Grundke, who serve as committee members for my thesis defence. Furthermore, I would like to thank Wen-Hsuan Huang, David Webb and Jing Zhang for their help in reviewing this thesis.

1 Introduction

1.1 Routing

A network consists of a set of nodes as shown in Figure 1. Generally there exists more than one route between two nodes. Routing is the technique which finds the route to transfer data packets from one node to another. Although the definition is simple, routing could be very complicated.

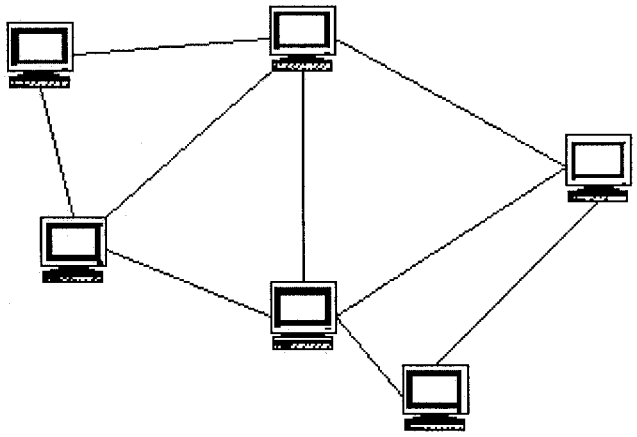

Figure 1: A simple network

RIP and OSPF are two traditional routing protocols. RIP is a distance vector protocol and OSPF is a link state protocol. In the distance vector protocol nodes use frequent broadcasts of their entire routing table to neighbors to exchange routing information. In the link state protocol, every node advertises its partial view of the entire network to all

other nodes. Distance vector and link state are used widely in many networks because they are easy to configure and maintain. However, the frequent periodic update process they use introduce considerate resource overhead. Therefore they cannot be applied in wireless networks. Thus, new approaches are needed for routing on mobile networks.

1.2 Ad-hoc Network Routing

Mobile ad-hoc networks developed very quickly and have been the focus of a lot of research recently. A mobile ad-hoc network is a connection of many wireless nodes with short transmission range that communicate with one another generally by multi-hop routes [11]. In wireless ad-hoc networks, every node also acts as a router, so the control of the network is distributed across the network instead of on several privileged central nodes as in Mobile IP.

Because a mobile node is a wireless device, it has to use wireless radio communication. That means the bandwidth is usually more limited than in wired networks. Also, to reserve power in wireless nodes and make use of the bandwidth efficiently, a node is characterized by a short transmission range. To save costs, the device in mobile ad-hoc networks is characterized by limited memory [12]. In addition, because of the frequent movement of mobile nodes, the topology of an ad-hoc network is generally dynamic. Due to the dynamic multi-hop communication, limited memory and bandwidth, it is critical to

develop new routing protocols distinct from traditional routing protocols for mobile ad-hoc networks.

There are already some mobile ad-hoc routing protocols developed such as Destination Sequenced Distance Vector (DSDV), Ad Hoc On-Demand Distance Vector (AODV) and Dynamic Source Routing (DSR) [6]. They all have their advantages and disadvantages. DSDV tends to provide low delay at the cost of much periodic update process overhead, while the others endeavor to reduce excessive overhead at the cost of introducing initial delay [7]. Furthermore, with contemporary networks becoming larger and more heterogeneous, research on more distributed and intelligent ad-hoc routing protocols becomes critical [4]. That is why the mobile agent has become an exciting and promising research area for mobile ad-hoc networks.

In this work, mobile agents are simple packets that explore the network and collect data to be used for routing. They communicate with one another indirectly to exchange routing information (they write and read data to /from routing tables) as ants exchange food information with each other. By using mobile agent exploration, we put the intelligence across the network and make the routing distributed. Furthermore, in mobile agent based routing protocols the overhead is proportional to the number of agents [4]. That may reduce a lot of overhead compared with traditional routing protocols that use

frequent updates. In this thesis our objective is to design and implement a new Mobile Agent Routing protocol for mobile ad-hoc networks.

1.3 Sections

In this work, we design and implement a Mobile Agent Routing protocol called MAR, analyze its performance and compare it with other protocols. Chapter 2 discusses some existing mobile ad-hoc network protocols and related research on mobile agent technology. Chapter 3 details the mobile software agents. Details of MAR are given in chapter 4. Chapter 5 presents simulation results and in chapter 6 conclusions and future work are discussed.

2 Literature Review

2.1 Pre-computed Routing and On-demand Routing

Current ad-hoc protocols can be divided into two groups on the basis of when the route is computed: pre-computed routing protocols and on-demand routing protocols [8]. In pre-computed routing protocols, routes to all destinations are computed and maintained by a periodic update process. In on-demand routing protocols, on the other hand, the route to a specific destination is computed only when necessary, e.g., data packets demand a route to be sent to a destination.

On-demand routing protocols can scale well to large networks. First, because the route is computed only when necessary, there is no permanent entry in the routing table. So generally, there are far fewer entries in the routing table at a time compared with pre-computed protocols. This can save scarce memory. Second, because there is no periodic update process, there is less overhead. However, on-demand routing also has some disadvantages. First because on-demand routing starts to compute routes when it needs to do, on-demand routing introduces initial delay to find necessary routes, which degrades the overall performance, especially for interactive applications. Second, on-demand routing increases the delay to queue many packets without valid routes to destinations [1].

Pre-computed protocols, on the other hand, generally are suitable for interactive applications because they do not have initial delay as on-demand protocols [2]. Pre-computed protocols have their own disadvantages as well. They consume more bandwidth resource by the frequent update process to exchange routing information. Moreover, they do not adapt fast enough to topology changes.

2.2 Destination Sequenced Distance Vector (DSDV)

The Destination Sequenced Distance Vector routing protocol, also known as DSDV, is a pre-computed routing protocol based on the classical Bellman-Ford routing algorithm. DSDV is essentially a variant of the RIP protocol. It adds sequence numbers to routing table entries to eliminate routing loops and keep routing entries fresh [6]. Each entry in the routing table has a sequence number. If a new entry is given, DSDV prefers the one with the greatest sequence number. If their sequence numbers happen to be the same, DSDV chooses the metric with the lowest value [6]. DSDV also uses triggered update processes to propagate topology changes. However, DSDV does not perform very well under quite dynamic environments [1].

2.3 Related Research

There have already been many experiments performed for the comparison among different mobile ad-hoc network routing protocols, especially for DSDV, AODV and

DSR [1]. In [1], the ad-hoc network routing protocols are divided into proactive and reactive, namely pre-computed and on-demand as we described above. DSDV is a proactive protocol, whereas AODV and DSR are reactive protocols. Extensive experiments in [1] show that reactive protocols such as AODV and DSR perform better than the proactive protocol DSDV in most scenarios. Especially in quite dynamic scenarios such as the Disaster scenario in [1], DSDV performs so poorly that it should not be used in this kind of environment. However, reactive protocols have much longer delay because of the initial delay they introduce. In the Disaster scenario experiment in [1], the average delay of DSDV is 0.196 seconds, while those of AODV and DSR are 0.988 seconds and 1.187 seconds respectively. This kind of long delay may not be acceptable in interactive applications [2].

In [2], a new perspective of Bluetooth-based mobile Ad-hoc networks is given. Bluetooth is a wireless technology that lets computers, mobiles phones and personal digital assistants (PDA) be easily connected with each other by short range wireless radio connections instead of cable connections. In a word, Bluetooth is a technology for cable replacement [13]. Recently Bluetooth is still only used for connections between central units and their peripheral devices such as wireless headsets and mice. Moreover, in this kind of promising network technology, it is demonstrated that proactive routing protocols are superior to reactive routing protocols because of the characteristics of small delay, quick adaptation and limited mobility of Bluetooth scenarios [2].

It seems that there is a conflict between overhead reduction and delay reduction in mobile ad-hoc network routing. A proactive protocol tends to have relatively small delay but introduces excessive overhead, thus decreasing the network scalability. A reactive protocol has less overhead but tends to have too long a delay to be suitable for interactive networks or Bluetooth-based ad-hoc networks.

In recent research, the "rumor routing" scheme is introduced as an attempt to solve the above problem [3]. In rumor routing, mobile agents explore the network to create routes to any event they have witnessed. In [3], events are described as abstractions to identify anything happening on the network such as the node's processing capability or sensor readings. Each node has an event table. When a node finds an event from its sensor readings, it adds this event with distance zero in its event table. It then probabilistically generates and sends an agent, and the probability is defined by some strategies [3]. Once the agent is created, it walks across the network and propagates the event to other nodes. The distance metric value increases by 1 in each hop the agent takes. The agent walks for some predefined time, and then dies. This kind of propagation is similar to the way that ants, social insects, leave chemical odor on the path they follow [5]. Rumor routing is designed specifically for large-scaled sensor networks to monitor events but is still suggestive for mobile ad-hoc network routing, especially agent-based routing.

The formal concept of software mobile agents is introduced in [4]. This kind of mobile agent is characterized by small size, longevity and mutual invisibility. However, by cooperating indirectly, they enable the network to adapt to the changes happening. We believe routing protocols with intelligent mobile agents make the network more adaptive since there are always some mobile agents exploring the network. Furthermore, because the mobile agent walks to only one specific node at a time instead of flooding the network with update packets, the overhead in agent based routing is proportional to the number of agents. This in return reduces the overhead compared to the traditional proactive protocols. In this way, agent based routing increases the scalability of the network. In addition, by using mobile agents exploring the network to compute routes actively, the delay can be decreased as well.

Therefore, mobile agent technology can likely be used to solve the problems mentioned above and improve ad-hoc network routing. Motivated by this promising perspective, and based on the study on software mobile agents, our objective is to develop a relatively simple, low overhead and rapidly adaptive routing protocol for mobile ad-hoc networks. The following section discusses software mobile agents in [4] in detail.

3 Mobile Agents for Dynamic Network Routing

3.1 Introduction to Mobile Agent

As indicated in previous sections, mobile agents are simple packets carrying data and exploring the network. The agents' data can be modified by the nodes, so that they can carry data collected by other agents. These mobile agents make the network routing decentralized and distributed. In this kind of decentralized routing, there is no need for central privileged nodes because the routing job is done by the cooperation of all software agents. This in return enables the network to become more scalable and more adaptive to changes.

The MIT Media Laboratory introduced a dynamic, wireless, peer to peer network with routing tasks performed in a distributed and decentralized environment using mobile software agents [4]. Generally the agent of Media Lab is characterized as follows [4]:

- An agent can easily explore the network by traveling from a node to its current neighbors.
- Every agent works independently without direct communication.
- An agent can share the information it gathered and write the information to a node as it travels across the network.
- An agent cooperates with other agents indirectly. This indirect inter-agent communication is called stigmergy.

- Every agent is small in size. This minimizes the overhead created by agent creation and transportation.
- Every agent is long-lived. In principle, every agent explores the network for a long enough time to collect routing information.

Such mobile agent technology looks promising for wireless routing protocols since it is simple, intelligent and efficient [4].

3.2 Mobile Agent Research in MIT Media Lab

The research described in [4] used mobile agents to route packets in mobile and multi-hop networks. This kind of mobile agent is unusual since it explores the network without a specific destination. In theory it never stops. It seems that it just travels across the network randomly, collects and leaves information, and goes on to another node. Once a routing agent arrives at a node, it updates the routing table of the node it is visiting with the data it gathered, and then moves again to another neighbor node by some predefined strategy.

The most important idea about the procedure of "mobile agents walking" is given by MIT Media Lab as follows [4]:

1. First, the agent looks at all the neighbors of the node it is currently visiting, and then it makes a decision about which one to visit next.

2. Second, the agent moves to the new node it has selected, and learns the new route from the latest visited node to the new node.

3. Third, the agent updates the routing table of the new node using the knowledge learned from its trip.

4. Then the agent repeats steps 1 - 3.

It is very important to have a good methodology to decide to which node the agent should move next. This strategy is a key to guarantee that the mobile agent is able to accumulate valuable routing information as quickly as possible. Here are two strategies given to do so:

1. Random walking

   The agent simply moves to a randomly selected neighbor. This method is easy to implement, and of course, reduces calculation time.

2. The oldest node

   For this strategy, the agent will prefer to visit a neighbor node that it visited a "long" time ago or has never visited before. It is said that this strategy will perform more fairly and efficiently than the "Random walking" strategy in [4].

3.3 Potential Drawbacks

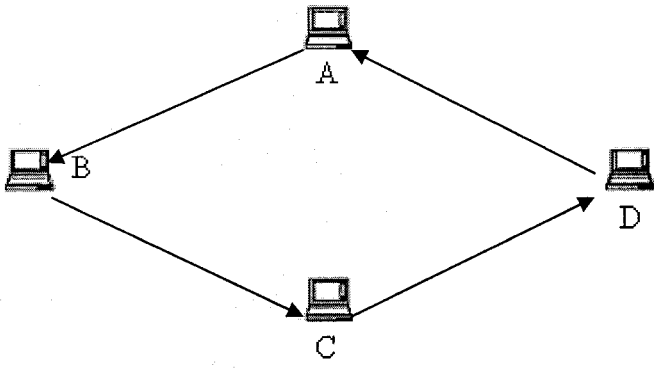

Figure 2: Problem of "the oldest node"

Figure 2 shows a small network consisting of 4 computers. Initially the agent starts at node A. Assume it moves to node B. By the oldest node strategy, it will be forced to move to node C, then node D, then node A again. It will not reverse direction so nodes cannot get shortest paths for some nodes. For instance, node A only knows that the distance to node B is 3 and the next hop is node D.

The tradeoff between agent number and performance was investigated in [4] and some analysis was given. However no experiments on throughput and delay were performed, which would provide the most important indices to evaluate a routing protocol.

Furthermore in [4], agents can only write information to the routing table, but cannot read from it. This may degrade the power of stigmergy. In addition, the problem where agents may die in real networks (because of unexpected situations) was not investigated. To be

more realistic, such a problem should be considered and a strategy should be designed to solve it.

Our objective is to build a system based on the concept of the MIT mobile agents, but which also solves those problems mentioned above. By extensive experiments, we demonstrate the feasibility of mobile agents for routing and present suggestions for using mobile agents for rumor routing. In the following section we present a Mobile Agent Routing protocol (MAR).

4 Mobile Agent Routing Approach

As described in the previous section, although the mobile agent system as designed by [4] has some advantages, it also has some disadvantages and potential problems. Thus, our approach is based on utilizing mobile agents for rumor routing in an attempt to solve those problems.

4.1 Mobile Agent Routing Methodology

To solve the disadvantages described above, our approach to routing is based on sharing local information to find paths to destinations. By local information, we mean the fact that every node on the network only knows about its neighbors to start with.

4.1.1 Thorough Stigmergy

In a network based on mobile agents, a mobile agent walks from node to node and collects information during its travels. In other words, the mobile agent explores the network and collects information for routing. The system performance relies on the cooperation of all mobile agents in the network. Agents in a population do not communicate with each other directly; instead, they leave some information in every node they visit, and then move on. This is called stigmergy.

In the system designed in [4], agents only leave information in the nodes they have visited, but they do not read any information from those nodes. That, actually, is not real stigmergy and limits the communication among mobile agents.

To be more co-operative, our MAR design enables the mobile agent not only to leave information on nodes it has visited, but also to read information left by other mobile agents. Specifically, when an agent arrives at a new node, it first updates the routing table of this node with the information it has collected. Then, it reads routing information from the local routing table. Therefore, mobile agents with this characteristic, like ants, leave information behind on nodes they have visited that other ants can use later. We call this "thorough stigmergy". In this way, all mobile agents co-operate with others more efficiently and more actively.

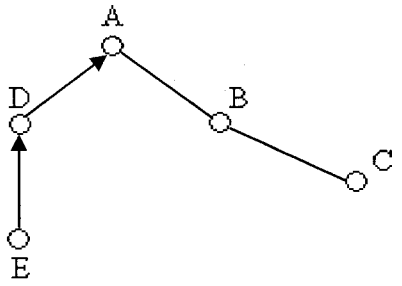

Figure 3: Stigmergy communication

In Figure 3, a mobile agent, which carries information about node D and node E, arrives at node A. Therefore this mobile agent has the information of D and E, whose distances from A are respectively 1 and 2. In node A, suppose that the routing table already has information about node B and node C, whose distances are 1 and 2 respectively. Once the

mobile agent leaves its information on node A, the number of routing entries in the routing table of node A increases from 2 to 4 (D, E, B and C). On the other hand, after the mobile agent reads information from the routing table of node A, it also acquires the information about node B and node C, even though it has not visited these nodes yet. This scheme becomes even more important when a new agent is created. Details of this will be discussed in the following section.

4.1.2 Controlling the Number of Agents

A mobile network is a dynamic network, and the routes are established and broken frequently as nodes move in and out of the transmission range. It is possible that a node can move out of the transmission range of all others and becomes isolated. If a mobile agent happens to reside at such a node, it becomes stranded. Thus its contribution to the population as a whole is lost. Also mobile agents may be lost during transmission. If we do nothing about such cases, the effective number of mobile agents will drop as time passes. As we know, in mobile agent based networks, system performance relies on the cooperation of all mobile agents [4]. Thus, a drop in the number of agents will degrade the network performance. In order to avoid this, we introduce a strategy to control the number of agents during the simulation. It is assumed that there is no node that crashes during the simulation.

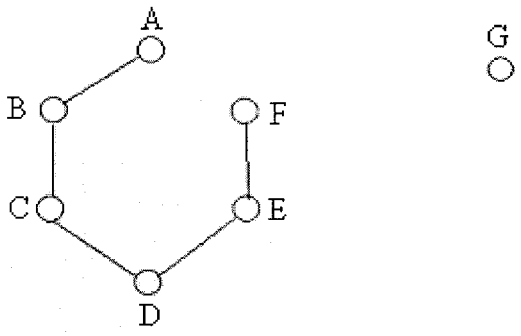

Figure 4: Stigmergy for new agent created on isolated node

Figure 4 shows node G that contains a mobile routing agent has moved out of the transmission range of other nodes and become "isolated". If it stays isolated for a long time, then the mobile agent in this node will lose its contribution to the population because it is not able to visit any other node and collect routing information.

Another possible condition is that the agent in node A detects node B as one of its neighbors and starts to move to B because node B is currently in the transmission range of node A. However, node B can start to move out of the transmission range exactly the same time. The mobile agent under this circumstance may be lost because node B may move out of the transmission range before it arrives. It should be noted that in our approach UDP is used in the transport layer.

In both of these two situations, a mobile agent can eventually become isolated. It is crucial to deal with this condition. We believe controlling the number of mobile agents is necessary under these circumstances, which will be demonstrated in our experiments.

To solve this problem, a global registry system is defined for the mobile network. Although we know that this is against the objective of using only local information, to be able to measure the performance of the system, we will use this simple and quick method. All mobile agents register when they are created, and re-register each time a node receives a mobile agent. We also set a threshold time and an agent's entry will "time out" in the registry if it does not re-register within this threshold time. The system counts the total number of mobile agents currently registered in the global register, and introduces new agents if it detects that an agent is "dead". The following is the detailed algorithm for controlling the number of agents:

Assume there are n nodes in the network, and we want m routing agents. It has been demonstrated in [4] that some m less than n can give fairly good results.

There is a global registry <agent id, expiry time> that registers all current routing agents.

1. Initially, nodes are created (node by node, from $id = 0$ to $id = n-1$) and a node will create an agent and send it out unless the size of registry reaches m. Every agent will be registered in the registry with an expiry time once it is created.

2. Once a node receives a routing agent, it updates the relevant entry in the registry with a new expiry time, which makes this entry fresh.

3. Periodically every node will check the registry and purge any expired routing agent. If for some reason such as mobile agent isolation, or signal interruption, some routing agents are dropped or lost, then the relevant entry in the registry will be purged. Then the node will check to see if there is a routing agent existing in the registry whose id equals to the node id. If that is not, this node will create a routing agent, register it in the global registry, and then send it to one of the neighbors. If the id already exists, then it does nothing, but just waits for other nodes to create new routing agents.

4. Periodically, a node checks the registry to see how many routing agents it has. If the size of the registry is less than *m,* the node will check if there is a routing agent existing in the registry whose id equals to the node id. If it is not, this node will create a routing agent, register it in the global registry, and then send it to one of the neighbors. If the id already exists, then it does nothing.

5. At any time when the size of registry reaches *m*, nodes will no longer create routing agents.

We believe that for the future work, a more efficient method should be developed to replace the "global registry".

4.1.3 Other Features

4.1.3.1 "Sending Count" Neighbor Selection Strategy

In every node of our protocol, there is already a neighbor list that stores all neighbors of that node. Typically, the entry in the neighbor list has only the name of the neighbor node. We also develop another strategy that stores the number of routing agents sent to the entries in the neighbor list. In other words, the neighbor list stores both the name of a neighbor node, and the number of times mobile routing agents move from the current node to the relevant neighbor, which we called "sending count".

When a routing agent arrives at a node, it checks the neighbor list. First, it will randomly select a neighbor that it has not visited previously. If all neighbors have already been visited, the neighbor node with the smallest number of sent agents will be selected as the next node to move to. After the routing agent has made this decision for a node, it updates the entry of the selected node with the value of "sending count" plus 1, and then moves to the selected node. This method is designed to find new paths, since it encourages mobile agents to visit the least visited node first. We test this method and the experimental results are given in the appendices.

4.1.3.2 Handling of Route Failures

In mobile networks, because nodes move frequently, routes are also established and broken frequently. Handling these changes is very important. We developed the following update scheme for MAR.

1. Set the expiry time for every entry in the routing table. Once an entry has expired, instead of deleting it immediately, set the distance in this entry to infinity. Because the distance of that entry becomes infinity, any new route will immediately replace the old entry.

2. Every node has its neighbor list. Periodically, every node broadcasts a "Hello" message but only to its immediate neighbors (one hop away), to announce "I am your neighbor". Periodically, each node will check its neighbor list as well. Once a node detects that a neighbor is down, it deletes that node from its neighbor list, and then deletes all entries in the routing table whose next hop is that neighbor.

We believe this scheme will help routing agents to find new routes instead of sticking to the old routes that do not exist any more.

4.1.3.3 "Trip Time" Instead of Hops

In MAR, we also wanted to test another hypothesis where we use "trip time" to replace the number of hops as the measurement for distance of a path. We let the routing agent also store the time spent traveling from one node to another and use trip time instead of hops to judge if a path is "good" or "bad". In MAR, when a routing agent arrives at a node, first it updates its information with the accumulated trip time, and then it updates the routing table. Because some routes are longer and some routes are shorter in a network, it seems good to use trip time act as the metric. However, our experimental results show that this scheme did not make much improvement compared to the scheme using "hops". The results are given in section 5.4.

4.1.3.4 "Hello" Messages for Updating Routing Tables

Once a node receives a "Hello message", it will update the routing table immediately instead of waiting for an agent. Without waiting for a mobile agent to update the routing table, this strategy is straightforward but it is also a crucial strategy because it saves a lot of routing update time.

4.2 Details of MAR

Mobile agents should be small in size to minimize the overhead. Thus, we designed a simple structure for agents in MAR. The update operation for the routing information in a mobile agent is based on the agent structure.

4.2.1 MAR Agent Format

In order to minimize the network overhead, our MAR agent has a simple structure. Figure 5 presents the format of the mobile agent developed for MAR:

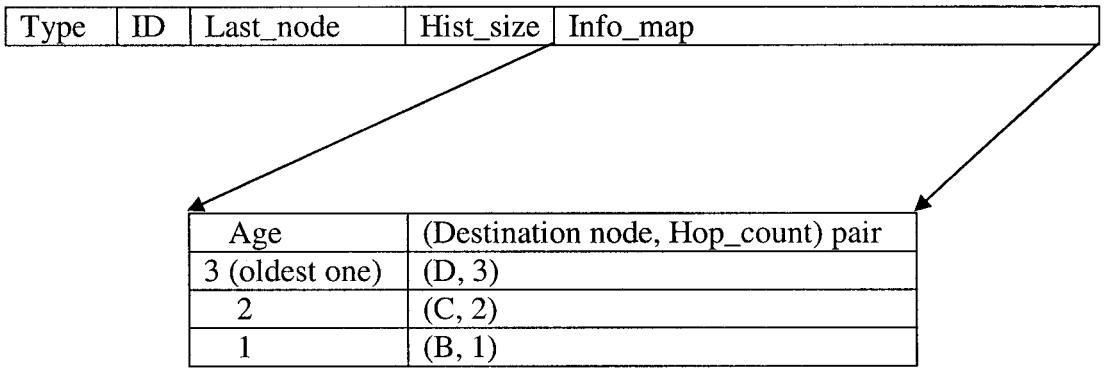

Figure 5: MAR routing agent format

In this structure, "Type" is used to differentiate routing agents from other packets. "ID" is used as a unique id number of every agent. "Last_node" stores the previous node just visited. "Hist_size" defines how big the Info_map is. "Info_map" is the container that stores all gathered routing information. Every time the agent arrives at a node, it updates

the Info_map based on its gathered routing information about the network. In Info_map, "Age" records the sequence of the nodes visited. The larger the sequence value is, the older the information is.

4.2.2 Detailed Operations Related to the Agent Information Update

When the "oldest node" strategy is adopted, the oldest node is the entry with largest sequence number, which means the largest Age value. The agent will try to select one neighbor that does not exist in its Info_map, which means that it has never been visited or it has been deleted from the history. If all the neighbors happen to be in its history, which means they have all been visited before, the agent will select the earliest visited node from its history.

Since the agent size should be small to minimize overhead, we set a history size threshold for every agent. Once an agent arrives at a new node, if the history size does not reach the threshold value, the new node entry will be inserted to the history. However if the history size has already reached the threshold value, before the new node entry is inserted, the oldest node, which is the entry with largest Age value, will be deleted from the history in order to open up space for the new one.

5 Simulation and Performance Analysis

5.1 The Simulation Tool

All simulations on our mobile agent routing system were conducted in the network simulator NS2. NS2 is an object-oriented discrete event simulator. It was developed at UC Berkeley, particularly for networking research [9]. It supports the transport, network and underlying layers. It also provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks including several ad-hoc routing protocols [9].

NS2 is a dual-language simulation environment [10]. Its classes are written in C++ and OTcl. OTcl is the object-oriented version for Tcl. This makes it relatively easy for end users to use. However, because of its special node structure, it is not easy, sometimes even difficult, to include other algorithms and protocols or to integrate other network architectures in NS2 [10].

We chose to use NS2 as our simulation environment because of its extensive support for wireless networks, especially for ad-hoc network routing protocols under wireless environments. It is also the simulation platform used in [1].

5.2 General Simulation Environment

In our system, every node moves randomly in a two dimensional space. Routes are established and broken as nodes move in and out of transmission range respectively. We assume that the radio connection is bi-directional and reliable once established.

In order to save power and use bandwidth more efficiently, nodes in our system are characterized by short communication ranges. Because of this characteristic, data packets from one node generally need multiple hops to reach their destinations. This basic scenario also matches realistic wireless technologies such as Bluetooth.

At first, we developed a general random scenario that was characterized by short transmission range nodes in a multi-hop environment. Nodes in this scenario moved completely randomly. After that, to make the experiment more realistic, we developed three realistic scenarios based on this general scenario.

In [1] many experiments have already been done to compare pre-computed protocols with on-demand protocols. Thus, in this work, the focus is to compare and evaluate the performance of MAR versus DSDV. The reason we chose DSDV is that it represents Bluetooth type technologies [2]. MAR is a pre-computed protocol. However, instead of

periodically broadcasting to exchange information as traditional pre-computed protocols do, MAR utilizes intelligent routing agents to walk across the network to gather routing information. On the other hand, DSDV use periodic update processes to exchange routing information. Hence, we compared MAR and DSDV under different scenarios. In all cases, we evaluate the performance in terms of the number of received packets and average delay. Here, the number of received packets means the number of data packets sent from source nodes and received in destinations successfully, and delay is the total trip time it takes for a data packet to move from its source to its destination. Similar to [1], we use CBR for all traffic flows, and we use UDP in the transport layer. All the experiments were conducted on a PC (Intel Celeon 1.0 GHz, 256M memory), running Linux Mandrake 8.2.

5.3 Key Parameters for MAR Performance

In the Mobile Agent Routing protocol, there are 2 key parameters that are critical for the protocol performance. These are the number of agents and the history size. In theory, the higher the agent number and history number are, the better the performance is expected [4]. However, more routing agents and long histories also introduce more overhead into the network. In this work, we controlled the number of agents using a global registry just to be able to conduct the experiments and measure the performance. However, we believe that future work is needed to make this scheme efficient and effective.

5.4 Random Scenarios

From appendices table 1 and table 2, we can see that "sending count" strategy did not perform better than "oldest node" strategy, and the agent using "trip time" did not perform better than using "hop count". Therefore, in the following experiments, we employed "oldest node" strategy and "hop count".

5.4.1 Configuration of the Simulation Environment

In the "Random" scenario mobile nodes moved randomly with a specified average speed (changed from scenario to scenario), where both the speed and the direction were based on the uniform probability distribution function. We evaluated the protocol performance under different dynamic network environments using different average speeds of the nodes. The average speed is constant during every simulation.

In our random scenarios, there were 50 mobile nodes in an area of 40 by 40 meters. Every mobile node moved randomly in the scenario and the transmission range was 10 meters, which was equal to the transmission range of a PDA defined in Bluetooth. Simulation time was set to 250 seconds. We ran the scenario with 6 different levels of speed. We tested "slow motion" by setting speed to 0.01, 0.05 and 0.1 meters per second. The goal of experiments at low speeds is to evaluate the performance in terms of delay and the number of received packets under relatively static scenarios. On the other hand, for high speed scenarios it is set to 1, 2 and 3 meters per second, which is similar with the

speed of people walking and jogging. If the transmission range of the node is only 10 meters, the scenario where nodes move by 3 m/s is actually quite dynamic. The goal of doing experiments under high speed environments is to evaluate the performance under very dynamic scenarios.

There are 15 traffic flows in the above cases. The traffic type is CBR, namely, constant bit rate. Every packet is 64 bytes in length. The packet sending rate is 5 packets per second. Considering that the simulation time is 250 seconds, there are 5*250*15, or 18750 data packets in total in every simulation. Table 1 shows the parameters used in detail:

Transmission range	10m
Bandwidth	2Mbps
Simulation time	250 seconds
Number of nodes	50
Pause time	2 seconds
Environment size	40m by 40m
Traffic type	CBR
Packet rate	5 packets/s
Packet size	64 bytes
Number of traffic flows	15

Table 1: Parameters for random scenarios

5.4.2 Experimental Results for Random Scenarios

In this sub-section, we detail the results for random scenarios.

5.4.2.1 Configuring the Parameters

We know that the number of agents and the history size are critical parameters for the MAR protocol [4]. They directly affect the performance of MAR under different dynamic environments. Different values of the history size and the number of agents are represented in the form "history size/the number of agents" in the rest of this work. In principle, the higher the number of agents and history size are, the better the performance. However, too many agents and too long a history introduce too much overhead to the network. It was demonstrated in [4] that some number of agents smaller than the number of nodes can give fairly good results, especially in a relatively static environment where routes are quite long-lived. In the system of MIT Media Lab there are 250 nodes with the baseline of 100 agents with history size 25. In our experiments there are 50 nodes, for slow speed scenarios, history size is set to 5, and the number of agents is set to 10. For high speed scenarios, history size is set to 7, and the number of agents is set to 50.

5.4.2.2 Experiment Results

We ran the simulation for 20 times for each different speed scenario, and each time nodes moved with random direction and distribution. However, the average speed is constant during the simulation. The raw data is shown in the appendices. We measured the average delay and the number of packets received, as shown in tables 2 - 3:

	0.01 m/s		0.05 m/s		0.1 m/s	
	Delay	Packets received ratio	Delay	Packets received ratio	Delay	Packets received ratio
MAR	0.553960s	75.74%	0.245980s	73.37%	0.152245s	68.07%
DSDV	0.211366s	80.96%	0.179969s	74.77%	0.289699s	69.05%

Table 2: Delay and packets received ratio for speed 0.01, 0.05 and 0.1 m/s

	1 m/s		2 m/s		3 m/s	
	Delay	Packets received ratio	Delay	Packets received ratio	Delay	Packets received ratio
MAR	0.259218s	43.14%	0.056125s	33.27%	0.022500s	27.12%
DSDV	0.385568s	39.59%	0.440061s	33.27%	0.427426s	28.50%

Table 3: Delay and packets received ratio for speed 1, 2 and 3 m/s

We can see that MAR generally performs as well as DSDV. In terms of the number of packets received, MAR has almost the same performance as DSDV. In addition MAR has much less delay than DSDV under high dynamic scenarios. It should be noted here that it takes some time for MAR to converge. This means that initially all the routing tables are empty, with agents exploring the network to acquire routing information, the network gradually converge. Some future work should be done for this issue.

5.4.2.3 Performance Analysis

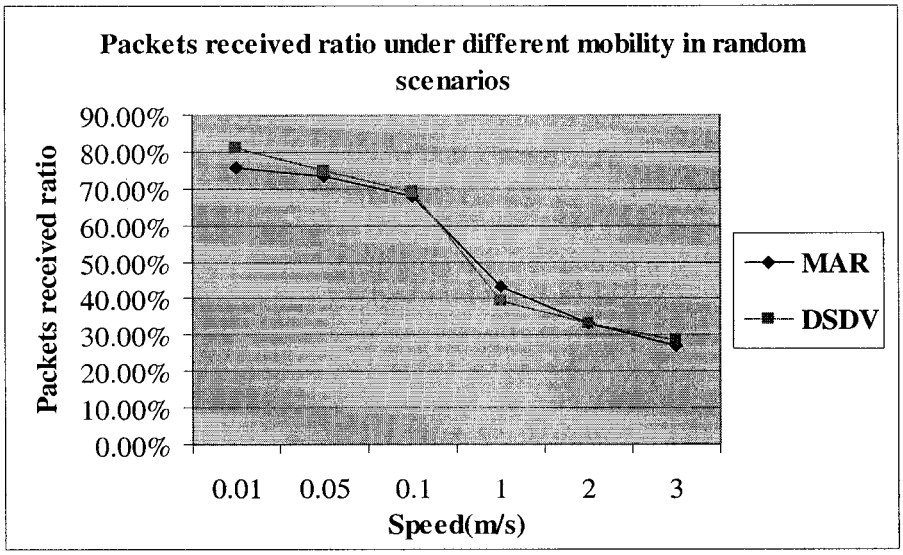

Figure 6: Packets received ratio under different mobility in random scenarios

From Figure 6, we can see that MAR and DSDV have very similar performance for the number of packets received for the 6 different speed scenarios. That means that MAR has almost the same ability, if not better, to route data packets as that of DSDV. Figure 7 shows the performance in terms of delay.

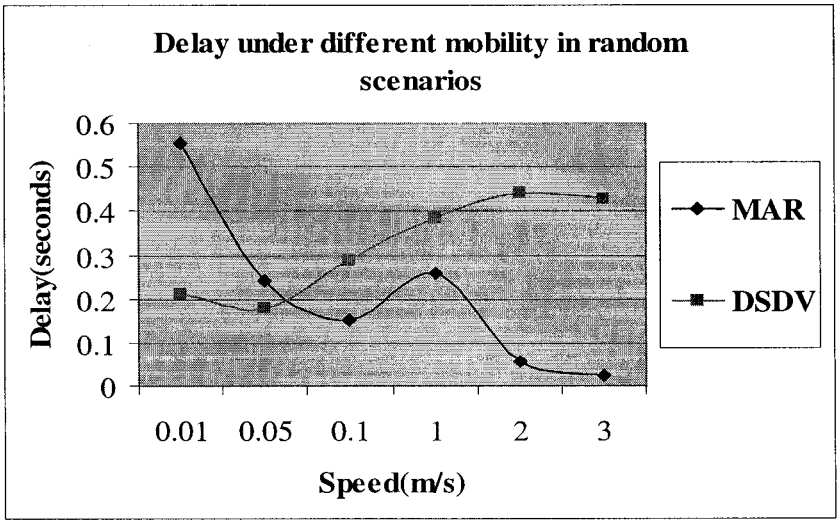

Figure 7: Delay under different mobility in random scenarios

We can see the delay performance is almost divided into two parts. MAR has a longer delay than DSDV at low speeds (smaller than 0.05 m/s). For speed 0.05 m/s, MAR has a similar delay to DSDV. In higher speed scenarios MAR has always lower delay than DSDV. That is really an advantage, especially for interactive applications that demand less delay. However, it should be noted that at those speeds, both MAR and DSDV could only deliver approximately 1/3 of the data packets.

On the other hand, in MAR, there is no periodic update process to exchange routing information, while our routing agents communicate indirectly with stigmergy. Thus, this reduces the overhead and improves the overall network performance.

5.5 Specific Realistic Scenarios

In order to evaluate the performance under more realistic environments, some realistic scenarios have been developed in [1]. To compare the performance of MAR and DSDV under more realistic environments, we also developed three more scenarios similar to [1]. The three scenarios we have developed are the conference scenario, the event scenario and the military scenario, detailed below. By simulating these 3 scenarios, we attempt to mimic realistic scenarios in real life, and evaluate protocol performance under these scenarios with different dynamic mobility.

5.5.1 Conference Scenario

The conference scenario is constrained in Figure 8.

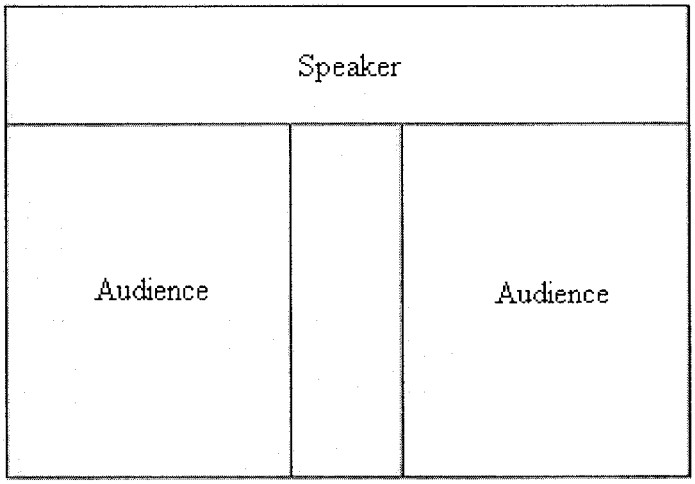

Figure 8: Conference scenario

The conference scenario models a conference environment such as a meeting room, a seminar or a group discussion environment. In this kind of an environment, generally the audience is quite stable while sometimes some people move randomly at slow speed. Therefore, they may break some links and may establish new connections. In our conference scenario, the maximum speed is 1 meter per second. It is defined that at any time only 10% of the nodes or fewer move. Generally a node moves at 0.5 m/s, occasionally at 0.4 m/s or 0.6 m/s.

The speaker is more active than members of the audience. He moves more frequently from the right to the left or vice versa. Furthermore, the speaker moves at a slightly higher speed. We define that the speaker moves at 1m/s in our conference scenario at most time.

There are 6 traffic flows in the scenario. Traffic is concentrated on the speaker. In our scenario, 4 out of 6 traffic flows are from the speaker to audience members. Table 4 shows the parameter setup for this scenario.

Number of nodes	50
Area	40m by 40m
Transmission range	10m
Speed for the speaker	1m/s
Speed for the audience	0.4 to 0.6m/s
Traffic number	6
Packet sending rates	5/s
Packet size	64 bytes
Simulation time	250s
Total sending packets	5x250x6=7500

Table 4: Parameters for the conference scenario

The simulation results are given in table 5.

		Delay (sec)	Packets received raito
DSDV		0.012251	77.4%
MAR		Delay (sec)	Packets received raito
History size	The number of agents		
7	3	0.043947	67.0%
9	3	0.060258	75.7%
12	3	0.056622	83.3%
15	***3***	***0.067228***	***85.0%***
18	3	0.070951	84.8%
20	3	0.084777	82.9%
7	7	0.222982	77.5%
5	8	0.173344	61.4%
6	8	0.155021	67.2%
7	9	0.261930	75.1%
5	10	0.256800	70.0%
6	10	0.323437	71.4%
7	10	0.333928	73.1%
4	11	0.244197	60.7%
5	12	0.333212	59.5%

Table 5: Performances for the conference scenario

In table 5, generally MAR and DSDV have good performance in terms of packets received ratio because of relatively static environment. We can see that MAR (15/3) has better performance. It has longer delay than DSDV. However, the packets received ratio is higher. In this scenario, the parameter values for the number of agents and history size has changed compared with those in random scenarios. A distinction between the conference scenario and random scenarios is that traffic is concentrated on one node in the conference scenario, whereas it is randomly distributed in random scenarios. This distinction may be the reason for the change of the values of history size and the number of agents.

5.5.2 "Event" Scenario

In the event scenario, it is assumed that people tend to form different groups for different events. There are also some scattered nodes, but not many, which do not belong to any group. Traffic is concentrated within a group. There is also low traffic between different groups or between a group and some scattered nodes. 60% people move at a medium speed, around 0.5m/s in their groups. Because groups are small (area of 150 meters by 150 meters), the average hops in groups are short, about 1.5 hops in this case. Occasionally, people move around with much higher speed and re-form new groups. In our event scenario, these higher speeds range from 2m/s to 4m/s. Furthermore, the characteristics of having a small number of average hops may make both MAR and DSDV perform well. In this scenario, we aim to test the performance under an environment where the topology changes by moving the groups.

Table 6 details the parameter setup for the "event" scenario:

Number of nodes	50
Area	400m by 400m
Transmission range	100m
Speed within groups	0.5 m/s
Speed to form a new group	2 m/s to 4 m/s
Traffic number	6
Packet sending rates	5/s
Packet size	64 bytes
Simulation time	250s
Total sending packets	5x250x6=7500

Table 6: Parameters for the event scenario

The simulation results of packets received and delay are shown in table 7.

		Delay (sec)	Packets received raito
DSDV		0.005714	70.2%
MAR		Delay (sec)	Packets received raito
History size	The number of agents		
15	3	0.046949	65.0%
15	5	0.056397	63.3%
13	6	0.062167	64.1%
12	7	0.110940	69.5%
13	7	0.109282	72.6%
15	7	0.115096	71.5%
18	7	0.129624	71.1%
10	8	0.099680	67.2%
12	8	0.093343	67.6%
13	8	0.099925	70.7%
15	8	0.100944	68.4%
5	10	0.074755	63.0%
10	***10***	***0.169360***	***73.5%***

Table 7: Performances for the event scenario

We can see that MAR (10/10) perform better than other configurations. MAR has a longer delay than DSDV, however it also routes more packets. On the other hand, the improvement in this scenario is not as good as in the conference scenario. We believe this is because DSDV benefits more from the low number of hops and a relatively static environment.

5.5.3 Military Scenario

In order to model an environment similar to a military or emergency/rescue scenario, we develop the military scenario. In this example, we assume that two marine groups have to perform their own missions in two different places. The communication between them can only be carried out through 2 helicopters between them. More than 90% nodes frequently move at very high speed within their zones to perform their missions. Two helicopters should try their best to stay in the best place to relay the traffic between these two marine groups.

In our military scenario, we define 2 subnets (one for each marine group) that the traffic flows between subnets can only be relayed by 2 vehicles, which always stay within the transmission range of these two subnets. These 2 vehicles move back and forth (northeast to southwest and southwest to northeast) with the speed of 1 m/s. Therefore, these two vehicles are critical for communication between sub networks. More than 90% nodes move at a very high speed (30 m/s) within the subnets they belong to. Therefore, this is a quite dynamic scenario. Mainly, traffic is concentrated between sub networks. There are 6 traffic flows in the scenario. Four out of 6 are between the two subnets. Table 8 shows the parameter setup for the military scenario:

Nodes number	50
Topology	1000m by 1000m
Transmission range	250m
Traffic number	6
Speed for helicopters	1m/s
Speed for other nodes	30m/s
Packet sending rates	5/s
Packet size	64 bytes
Simulation time	250s
Total sending packets	5x250x6=7500

Table 8: Parameters for the military scenario

Since we expected to get good performance of MAR in quite dynamic environments because of its low overhead, we performed 20 runs in the military scenario with different source and destination nodes (always 6 pairs though). The graph for the simulation results is shown in Figure 9.

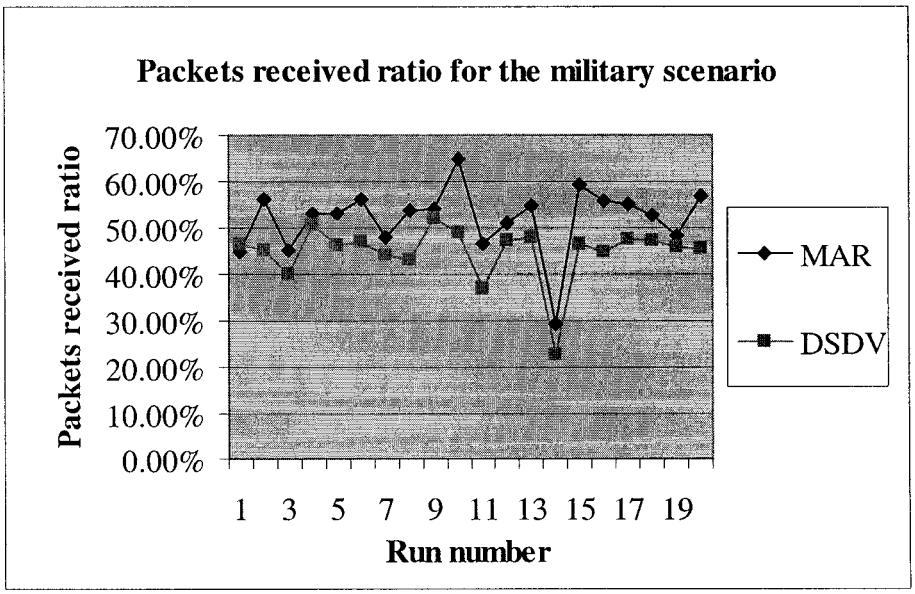

Figure 9: Performance of packets received ratio for military scenario

The average performance results are shown in table 9.

	Packets received ratio	Delay (sec)
DSDV	44.8%	0.049445
MAR (10/30)	52.0%	0.084636

Table 9: Average performance for the military scenario

From the results above we can see that MAR (10/30) performs much better than DSDV in terms of the number of packets received. Although the delay of MAR is a bit longer than that of DSDV, we believe it is worthy because of the improvement in the number of packets received.

6 Conclusion and Future Work

6.1 Conclusion

Basically, DSDV runs quite well in relatively static environments. It has an acceptable packets received ratio and low delay as shown in Figure 6. This is because DSDV, like other traditional proactive protocols, uses event driven update processes to adapt to topology changes. In relatively static environments, there are few link failures so DSDV has little bandwidth overhead. However, in dynamic random scenarios, because there are many more frequent changes taking place on the network, DSDV consumes much more bandwidth. This, therefore, results in the degradation of overall performance. The number of received packets drops as shown in Figure 6 and delay increases as shown in Figure 7.

MAR, on the other hand, performs almost the same as DSDV in terms of the number of packets received in relatively static environments. This is clear in Figure 6. However, in dynamic scenarios, MAR has much lower delay than DSDV as shown in Figure 7. This is understandable because in MAR, the overhead is proportional to the number of agents. Although MAR performs better than DSDV in terms of delay in dynamic scenarios, the number of packets received is very similar for both algorithms.

In three realistic scenarios, MAR generally has the same or better performance than DSDV in terms of the number of received packets. In the conference scenario, MAR has

more packets received but longer delay. In the event scenario, there is not much difference between DSDV and MAR. In the military scenario, MAR begins to show its advantage, as shown in Figure 9. It performs better than DSDV in terms of the number of received packets but the delay increases when the number of received packets increases.

In summary, as a new ad-hoc routing protocol, MAR can perform as well as DSDV and even better in quite dynamic scenarios. Tables 10 - 11 shows the results of the T-test analysis we conducted on the different scenarios within 95% confidence interval.

	Mean for DSDV	Mean for MAR	t-Test results
Random scenario 0.01 m/s	80.96%	75.74%	DSDV is better by 6%
Random scenario 0.05 m/s	74.77%	73.37%	No significant difference
Random scenario 0.1 m/s	69.05%	68.07%	No significant difference
Random scenario 1 m/s	39.59%	43.14%	MAR is better by 8%
Random scenario 2 m/s	33.27%	33.27%	No significant difference
Random scenario 3 m/s	28.50%	27.12%	No significant difference
Military scenario	44.80%	52.00%	MAR is better by 14%

Table 10: t-Test results for packets received ratio

	Mean for DSDV	Mean for MAR	t-Test results
Random scenario 0.01 m/s	0.211366	0.553960	DSDV is better by 62%
Random scenario 0.05 m/s	0.179969	0.240598	DSDV is better by 15%
Random scenario 0.1 m/s	0.289699	0.152245	MAR is better by 49%
Random scenario 1 m/s	0.385568	0.259218	MAR is better by 33%
Random scenario 2 m/s	0.440061	0.056125	MAR is better by 86%
Random scenario 3 m/s	0.427426	0.022500	MAR is better by 95%
Military scenario	0.049445	0.084636	DSDV is better by 37%

Table 11: t-Test results for delay

6.2 Future Work

In this thesis we investigated the use of mobile agents for rumor routing on mobile ad-hoc networks. The results are promising. However, there are still many open problems and research opportunities:

1. The avoidance of using a global registry. As we know, an agent-based system is a de-centralized and distributed system. By forcing mobile agents to explore the network, the system causes the routing task to be performed in a distributed way, which requires only local information, i.e. every node on the network should know only its neighbors. Therefore, the network is scalable and adaptive. However, to control the number of mobile agents in our experiments, we added a

global registry for agent registration. This will degrade the scalability of the network. Therefore, some future work should be done to avoid the use of a global registry.

2. The automatic setting of optimal configurations for the number of agents and history size under different scenarios. We performed a lot of experiments for performance evaluation of MAR. From these experiments we know that different configurations for the number of agents and the history size result in different performance. Therefore, in the future work, an intelligent algorithm is needed to adapt the history size and the number of agents to the changes in the network.

3. More experiments for different scenarios. In real life, there are may be some obstacles which can prevent radio communication in the network, as well as interference for the radio signal. Therefore, we need more scenarios where these factors are included. Furthermore, more experiments should be performed to measure other indices such as queue size.

4. The improvement of the strategy for node selection. To make the routing agent more intelligent, more effective strategies may be developed for selecting the next hop.

Appendices

Transmission range	10m
Bandwidth	2Mbps
Simulation time	250 seconds
Number of nodes	50
Pause time	2 seconds
Environment size	40m by 40m
Traffic type	CBR
Packet rate	5 packets/s
Packet size	64 bytes
Number of traffic flows	15

Table 12: Parameters of random scenario for "Sending count" and "Trip time"

Protocol / Mobility	Packets received for MAR with "oldest node"	Packets received for MAR with "sending count"
0.01m/sec	8414	8042
0.05m/sec	8194	8562
0.1m/sec	6667	5812
1m/sec	6852	4701
2m/sec	4164	3170
3m/sec	4413	3721

Table 13: Experimental results for "Sending Count"

Protocol / Mobility	Packets received for MAR with "hop count"	Packets received for MAR with "trip time"
0.01m/sec	8414	7075
0.05m/sec	8194	6189
0.1m/sec	6667	3825
1m/sec	6852	5458
2m/sec	4164	3933
3m/sec	4413	4329

Table 14: Experimental results for "Trip Time"

	0.01 m/s		0.05 m/s		0.1 m/s	
MAR	Delay	Packets	Delay	Packets	Delay	Packets
Test1	0.289978	14548	0.298978	15056	0.192888	11811
Test2	0.367371	11680	0.158263	16518	0.195896	14163
Test3	0.315236	15664	0.219054	12370	0.201996	14162
Test4	0.491853	14613	0.332617	12686	0.029483	12079
Test5	0.959154	16343	0.322406	15959	0.199871	12724
Test6	0.716344	11323	0.157235	15323	0.250676	12585
Test7	0.754895	16336	0.276726	8467	0.095806	11912
Test8	0.627998	15177	0.276837	14845	0.132820	13863
Test9	0.517394	12929	0.194134	12959	0.090761	12399
Test10	0.632827	13846	0.365988	14003	0.125591	11213
Test11	0.422166	12953	0.084351	15614	0.126624	14139
Test12	0.166465	13358	0.288282	13927	0.129248	10785
Test13	0.362050	14569	0.246552	14097	0.179328	9774
Test14	1.193492	12877	0.124482	16429	0.178981	14252
Test15	0.375207	16509	0.217156	13154	0.052052	14323
Test16	0.890390	13654	0.184019	15819	0.460898	11829
Test17	0.461087	13499	0.433237	12094	0.085636	12778
Test18	0.257987	15396	0.287163	11742	0.115386	12985
Test19	0.771725	14088	0.134779	13295	0.094626	13840
Test20	0.505586	14696	0.209692	10777	0.106328	13631
Average	0.553960	14202.9	0.240598	13756.7	0.152245	12762.35

Table 15: Delay and the number of received packets for MAR at speeds 0.01 m/s, 0.05 m/s and 0.1 m/s

	0.01 m/s		0.05 m/s		0.1 m/s	
DSDV	Delay	Packets	Delay	Packets	Delay	Packets
Test1	0.135061	15283	0.201068	15168	0.487482	12397
Test2	1.383763	12882	0.016696	16492	0.078112	14096
Test3	0.012695	16197	0.276175	12887	0.147752	13833
Test4	0.043191	16370	0.368696	13071	0.053210	14316
Test5	0.023258	16266	0.059700	15666	0.117051	14101
Test6	1.276800	11035	0.037484	14607	0.297409	12605
Test7	0.037974	16220	0.030174	8937	0.894812	10229
Test8	0.035578	15910	0.069132	15764	0.233989	13441
Test9	0.182915	14090	0.268598	13932	0.221469	12838
Test10	0.034205	15791	0.064468	14667	0.731616	10947
Test11	0.146049	14554	0.015520	15758	0.092593	14298
Test12	0.072213	15377	0.369553	13563	0.527237	11426
Test13	0.051859	16082	0.141845	14100	0.682404	9871
Test14	0.107955	13819	0.011002	16261	0.145913	14068
Test15	0.042639	16504	0.274639	13541	0.075394	14192
Test16	0.087551	14913	0.024020	15832	0.253445	12972
Test17	0.134535	15102	0.320454	12317	0.154512	13398
Test18	0.077400	16204	0.305507	12253	0.352730	12343
Test19	0.222048	15399	0.148127	14106	0.103436	13570
Test20	0.119635	15605	0.596513	11462	0.143408	14003
Average	0.211366	15180.15	0.179969	14019.2	0.289699	12947.2

Table 16: Delay and the number of received packets for DSDV at speeds 0.01 m/s, 0.05 m/s and 0.1 m/s

	1 m/s		2 m/s		3 m/s	
MAR	Delay	Packets	Delay	Packets	Delay	Packets
Test1	0.090030	9749	0.053673	5896	0.016395	5260
Test2	0.210061	7877	0.055338	6672	0.015992	4557
Test3	0.321642	7440	0.059729	5182	0.038996	4368
Test4	0.273618	9607	0.021643	5499	0.013177	6000
Test5	0.334530	7896	0.065324	6908	0.014184	4668
Test6	0.207014	7335	0.027523	6633	0.011985	4636
Test7	0.306794	8752	0.069876	6570	0.013299	4962
Test8	0.208505	7941	0.025706	5756	0.059957	4531
Test9	0.280104	7097	0.050151	7678	0.013481	5403
Test10	0.231596	8038	0.067307	6070	0.025081	5752
Test11	0.418398	8387	0.100471	6440	0.020409	5547
Test12	0.273455	7381	0.047006	5864	0.013182	5298
Test13	0.284822	8823	0.037311	5944	0.014473	4209
Test14	0.140841	8145	0.047552	5865	0.023578	4932
Test15	0.338059	8143	0.100010	5249	0.041076	6299
Test16	0.195716	7423	0.049231	6491	0.037727	4645
Test17	0.289757	8286	0.017505	5582	0.015954	5397
Test18	0.115930	8400	0.125346	6642	0.011944	5278
Test19	0.412596	7946	0.041008	6340	0.024034	4217
Test20	0.250886	7124	0.060790	7473	0.025083	5755
Average	0.259218	8089.5	0.056125	6237.7	0.0225	5085.7

Table 17: Delay and the number of received packets for MAR at speeds 1 m/s, 2 m/s and 3 m/s

	1 m/s		2 m/s		3 m/s	
DSDV	Delay	Packets	Delay	Packets	Delay	Packets
Test1	0.164581	8411	0.429166	5723	0.278029	5108
Test2	0.521676	7420	0.496271	6675	0.510697	4731
Test3	0.345908	7827	0.459629	5843	0.339555	5008
Test4	0.130611	8540	0.579384	4946	0.244326	6177
Test5	0.359182	7477	0.495685	6597	0.479456	5059
Test6	0.430269	6618	0.319511	7240	0.742547	4691
Test7	0.285658	8199	0.560110	6061	0.351210	5004
Test8	0.339180	6905	0.575229	6061	0.430055	5141
Test9	0.472238	6110	0.207500	7579	0.350103	5569
Test10	0.504562	6893	0.350032	6831	0.321301	6066
Test11	0.286363	7484	0.445996	6132	0.301858	5289
Test12	0.643716	6459	0.443219	5775	0.382866	5429
Test13	0.324908	8598	0.485512	5295	0.448285	4865
Test14	0.340318	7500	0.483688	5815	0.248281	5210
Test15	0.426532	8546	0.517793	5392	0.337306	6050
Test16	0.247699	7297	0.258900	6316	0.561932	5591
Test17	0.503263	7507	0.494301	6117	0.452493	6354
Test18	0.343387	7712	0.408345	6115	0.451395	5476
Test19	0.525940	6826	0.441145	6861	0.870105	3744
Test20	0.515371	6146	0.349809	7391	0.446728	6320
Average	0.385568	7423.75	0.440061	6238.25	0.427426	5344.1

Table 18: Delay and the number of received packets for DSDV at speeds 1 m/s, 2 m/s and 3 m/s

References

[1] Per Johansson, Tony Larsson, Nicklas Hedman, Bartosz Mielczarek and Milael Degermark, "Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks", Mobicom '99 Seattle Washington USA.

[2] D. Groten and J.R. Schmidt, "Bluetooth-based Mobile Ad Hoc Networks: Opportunities and Challenges for a Telecommunications Operator", 0-7803-6728-6/01 2001 IEEE.

[3] David Braginsky and Deborah Estrin, "Rumor Routing Algorithm for Sensor Networks", WSNA '02 September 28, 2002, Atlanta Georgia USA.

[4] Nelson Minar, Kwindla Hultman Kramer and Pattie Maes, "Cooperative Mobile Agents for Dynamic Network Routing".
Retrieved on January, 12, 2002 from the World Wide Web:
http://xenia.media.mit.edu/~nelson/research/routes-bookchapter/

[5] Gianni Di Caro and Marco Dorigo, "AntNet: Distributed Stigmergetic Control for Communications Networks", Journal of Artificial Intelligence Research, 9, pp. 317-365, 1998.

[6] MS Thesis of Emre Celebi, 2001 - Performance Evaluation of Wireless Mobile Ad Hoc Network Routing Protocols.
Retrieved on August, 2, 2002 from the World Wide Web:
http://www.cmpe.boun.edu.tr/~emre/research/msthesis/

[7] Kaizar A. Amin and Armin R. Mikler, "Towards Resource Efficient and Scalable Routing: An Agent-based Approach".
Retrieved on August, 26, 2002 from the World Wide Web:
http://students.csci.unt.edu/~amin/publications/advr2/advrPaper2.pdf

[8] Xukai Zou, Byrav Ramamurthy and Spyros Magliveras, "Routing Techiques in Wireless Ad Hoc Networks – Classification and Comparison".
Retrieved on January, 16, 2003 from the World Wide Web:
http://csce.unl.edu/~xkzou/papers/routing_schemes.pdf

[9] The Network Simulator - NS-2.
Retrieved on November, 12, 2001 from the World Wide Web:
http://www.isi.edu/nsnam/ns/

[10] Jae Chung and Mark Claypool, NS by Example.
Retrieved on November, 16, 2001 from the World Wide Web:
http://nile.wpi.edu/NS/

[11] Project: Wireless Ad Hoc networks.
Retrieved on September, 6, 2002 from the World Wide Web:
http://w3.antd.nist.gov/wctg/manet/

[12] Kwindla Hultman Kramer, Nelson Minar, and Pattie Maes, "Tutorial: Mobile Software Agents for Dynamic Routing".
Retrieved on March, 18, 2002 from the World Wide Web:
http://xenia.media.mit.edu/~nelson/research/routes-sigmobile/sigmobile/

[13] Bluetooth Tutorial – Specifications.
Retrieved on June, 11, 2002 from the World Wide Web:
http://www.palowireless.com/infotooth/tutorial.asp

[14] Charles E. Perkins, Mobile Networking Through Mobile IP.
Retrieved on June, 21, 2002 from the World Wide Web:
http://www.computer.org/internet/v2n1/perkins.htm