Design, Implementation and Analysis of Wireless Ad Hoc Messenger

by

Jin-Hee Cho

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in the fulfillment of the requirements for the degree of

Master of Science
in
Computer Science

Committee:
Dr. Ing-Ray Chen, Committee Chair
Dr. Luiz A. DaSilva
Dr. Denis Gracanin

Date: July 29, 2004
Falls Church, Virginia

Keywords: Mobile ad hoc networks, dynamic source routing (DSR), network routing protocols, instant messaging, random waypoint mobility model, wireless communication.

Abstract

Design, Implementation and Analysis of Wireless Ad Hoc Messenger

by

Jin-Hee Cho

Master of Science in Computer Science

Virginia Polytechnic Institute and State University

Dr. Ing-Ray Chen, Committee Chair

Popularity of mobile devices along with the presence of ad hoc networks requiring no infrastructure has contributed to recent advances in the field of mobile computing in ad hoc networks. Mobile ad hoc networks have been mostly utilized in military environments. The recent advances in ad hoc network technology now introduce a new class of applications.

In this thesis, we design, implement and analyze a multi-hop ad hoc messenger application using Pocket PCs and Microsoft .Net Compact Framework. Pocket PCs communicate wirelessly with each other using the IEEE 802.11b technology without the use of an infrastructure. The main protocol implemented in this application is based on Dynamic Source Routing (DSR), which consists of two important mechanisms, Route Discovery and Route Maintenance. We adopt DSR since DSR operates solely based on source routing and "on-demand" process, so each packet does not have to transmit any periodic advertisement packets or routing information. These characteristics are desirable for the ad hoc messenger application for which a conversation is source-initiated on-demand.

To test our application easily, we have developed a testing strategy by which a mobility configuration file is pre-generated describing the mobility pattern of each node generated based on the random waypoint mobility model. A mobility configuration file thus defines topology changes at runtime and is used by all nodes to know whether they can communicate with others in a single-hop or multi-hops during an experimental run.

We use five standard metrics to test the performance of the wireless ad hoc messenger application implemented based on DSR, namely, (1) average latency to find a new route, (2) average latency to deliver a data packet, (3) delivery ratio of data packets, (4) normalized control overhead, and (5) throughput. These metrics test the correctness and efficiency of the wireless

ad hoc messenger application using the DSR protocol in an 802.11 ad hoc network that imposes limitations on bandwidth and resources of each mobile device.

We test the effectiveness of certain design alternatives for implementing the ad hoc messenger application with these five metrics under various topology change conditions by manipulating the speed and pause-time parameters in the random waypoint model. The design alternatives evaluated include (1) Sliding Window Size (SWS) for end-to-end reliable communication control; (2) the use of per-hop acknowledgement packets (called receipt packets) deigned for rapid detection of route errors by intermediate nodes; and (3) the use of cache for path look-up during route discovery and maintenance.

Our analysis results indicate that as the node speed increases, the system performance deteriorates because a higher node speed causes the network topology to change more frequently under the random waypoint mobility model, causing routes to be broken. On the other hand, as the pause time increases, the system performance improves due to a more stable network topology. For the design alternatives evaluated in our wireless ad hoc messenger, we discover that as SWS increases, the system performance also increases until it reaches an optimal SWS value that maximizes the performance due to a balance of a higher level of data parallelism introduced and a higher level of medium contention in 802.11 because of more packets being transmitted simultaneously as SWS increases. Beyond the optimal SWS, the system performance deteriorates as SWS increases because the heavy medium contention effect outweighs the benefit due to data parallelism. We also discover that the use of receipt packets is helpful in a rapidly changing network but is not beneficial in a stable network. There is a break-even point in the frequency of topology changes beyond which the use of receipt packets helps quickly detect route errors in a dynamic network and would improve the system performance. Lastly, the use of cache is rather harmful in a frequently changing network because stale information stored in the cache of a source node may adversely cause more route errors and generate a higher delay for the route discovery process. There exists a break-even point beyond which the use of cache is not beneficial.

Our wireless ad hoc messenger application can be used in a real chatting setting allowing Pocket PC users to chat instantly in 802.11 environments. The design and development of the dynamic topology simulation tool to model movements of nodes and the automatic testing and data collection tool to facilitate input data selection and output data analysis using XML are also a major contribution. The experimental results obtained indicate that there exists an optimal operational setting in the use of SWS, receipt packets and cache, suggesting that the wireless ad hoc messenger should be implemented in an adaptive manner to fine-tune these design

parameters based on the current network condition and performance data monitored to maximize the system performance.

Acknowledgement

Part of the thesis research is funded by Microsoft Research in the project entitled "*Wireless Ad Hoc Messenger*," for which the PI is Dr. Ing-Ray Chen and Co-PIs are Dr. Scott Midkiff, Dr. Luiz DaSilva, Dr. Denis Gracanin and Dr. Shawn Bohner.

I would like to thank my advisor, Dr. Ing-Ray Chen, for giving me great learning experiences. I could not finish this thesis without his help. I fully appreciate his time, patience, and enthusiasm for this work.

In addition, I thank Dr. Gracanin and Dr. DaSilva for serving on the thesis committee and providing very helpful comments to improve this thesis.

This thesis is dedicated to my parents and my husband. I thank my mother for her patience and understanding that I chose this way of my life. Especially, I thank my husband, Chung, for his endless support and love.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

Mobile devices, such as laptop computers, Pocket PCs, cellular phones, etc., are now easily affordable, and are becoming more popular in everyday life [Johnson1994, Johnson1996]. At the same time, network connectivity options for mobile hosts have grown tremendously, as the support for wireless networking products based on radio and infrared has been greatly increased over the past few years.

With the availability of mobile computing devices, mobile users have a natural tendency to share information between them. Often mobile users want to have a meeting, even though it is not planned in advance and there is no Internet connection available. For instance, there may be situations that employees find themselves together in a meeting room, or friends or business acquaintances may encounter each other in an airport terminal, or some scholars and researchers may meet in a hotel ballroom for a conference or workshop. In those situations, requiring each user to connect to a wide-area network to communicate with each other may not be convenient or practical because of the lack of Internet connectivity or because of the time or cost required for such a connection.

This thesis aims to design, implement and evaluate an instant messaging application for mobile users in these situations without requiring the users to connect to the Internet. A network of mobile hosts without an infrastructure is known as an *ad hoc network* [Johnson1996]. According to Johnson [Johnson1994], an ad hoc network is defined as follows:

> "An *ad hoc network* is a collection of wireless mobile hosts forming a temporary network without the aid of any centralized administration or standard support services regularly available on the wide-area network to which the hosts may normally be connected."

In general, a multi-hop routing protocol is needed in a mobile ad hoc network, because two hosts wishing to exchange packets may not be able to communicate directly with each other because they are out of radio range [Johnson1994]. Johnson illustrated a simple ad hoc network of three mobile hosts using wireless network interfaces, as shown in Figure 1.1 [Johnson1994]. Host **C** is not included within the wireless transmission range of host **A**, as indicated by the circle around **A**. Also, host **A** is not within the wireless transmission range of host **C**. If **A** and **C** want to communicate with each other by exchanging packets, they may ask

host **B** to forward packets for them because host **B** is within the overlapped wireless transmission range between host **A** and host **C**.

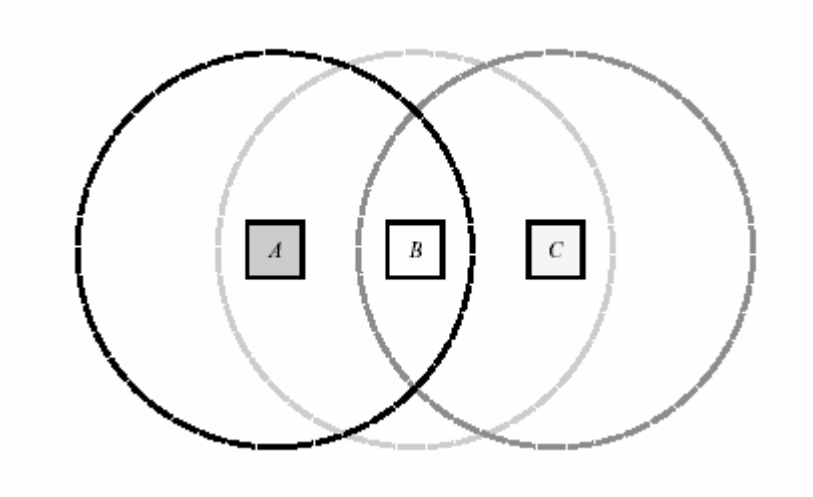

Figure 1.1: An Ad Hoc Network with Three Wireless Mobile Hosts.

In any practical ad hoc network, the maximum number of network hops for a packet to travel from one mobile host to another mobile host may be small but is likely to be greater than one as demonstrated in Figure 1.1. In a real ad hoc network, the routing problem may be even more complex than this example shown, because wireless transmission has inherent non-uniform propagation features and any or all of the hosts associated with the network may move at any time [Johnson1994].

1.1 Mobile Ad Hoc Routing Protocols

A conventional way of providing routing in an ad hoc network is to make each mobile host take the role as a router, and apply an existing routing protocol between them [Jubin1987, Perkins1994]. This Section briefly summarizes existing routing protocols devised for mobile ad hoc networks and the reason why we adopt DSR as the routing protocol for our implementation of the instant messaging application.

1.1.1 Reactive vs. Proactive Ad Hoc Routing Protocols

Most routing protocols in mobile ad hoc networks derive from distance vector or link state algorithms.

In *distance-vector routing*, each router maintains a table containing the distance from itself to all possible destinations. Each router periodically transmits this table information to all its neighbor routers, and updates its own table by using the values received from its neighbors. Based on the comparison of the distances obtained from its neighbors for each destination, a router can decide the next hop as the shortest path from itself to the specified destination. When each router has a packet to send to some destination, it simply forwards the packet to the decided next hop router. When the routing table is frequently updated, the algorithm speeds up the convergence to the correct path. However, the overhead in CPU time and network bandwidth for flooding routing updates also increases. Perkins and Bhagwat [Perkins1994] devised a *Destination-Sequenced Distance Vector (DSDV)* protocol based on the classical *Bellman-Ford* routing algorithm to apply to mobile ad hoc networks. DSDV also has the feature of the distance-vector protocol in that each node holds a routing table including the next-hop information for each possible destination. Each entry has a sequence number. If a new entry is obtained, the protocol prefers to select the entry having the largest sequence number. If their sequence number is the same, the protocol selects the metric with the lowest value. Each node transmits advertisement packets using increasing sequence numbers [Perkins1994]. A study of performance evaluation on DSDV [Broch1998] shows that DSDV is able to deliver virtually all data packets when each node moves with relatively low speed. However, when the mobility of each node increases, the speed at which the system converges to the correct path decreases [Broch1998].

While DSDV is a proactive protocol that always tries to maintain the correct information regarding network topology, *Ad hoc On-demand Distance Vector (AODV)* [Perkins1999] is a reactive protocol to perform Route Discovery only when a new route needs to be found. Thus, AODV does not maintain any routing information nor transmit any periodic advertisement packets for exchanging routing tables. That is, only when two nodes need to communicate with each other, will they forward routing packets to maintain connectivity between the two nodes. Usually, when there is a need for communication between two nodes, each mobile node transmits a local broadcast packet known as a *hello* message. Routing tables of the nodes within the neighborhood are organized for the optimization of response time to local movements and the support of rapid response time for requests to establish a new route. AODV is similar with the Dynamic Source Routing (DSR) protocol, which will be explained with more details in the following sections, in terms of the nature of *on-demand*. However, while DSR is based on source routing, AODV is dependent upon dynamically establishing route table entries at intermediate nodes.

In *link state routing*, each router has a complete picture of the whole network topology. Each router checks the cost of the link to each of its neighbor routers, and floods periodically updated information to all other routers in the network. After each router receives this updated information of the cost of each link in the network, each router calculates the shortest path to each possible destination. When each router needs to forward a packet to some destination, it transmits the packet to the next hop router based on the best path to the destination acquired from the updated information. The link state routing protocols show high speed of convergence to the correct network picture when the network is changed. However, in general, compared to the distance vector algorithm, this protocol requires more CPU time for computing the complete shortest route to each possible destination, and more network bandwidth for broadcasting the routing updates from each router to all other routers in the whole network. In order to reduce the disadvantage of the link state routing algorithm, one optimized protocol is proposed, called an *Optimized Link State Routing (OLSR)* [Jacquet2001].

OLSR is one of protocols using link state routing with a proactive nature, which allows periodic exchange of knowledge of network topology among all the nodes of the network. Because of this proactive nature in OLSR, it has a benefit of knowing the routes immediately when needed. A pure link state routing protocol has the characteristic that all the links with neighbor nodes are declared and are broadcast to the whole network. However, since OLSR is the optimization of the link state routing protocol for an ad hoc network, it uses the reduced size of control packets, declaring only a subset of links with its neighbors who are its *multipoint relay selectors*, instead of all links. Besides, OLSR minimizes flooding of this control packet by using only the selected nodes, called *multipoint relay*, to disperse its messages in the network. The multipoint relay is the idea to diminish the flooding of broadcast packets in the network by lessening duplicate retransmission in the same region.

In addition to the viewpoint categorizing routing protocols in terms of either distance vector or link state routing, routing protocols for ad hoc networks also can be classified as reactive routing protocols versus proactive routing protocols, as mentioned previously [Jacquet2001]. In the reactive routing approach, a routing protocol does not initiate Route Discovery until it is needed. The protocols only attempt to find a route to a destination totally based on *demand*. Examples of reactive routing protocols for ad hoc networks include AODV [Perkins1999], DSR [Johnson2001, Johnson1999, Maltz2001], ODMRP [Lee2000], and TORA [Park1997]. On the contrary, the proactive routing approach is based on the exchange of knowledge of network topology periodically [Jacquet2001]. The proactive protocols provide a needed route instantly at the expense of bandwidth because of transmitting periodic updates of

topology frequently. Examples of proactive routing protocols include DSDV [Perkins1994], STAR [Garcia-Luna-Aceves1999], and TBRPF [Orgier2004], etc.

1.1.2 Adopting DSR as the Routing Protocol for the Ad Hoc Messenger Application

In the previous section, we gave an overview of reactive and proactive protocols in mobile ad hoc networks based on distance vector or link state routing. In this Section, we list some known problems [Johnson 1994] associated with these protocols and give reasons why we adopt DSR as the protocol for the implementation of our *Ad Hoc Messenger* application.

First, sending and receiving between two hosts over a wireless network does not necessarily work equally well in both directions. In Figure 1.1, host **A** may receive routing information from **B** reporting that **B** is the closest node to **C** and thus would be the next hop for **A** to obtain the shortest path to **C**. However, host **A** may not be able to transmit a packet back to **B**. Figure 1.1 indicates that the transmission range of all hosts is the same and uniform on all sides of the host, but radio and infrared propagation does not always work that nicely in real situations. Therefore, routes discovered by distance vector or link state routing based protocols may not work in wireless environments.

Second, most links between routers provided by the routing protocols may be redundant. In Figure 1.1, assume that there are many mobile hosts within the transmission range of host **A**. Then all intermediate hosts including host **B** are equally good as an intermediate router for transmitting packets of host **A** to host **C**. These redundant routes in a wireless network cause a high volume of route updates flooded over the network, and cause high CPU overhead to process each routing update and to calculate new routes.

Third, network bandwidth is often wasted in forwarding routing updates periodically. Under a stable network that does not change the topology often, even though nothing changes from one routing update to the next, each router still has to transmit periodic updates. Routing updates from mobile hosts outside each other's transmission range will not give any interference to each other. However, when there are many mobile hosts within the transmission range of each other, their routing updates will consume each other's network bandwidth.

Forth, battery power is wasted because of periodically transmitting routing updates. Most mobile hosts in an ad hoc network will be operated based on battery power. Thus, transmitting each packet uses a considerable amount of battery power. Even though generally receiving a packet requires less consumption of battery power than sending it, the periodically

arriving routing updates effectively prevent a host from conserving its own battery power by changing its mode to “sleep” or “standby” when not busy.

Finally, distance vector or link state routing based protocols may not be suitable for dynamically changing networks that may often occur in ad hoc networks. In ad hoc networks, since movements of mobile hosts are common, using distance vector or link state routing based protocols may bring slow convergence to new and stable routes. The speed of convergence may be improved by flooding routing updates more frequently. However, such a modification wastes more bandwidth and battery power under a less frequently changing network topology.

Some protocols have been purposed to improve the disadvantages of the conventional protocols, such as OLSR, as we introduced previously. However, many of the problems still remain unsolved.

We adopt the Dynamic Source Routing (DSR) protocol [Johnson1994, Johnson1996, Johnson2001, Maltz2001, Broch1998, Zhong2003, Leung2001, Raju2001, Wenjing2002, Maltz1999, Camp2002, Jiang2002, Wu2002, Seet2003, Hu2000] for our implementation of the wireless ad hoc messenger application. The main reason is that DSR is a reactive on-demand protocol initiated by a source node to send information to a destination node without the need to maintain routing table information, thus eliminating many of the problems associated with distance vector or link state routing based protocols. Moreover, our wireless ad hoc messenger application also has the property that a chatting session is initiated by a user on-demand and that nodes move without a pattern so asking nodes to maintain routing table information is not practical.

Another reason is that while some previous research work has been done to study the performance of DSR, most researches just use simulation experiments (e.g., based on the *ns*-2 network simulator). We aim to develop a real-world application using DSR as the underlying routing protocol and to test the effectiveness of certain design alternatives on the performance of the application.

1.2 Purpose of Study

As mentioned in the beginning of the introduction Section, people with mobile devices, such as a Pocket PC, often have the desire to share information when they are together unexpectedly and there is no Internet connection available. For example, users carrying Pocket PCs may want to exchange information during the middle of a conference or desire to share their ideas while listening to others’ presentations via Instant Messaging without using any wireless

infrastructure. In order to achieve the situation that each user carrying a Pocket PC can communicate with another by sending instant messages without the aid of any infrastructure, we have designed, implemented and evaluated an ad hoc messenger using Pocket PCs in wireless ad hoc environments in this thesis.

The purposes of this thesis are as follows:

- To design and implement a real-world chatting prototype for Pocket PCs as a multi-hop ad hoc instant messenger, using IEEE 802.11b wireless cards, Microsoft .Net Compact Framework (C#), and XML technology.
- To implement the DSR protocol on a real-world multi-hop ad hoc chatting application using Pocket PCs in terms of its main two mechanisms, Route Discovery and Route Maintenance.
- To utilize XML technology for description and representation of data and control packets and experimental results to facilitate data exchange between nodes.
- To design and implement a dynamic topology simulation tool to ease testing and evaluation of our wireless ad hoc messenger application through generating a "mobility" configuration file based on the random waypoint mobility model to be followed by all nodes.
- To design and implement an automatic data collection tool to ease performance data collection including the tasks of automatic transmission of data packets, automatic generation of result files, and collection of result files through wireless transmission.
- To test and analyze design alternatives in implementing the wireless ad hoc messenger and investigate the effect of these design alternatives on the performance of the application.

1.3 Contributions

The contributions of the thesis are as follows:

- As many users are carrying various types of mobile devices, such as laptops, Pocket PCs, cell phones, etc., there are increasing needs to communicate with each other without any aid of infrastructure. The application developed and evaluated in this thesis takes a role as an initiative to realize the popular use of Pocket PCs for exchanging information under environments with no infrastructure or Internet connections.
- While other studies on the DSR protocol have usually evaluated its performance using a simulator such as the *ns*-2 network simulator, this thesis designs and implements a real world chat program, a wireless ad hoc instant messenger application, using Pocket PCs, IEEE

802.11b wireless technology, Microsoft .Net Compact Framework (C#), and XML technology. It is valuable that a real ad hoc chat application using Pocket PCs has been properly designed, implemented and evaluated based on the DSR protocol. This work deserves some attention in that the performance of the DSR protocol is properly evaluated by developing a real world mobile ad hoc chat application.

- An innovative evaluation environment has been designed and implemented to ease the analysis of the effect of topology changes and design alternatives, and to ease performance data collection, including the design to generate network configuration files based on the random waypoint mobility model and the design to automatically generate test data and collect performance data based on XML data format and representation.
- Several design alternatives have been implemented and evaluated, and conditions under which one design alternative should be adopted over others are identified to improve the performance of the wireless ad hoc instant messenger application.

1.4 Structure of Thesis

This thesis consists of five chapters. The rest of the thesis is organized as follows.

Chapter 2 describes background and related work. In particular, knowledge backgrounds for readers to easily understand the analysis regarding the results of experiments are discussed, including the characteristics of wireless ad hoc networks (Section 2.1), the main features of the DSR protocol (Section 2.2), and the IEEE 802.11 MAC protocol (Section 2.3). In Section 2.4, related work is introduced in terms of evaluation of the DSR protocol and testing of the optimized features of DSR. Section 2.4 also summarizes Chapter 2.

In Chapter 3, detailed methodologies of this study are described, such as environment of the application development (Section 3.1), application system design (Section 3.2), performance evaluation (Section 3.3) in terms of input parameters and metrics used, and XML files used for describing and representing the results of experiments (Section 3.4). Then, Section 3.5 gives a summary of Chapter 3.

Chapter 4 presents the experimental results based on the performance metrics used, and gives the interpretations of the results obtained. Section 4.1 describes the default conditions for

input variables. From Section 4.2 to Section 4.6, the results of experiments under a variety of scenarios are presented and discussed.

Finally, Chapter 5 concludes the thesis and outlines future work. Section 5.1 summarizes the results reported in Chapter 4. Section 5.2 describes the achievements of this thesis work. Section 5.3 suggests future work in terms of possible improvements of design and implementation of DSR on Pocket PCs in an ad hoc network.

The bibliography includes all references used in this work. The appendices provide additional information that readers can refer if needed, such as user interfaces for automatic data collection, and complete XML files used to generate results.

Chapter 2
Background and Related Work

This Chapter describes relevant background knowledge and related work for readers to easily understand the analysis conducted in our experiments to be presented and discussed in Chapter 4.

The first three sections will discuss the relevant background knowledge. In Section 2.1, wireless ad hoc networks and their significant characteristics are discussed. Section 2.2 describes key features of the DSR protocol, in terms of its two main mechanisms, *Route Discovery* and *Route Maintenance*. Section 2.3 gives a brief review of the IEEE 802.11 MAC protocol. Since Pocket PCs in our system communicate with each other using Compaq® IEEE 802.11 wireless LAN cards, knowing main characteristics of the IEEE 802.11 MAC protocol will help readers to understand the experimental results reported in Chapter 4. Section 2.4 addresses other related works, primarily focusing on the performance of DSR in ad hoc networks. Finally, Section 2.5 provides a summary of this Chapter.

2.1 Wireless Ad Hoc Networks

In recent years, wireless technologies and applications have received a lot of attention [Sheu2002]. Owing to rapidly emerging wireless computing, users can communicate with each other through network connectivity without being tethered off of a wired network [Crow1997]. An ad hoc wireless local area network (WLAN) consists of a group of mobile hosts creating a temporary network without the help of any pre-existed infrastructure or centralized administration [Sheu2002, Johnson1994, Frodigh2000, Perkins2000]. Since the nodes in an ad hoc network are able to serve as routers and hosts, they can forward packets on behalf of other nodes and run user applications [Frodigh2000]. That is, when two nodes are within the wireless transmission range of each other, they can communicate with each other directly. However, if they are out of their transmission range to each other, intermediate nodes can forward a packet for the two nodes to communicate with. Since ad hoc networks do not need any help of centralized infrastructure, wireless applications are becoming more and more popular where wired networking is not available or not economically feasible [Sheu2002]. In general, a wireless ad hoc network that consists of mobile nodes and communicates over radio without the aid of any infrastructure is popularly called MANET (Mobile Ad-hoc NETwork) [Gunes2002].

Figure 2.1 [Frodigh2000] depicts our vision of ad hoc networking including scenarios that may occur at an airport where people carry devices that can be connected based on an ad hoc network. A user's devices can both interconnect with one another and connect to local information point, for instance, in order to obtain updated information on flight departures, gate changes, and so on. The ad hoc devices can also forward traffic on behalf of two devices that are out of range each other. Thus, the below airport scenario includes a mixture of single and multiple radio hops.

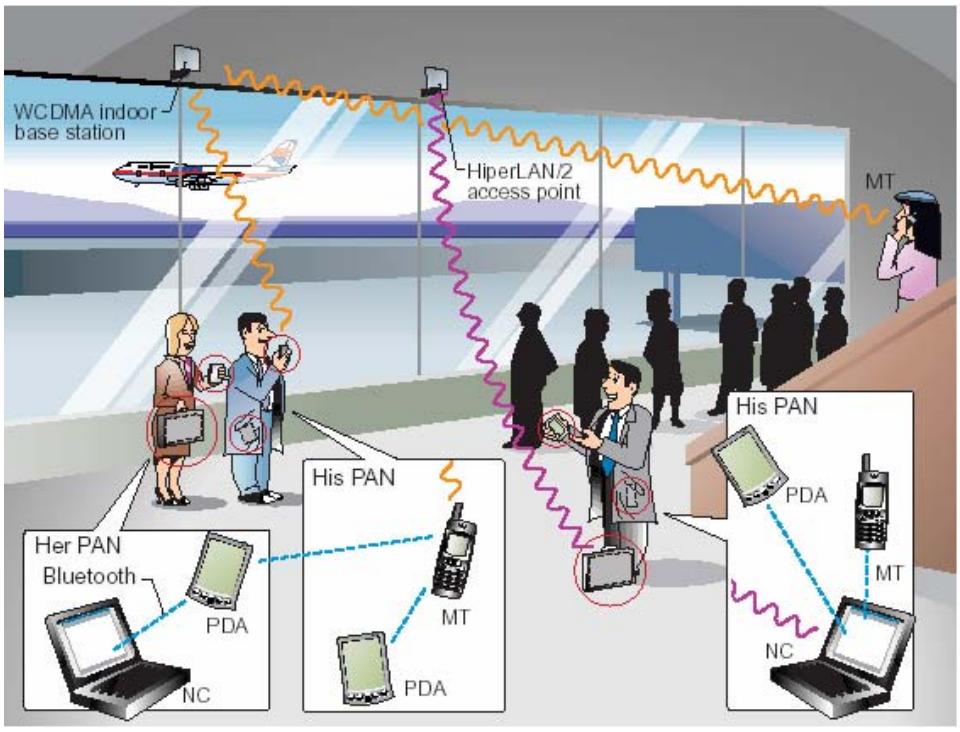

Figure 2.1: Example of an Ad Hoc Network: Airport Scenario.

Figure 2.1 describes the following scenario that can often occur in an airport. At an airport, where people can access local-and wide-area networks, ad hoc Bluetooth connections are used to interconnect carried devices, such as PDAs, WCDMA mobile phones, and notebook computers. For instance, a user might retrieve email via a HiperLAN/2 interface to a notebook computer in a briefcase, but read messages and reply to them via his or her PDA [Frodigh2000].

2.1.1 Characteristics of Wireless Mobile Ad Hoc Networks

Wireless mobile ad hoc networks have significant characteristics as follows [Hekmat2004, Corson1999, Frodigh2000]:

1) Dynamic Network Topology

Each node in an ad hoc network is free to move randomly. This feature makes the network topology change unpredictably. Also, an ad hoc network may be comprised of both bi-directional and unidirectional links [Corson1999]. Thus, using ad hoc networks could augment mobility and flexibility of nodes in the network [Frodigh2000]. Even though the network topology varies, connectivity in the network should be maintained to allow applications and services to operate without disruption. In particular, this characteristic will affect the design of routing protocols. In addition, a user in an ad hoc network will require access to a fixed network, such as the Internet, even if nodes are mobile. This needs mobility management functions allowing network access for devices located several radio hops away from a network access point [Hekmat2004].

2) Bandwidth-Limited and Fluctuating Capacity Links

Wireless links will remain to have substantially lower capacity compared to their hardwired counterparts [Corson1999]. Besides, the throughput of wireless communications in real environments is often much less than a radio's maximum transmission rate, because there may be the effects of multiple access, fading, noise, and interference conditions, and so on [Corson1999].

The effects of high bit-error rates may be more severe in a multi-hop ad hoc network, because the aggregate of all link errors affects a multi-hop path. Moreover, more than one end-to-end route can use a given link if the link were to break. This could disrupt several sessions during periods of high bit-error transmission rates. Thus, this will affect the routing function. However, efficient functions for link layer protection, such as forward error correction (FEC), and automatic repeat request (ARQ), can significantly improve the link quality [Hekmat2004].

3) Low-Power and Resource-Limited Operation

In most cases, the network nodes in a wireless ad hoc network may depend on batteries or other exhaustible means for their energy [Corson1999]. This feature makes the power budget tight for all the power-consuming components in a mobile device. For example, this will affect CPU processing, memory size and usage, signal processing, and transceiver output/input power [Hekmat2004]. For these nodes, energy conservation should be considered for the optimization as a key system design criterion [Corson1999].

4) Constrained Physical Security

In general, mobile wireless networks are more likely to be vulnerable to physical security threats than are fixed-cable nets. For example, there are the increased possibility of eavesdropping, spoofing, and denial-of-service attack that should be carefully considered. Often current link security techniques are applied to wireless networks to diminish security threats [Corson1999, Zhou1999].

5) Decentralized Network Control

As an advantage, the decentralized nature of network control in mobile ad hoc networks supports extra robustness against the single points of failure of more centralized approaches [Corson1999, Zhou1999].

2.2 Description of the DSR Protocol

This Section gives a brief overview of the Dynamic Source Routing (DSR) protocol, in terms of its main two mechanisms, namely, *Route Discovery* and *Route Maintenance*.

2.2.1 Properties of the DSR Protocol

The Dynamic Source Routing (DSR) protocol [Johnson2001, Johnson1996, Maltz2001, Broch1998] is a simple and efficient routing protocol designed particularly for utilization in multi-hop wireless ad hoc networks of mobile nodes. The network using DSR is entirely self-organizing and self-configuring, and it does not require existing network infrastructure or administration, as mentioned in the previous sections. Network nodes cooperate with each other to transmit packets for communication through multi-hops between nodes, which are not directly located within the wireless transmission range of one another.

The DSR protocol [Johnson2001, Johnson1996, Maltz2001, Broch1998] is based on source routing. Source routing means that a source of each packet has an ordered list of nodes to reach a destination node. The main advantage of a source routing design is that intermediate nodes do not have to contain the latest routing information in order to forward packets because the packet's source has already decided all of the routing information. Thus, intermediate nodes just forward the packets to the next node based on the route contained in the packet that is forwarded. Because of this characteristic, associated with the completely on-demand nature of the DSR protocol, each node does not have to transmit any type of periodic route advertisement or neighbor detection packets.

The DSR protocol [Johnson2001, Johnson1996, Maltz2001, Broch1998] consists of two main mechanisms that allow the discovery and maintenance of source routes in ad hoc networks. They are *Route Discovery* and *Route Maintenance*. *Route Discovery* is the mechanism by which a node **S** trying to send a packet to a destination node **D** acquires a route from the source node **S** to the destination node **D**. Route Discovery is performed only when **S** wishes to transmit a packet to **D** but the source **S** does not know any route to **D**. *Route Maintenance* is the mechanism by which the source node **S** detects a link failure as it sends a packet to the destination node **D** and repairs or finds a new route to maintain the connectivity. Specifically, when **S** sends a packet to **D**, the network topology may have been changed and the source route to **D** is not valid any longer. Then **S** can try to utilize alternative routes it knows to **D**. However, if **S** does not know any other route, it performs Route Discovery again to find a new route. Route Maintenance is invoked while **S** is sending data packets to **D** after Route Discovery.

Route Discovery and Route Maintenance are respectively performed totally *on demand*. In particular, different from any other protocols, DSR does not require periodic packets of any kind at any level within the network. For instance, DSR does not send any periodic routing advertisement, link status sensing, or neighbor detection packets, and does not depend on these functions from any basic protocols in the network. Due to this on-demand behavior and no use of periodic advertisement packets, using fewer overhead packets enables DSR to be scalable smaller, especially when all nodes are almost stationary with respect to each other and thus all routes needed for the current interaction have already been found. As the network topology begins to change or communication patterns change, the routing packet overhead of DSR automatically focuses only on the need to track the routes currently in use [Johnson2001, Maltz2001].

After Route Discovery is performed, a source node may learn and cache multiple routes to any destination. This feature gives more benefits in avoiding the overhead of needing to perform a new Route Discovery every time when a route in use is broken. Figure 2.2 shows the basic functions of the DSR protocol [Maltz2001].

Maltz explains the construction of a source route by broadcasting a ROUTE REQUEST in his paper, as Figure 2.2 shows [Maltz2001]. The source receives a ROUTE REPLY containing the route found that can be followed to forward data packets. On the other hand, a ROUTE ERROR is sent to the source upon the detection of route failure. The address in parentheses indicates the next hop to forward a packet [Maltz2001].

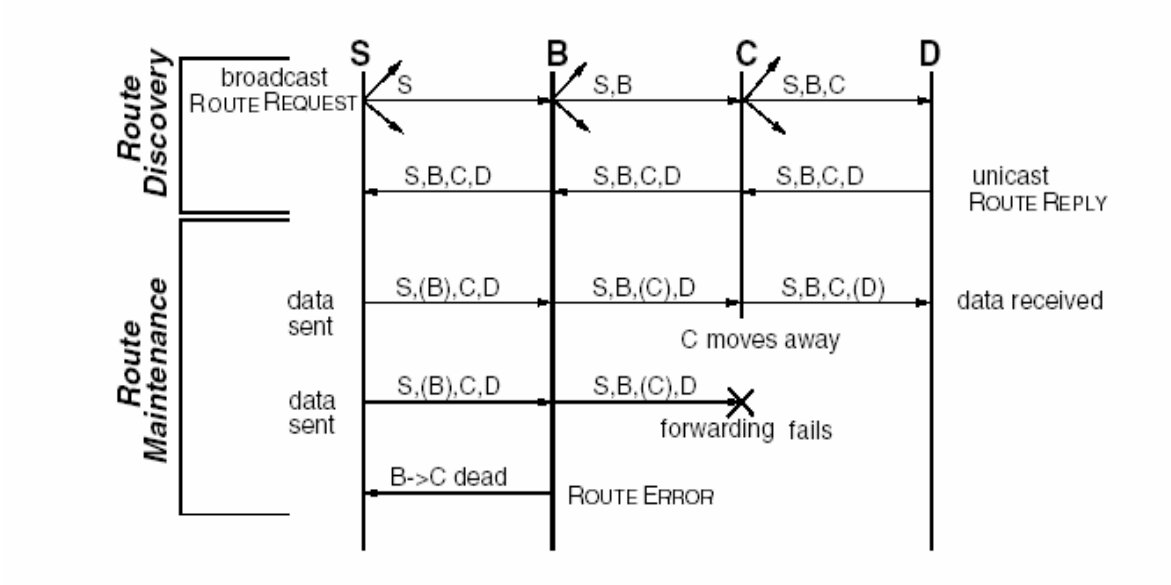

Figure 2.2: Overall Basic Operation of the DSR Protocol.

In the next subsection we describe basic features of DSR Route Discovery and Route Maintenance [Johnson2001, Johnson1996, Maltz2001]. Then, the following subsections will delve more into additional features of the DSR protocol, in terms of the optimized features of Route Discovery and Route Maintenance [Johnson2001, Johnson1996, Maltz2001].

2.2.2 Basic DSR Route Discovery

When node **S** initiates to transmit a new packet to some other node **D**, it stores a source route indicating the sequence of hops that the packet should be forwarded to be delivered to **D** in the header of the packet [Johnson2001]. Usually, **S** will acquire a proper source route by looking at its *Route Cache* of routes obtained in the past. However, if there is no route available in its cache, it will perform the Route Discovery protocol to dynamically find a new route to **D**. In this case, **S** is called the *initiator* and **D** is called the *target* of the Route Discovery.

For instance, Figure 2.3 [Johnson2001] gives an illustration of an example of the Route Discovery, where node **A** is trying to discover a route to node **E**. To perform the Route Discovery, **A** floods a ROUTE REQUEST message as a single local broadcast packet to all nodes currently within the wireless transmission range of node **A** [Johnson2001]. Each ROUTE REQUEST message finds the initiator and target of the Route Discovery, and also includes a unique *request id*, given by the initiator of the ROUTE REQUEST.

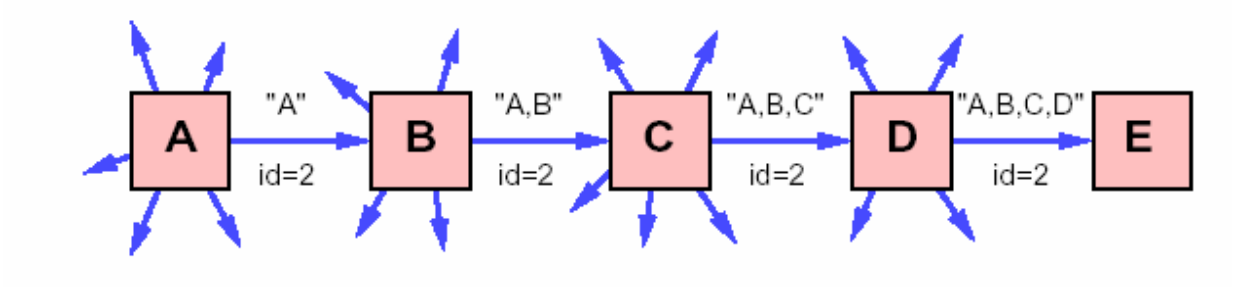

Figure 2.3: Route Discovery Example: Node A is the initiator, and node E is the target.

Each ROUTE REQUEST also holds a record indicating the list of the address of each intermediate node through which this particular copy of the ROUTE REQUEST message has been forwarded [Johnson2001]. This route record is started from an empty list when the initiator starts to perform the Route Discovery. When another node receives a ROUTE REQUEST and if the node is the target of the Route Discovery, it sends a ROUTE REPLY back to the initiator of the Route Discovery, containing a copy of the accumulated route record from the ROUTE REQUEST [Johnson2001]. When the initiator receives the ROUTE REPLY, it stores this route in its Route Cache for utilization in transmitting succeeding packets to this destination. Otherwise, if this node receiving the ROUTE REQUEST has lately seen another ROUTE REQUEST message containing this same *request id* from the same initiator, or if it recognizes that its own address is already contained in the route record in the ROUTE REQUEST message, it ignores the REQUEST [Johnson2001]. Otherwise, this node appends its own address to the route record in the ROUTE REQUEST message and floods it to its neighbors, which are within the transmission range of this node, as a local broadcast packet with the same *request id* [Johnson2001].

When the destination node returns the ROUTE REPLY to the initiator of the Route Discovery, the destination node will usually search its own Route Cache for a route back to the initiator. If the route is available in its Route Cache, the destination node will use it for the source route to return the ROUTE REPLY. Otherwise, the destination node **D** may perform its own Route Discovery for the target node **A**. However, in order to avoid possible infinite recursion of Route Discoveries, it must piggyback this ROUTE REPLY on its own ROUTE REQUEST message for the initiator. The destination node can also simply use the reverse route of the route record in the ROUTE REQUEST. For the MAC protocols such as IEEE 802.11 requiring a bi-directional frame exchange as a part of the MAC protocol, this route reversal is preferred since it prevents the overhead of a possible second Route Discovery, and it tests the

found route to make sure that it is bi-directional before the Route Discovery initiator starts using the route [Johnson2001].

When the initiator performs a Route Discovery, it saves a copy of the original packet in a local buffer called the *Send Buffer*. The Send Buffer includes a copy of each packet that cannot be sent by this node because the initiator does not have a source route to the destination of the packet yet. Each packet in the Send Buffer is stamped with the time that it was stored into the Buffer and is discarded after placing in the Send Buffer for some timeout period [Johnson2001]. In order to prevent the Send Buffer from overflowing, a FIFO or other replacement strategy may be used to evict packets before the timeout period expires. While a packet resides in the Send Buffer, the initiator should initiate a new Route Discovery for the packet's destination address. However, the initiator should limit the number of initiations of Route Discovery to the same destination because the destination node may not be reachable at the moment [Johnson2001]. Particularly, the limited wireless transmission range and mobility of the nodes may cause disconnection between the initiator and the destination. Due to the movement patterns and the density of nodes in the network, network partitions may be rare or may be common [Johnson2001]. If a new Route Discovery were performed for each packet transmitted by the initiator, a large number of unproductive Route Request packets would be flooded throughout the subset of the ad hoc network within the transmission range of the node. In order to reduce the overhead from such Route Discoveries, exponential backoff can be used to limit the rate of performing new Route Discoveries that may be initiated by any node for the same target. If the node tries to send additional data packets to this same node more frequently than this limit, the following packets should be buffered in the Send Buffer until a Route Reply is received, but the initiator should not perform a new Route Discovery until the minimum allowable interval between new Route Discoveries for this destination has been elapsed [Johnson2001].

2.2.3 Basic DSR Route Maintenance

When a node transmits or forwards a packet to some destination node, Route Maintenance is used to detect if there was any change in the network topology that has caused the route used by the packet to be broken [Maltz2001]. Thus, when a node initiates to send a packet using a source route, each node forwarding the packet has responsibility to confirm that the packet has been received by the next hop along the source route [Johnson2001]. The data packet can be retransmitted up to a maximum number of attempts until this confirmation of receipt is received in the previous node [Johnson2001]. For instance, as illustrated in Figure 2.4, node **A**

has initiated a packet for **E** using a source route through intermediate nodes **B**, **C**, and **D**. In this situation, node **A** has the responsibility to confirm the receipt of the packet at node **B**, node **B** is responsible for the receipt of the packet at node **C**, node **C** is responsible for the receipt at node **D**, and node **D** is responsible for receipt at the target **E** finally.

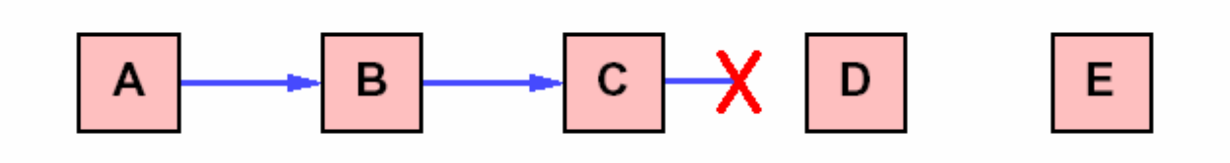

Figure 2.4: Route Maintenance Example: Node C is unable to forward a packet from A to E over its link to next hop D.

In many wireless MAC protocols, such as IEEE 802.11 [Brenner1997], the MAC protocol retransmits each packet until a link-layer acknowledgement is received, or until a maximum number of transmission trials has been made. As an alternative, DSR may utilize a *passive acknowledgement* or may request an explicit network-layer acknowledgement [Maltz2001].

If the packet is retransmitted by some hop with the maximum number of times and no receipt confirmation is received, this intermediate node returns a ROUTE ERROR message to the original initiator of the packet [Johnson2001]. When the initiator receives the ROUTE ERROR message, it removes this broken route from its cache. In order to retransmit the original packet, if the initiator has another route in its Route Cache (that stores alternative routes obtained from additional ROUTE REPLYs for its previous Route Discoveries), it can retransmit the packet using the new route without delay. Otherwise, it may initiate a new Route Discovery for this destination node [Johnson2001].

2.2.4 Optimizations of the DSR Protocol

This subsection will explain additional features of Route Discovery and Route Maintenance of DSR to optimize its performance. The current implementation of our application is mostly based on basic operations of DSR; the advanced features discussed here due to the optimized DSR protocol will be explored in the future work.

2.2.4.1 Additional Route Discovery Features

This Section introduces four optimization features related with the Route Discovery phase in DSR as follows: caching overheard routing information, replying to ROUT REQUESTs using cached routes, preventing route reply storms, and ROUTE REQUEST hop limits.

1) Caching Overheard Routing Information

If a node, which is neither a source nor a target, forwards or overhears any packet, the node may add the routing information from the packet to its own Route Cache. That is, the source route used in sending a data packet, the accumulated route in the route record of a ROUTE REQUEST, or the route contained in a ROUTE REPLY may all be cached by any node [Johnson2001].

2) Replying to ROUTE REQUESTs using Cached Routes

A node receiving a ROUTE REQUEST, which is not a target but an intermediate node, searches its own Route Cache for a route to the target of the ROUTE REQUEST [Johnson2001]. If a route is available in its Route Cache, the node usually returns a ROUTE REPLY to the initiator of the ROUTE REQUEST, rather than forwarding the ROUTE REQUEST to the nodes within the transmission range of the node. In the ROUTE REPLY, it sets the route record to the combined route gained by concatenating the accumulated route record of the current ROUTE REQUEST with the available route to the target found in the Route Cache of the intermediate node [Johnson2001]. This feature may shorten the latency for Route Discovery process.

3) Preventing Route Reply Storms

In some cases, ROUTE REPLY storms may occur [Johnson2001]. For example, when a node initiates a ROUTE REQUEST to its neighbors, who are located within the wireless transmission range of the node, neighboring nodes having ability to reply to the ROUTE REQUEST based on information in their Route Caches may attempt to reply by transmitting ROUTE REPLYs to the initiator of the ROUTE REQUEST. If each of those nodes has a route to reach the target of the ROUTE REQUEST, it will send a ROUTE REPLY to the initiator of the REQUEST. In that situation, large number of ROUTE REPLYs may be received by the initiator, thereby wasting bandwidth and possibly increasing the number of network collisions in the area [Johnson2001, Maltz2001]. Besides, since some nodes may be closer to other nodes, there will be different ROUTE REPLYs containing routes of different lengths. If nodes are able to promiscuously listen to the channel, they can lessen the number of ROUTE REPLYs transmitted to the initiator of the ROUTE REQUEST by deferring the transmission of ROUTE

REPLYs for a random period of time based on the length of the source route in their ROUTE REPLYs. If a node delaying the transmission of ROUTE REPLY hears that the initiator of the ROUTE REQUEST uses a shorter source route to the target than the one it is deferring, the node does not send the ROUTE REPLY because the initiator already has a shorter route to the target [Johnson2001, Maltz2001].

4) ROUTE REQUEST Hop Limits

In order to limit the number of intermediate nodes to forward the copy of the ROUTE REQUEST, each ROUTE REQUEST message contains a *hop limit*. As the ROUTE REQUEST is forwarded, this limit is decremented by one, and the REQUEST packet is discarded if the limit reaches zero before finding the destination [Johnson2001]. As an example, at first, a *non-propagating* ROUTE REQUEST with hop limit zero is transmitted. This mechanism is inexpensive method to decide if the destination is currently a neighbor of the initiator or if a neighbor node has a cached route to the destination, effectively utilizing the caches of the neighbor nodes as an extension of the initiator's own cache. If no ROUTE REPLY is received within a short timeout, then a propagating ROUTE REQUEST with no hop limit is transmitted [Johnson2001].

2.2.4.2 Additional Route Maintenance Features

This Section provides four optimization features associated with Route Maintenance in the DSR protocol as follows: packet salvaging, automatic route shortening, increased spreading of ROUTE ERROR messages, and caching negative information.

1) Packet Salvaging

When the route used in sending a data packet is broken, a ROUTE ERROR message is sent to a source node, as a part of Route Maintenance mechanism [Johnson2001]. Then, the intermediate node detecting the broken route may attempt to salvage the data packet brought the ROUTE ERROR rather than discarding it. In order to salvage the data packet, the intermediate node sending a ROUTE ERROR looks up its own Route Cache for a route from itself to the target of the data packet causing the ROUTE ERROR. If such a route is available in the node's Route Cache, the node may salvage the data packet after informing the ROUTE ERROR by substituting the original source route on the data packet with the route from its own Route Cache. Then, the node forwards the data packet to the next node indicated along this source route. When

salvaging a packet in this manner, the data packet is also indicated as having been salvaged, in order to avoid salvaging a same packet multiple times. Otherwise, it is possible for the packet to go into a routing loop, as different nodes repeatedly salvage the same packet and replace the source route on the packet with routes from their own Route Caches [Johnson2001].

2) Automatic Route Shortening

If one or more intermediate hops of the route in use become no longer needed, the source route in use may be automatically shortened. This method of automatically shortening a route in use is quite similar to the use of passive acknowledgements. Specifically, if a node is able to overhear a packet containing a source route by setting its network interface in promiscuous receive mode, this node investigates the unused part of the source route. If the node is not the next hop recorded in the packet but is recorded in the later unused part of the source route, then it can assume that the intermediate nodes before itself in the source route contained in the packet are not needed any more for the packet to be delivered in the target node [Johnson2001].

3) Increased Spreading of ROUTE ERROR Messages

When a source node receives a ROUTE ERROR from an intermediate node having a route failure, this source node may flood this ROUTE ERROR to its neighbor nodes by piggybacking it on its next ROUTE REQUEST. This method prevents stale information in the caches of nodes around this source node from generating ROUTE REPLYs containing the same invalid link for which this source node received the ROUTE ERROR [Johnson2001].

4) Caching Negative Information

In some situations, DSR could probably have a benefit from nodes that cache *negative* information in their Route Caches [Johnson2001]. As an example in Figure 2.4, if node **A** caches the information that the link between **C** and **D** is currently broken, rather than simply removing the broken route from its Route Cache, node **A** is able to guarantee that no ROUTE REPLY containing the broken route in response to its new ROUTE REQUEST is accepted. This negative cached information must have a short expiration period, because node **A** will not accept to place any routes including the broken link in its Route Cache, while this negative information resides in the cache, even if this broken link starts working again [Johnson2001].

There is another case in which cached negative information might be beneficial. The case is where a link is supporting highly changeable service, sometimes working properly but sometimes not working. For example, this kind of case can occur in the situation where the link

is on the border of the limit of the sending node's wireless transmission range and there are considerable sources of interference near the receiving node on this link. In this situation, caching the negative information containing the broken route can prevent a source node from adding this troublesome link back to its Route Cache during the brief period in which it is working properly [Johnson2001].

2.3 IEEE 802.11 MAC Protocol

In our system, WLAN cards with the IEEE 802.11 P2P (Peer-To-Peer) capability are used for wireless communication between Pocket PCs. Thus, in order to help readers easily understand the analyses of experimental results reported in Chapter 4, this Section will give a brief overview of the IEEE 802.11 technology, particularly in terms of its architecture, physical layer and MAC sublayer, and basic access method to the medium, known as *Distributed Coordination Function (DCF)*, based on a *Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)* for wireless transmission.

2.3.1 Architecture Components

The *basic service set* (BSS) is the primary building block of the IEEE 802.11 architecture [Crow1997]. A BSS refers to a group of stations that are directly controlled by a single coordination function, such as Distributed Coordination Function (DCF) or Point Coordination Function (PCF), which will be explained below. The geographical area composed of the BSS is called as the *basic service area* (BSA), which is similar to a cell in a cellular communication network. Theoretically, all stations in a BSS can have communications directly with all other stations in a BSS. However, since transmission medium degradation may be caused by multi-path fading or interference from adjacent BSSs reutilizing the same physical-layer characteristics, such as frequency and spreading code or hoping pattern, it can produce some stations to appear "hidden" from other stations [Crow1997].

An ad hoc network is an intentional grouping of stations into a single BSS for the goals of internetworked communications with no help of any infrastructure network. Figure 2.5 illustrates an *independent* BSS (IBSS), which is the formal name of an ad hoc network in the IEEE 802.11 standard.

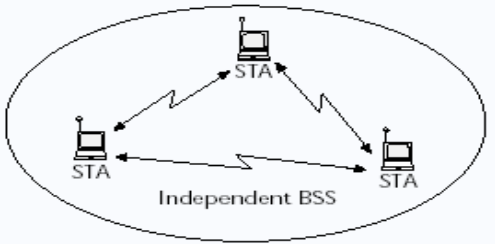

Figure 2.5: Sketch of an Ad Hoc Network.

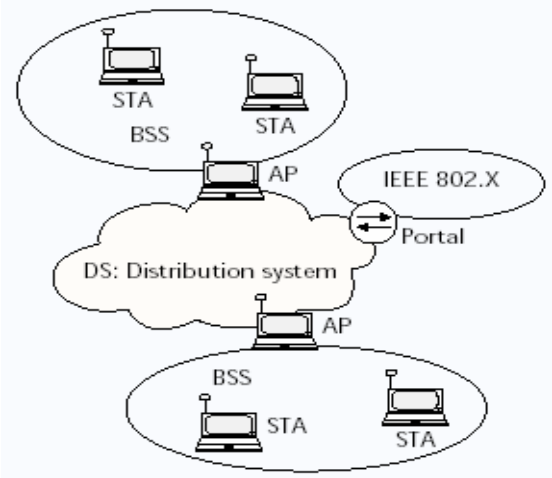

Figure 2.6: Sketch of an Infrastructure Network.

In the BSS, any station can communicate with any other station without requiring a centralized access point (AP) to channel all traffics through it [Crow1997].

Different from the ad hoc network, infrastructure networks are utilized to provide wireless users for specific services and range extension. In the context of the IEEE 802.11, infrastructure networks are constructed using APs. The AP is similar to the base station in a cellular communications network. The AP provides range extension by supporting the integration points needed for network connectivity between multiple BSSs, therefore forming an *extended service set (ESS)* [Crow1997].

The BSS appears one large BSS to the *logical link control (LLC)*. The ESS is made up of multiple BSSs that are integrated together by a common *distribution system (DS)*. The DS can be regarded as a backbone network that is in charge of MAC layer transport of *MAC service data units (MSDUs)*. Based on the specifications of the IEEE 802.11, the DS is implementation independent. Thus, the DS could be a wired IEEE 802.3 Ethernet LAN, IEEE 802.4 token bus LAN, IEEE 802.5 token ring LAN, fiber distributed data interface (FDDI) metropolitan area network (MAN), or another IEEE 802.11 wireless medium. Even though the DS could physically be the same transmission medium as the BSS, they are logically not same because the DS is exclusively used as a transport backbone to transfer packets between different BSSs in the ESS [Crow1997].

In addition, an ESS can provide a gateway access for wireless users into a wired network such as the Internet. This is achieved by way of a device known as a *portal*. The portal is a logical entity specifying the integration point on the DS where the IEEE 802.11 network

incorporates with a non-IEEE 802.11 network. If the network is an IEEE 802.x, the portal integrates functions that are similar to a bridge. That is, the portal provides range extension and the translation between different frame formats. Figure 2.6 gives an illustration of a simple ESS developed with two BSSs, a DS, and a portal access to a wired LAN [Crow1997].

2.3.2 IEEE 802.11 Layers Description

As any other 802.x protocol, the IEEE 802.11 protocol covers the physical layer and the Medium Access Control (MAC) layer [Brenner1997, Crow1997]. Figure 2.7 summarizes a brief description of the IEEE 802.11 layers [Crow1997].

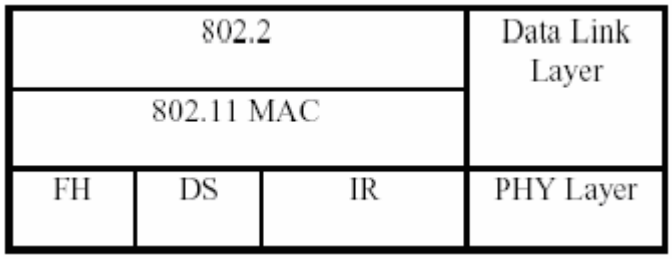

Figure 2.7: IEEE 802.11 Layers Description.

2.3.2.1 Physical Layer

The IEEE 802.11 draft specification requires three different physical-layer implementations, which are *Frequency Hopping Spread Spectrum (FHSS), Direct Sequence Spread Spectrum (DSSS),* and Infrared (*IR)* light [Brenner1997].

The FHSS uses the 2.4 GHz Industrial, Scientific, and Medical (ISM) band, for instance between 2.4000 and 2.4835 GHz. In the U.S., a maximum of 79 channels are stated in the hopping set. The first channel has a center frequency of 2.402 GHz, and all succeeding channels are spaced 1 MHz apart. The 1 MHz separation is required by the Federal Communications Commission (FCC) for the 2.4 GHz ISM band. The channel separation matches to 1 Mb/s of instantaneous bandwidth. Three different hoping sequence sets are formed with 26 hopping sequences per set. Different hopping sequences make multiple BSSs possible to coexist in the same geographical area, which may become crucial to reduce congestion and maximize the total throughput in a single BSS. Having three different sets is in order to avoid extended collision

periods between different hopping sequences in a set. The minimum hop rate allowed is 2.5 hops/s [Crow1997].

The DSSS also uses the 2.4 GHz ISM frequency band, where the 1 Mb/s basic rate is encoded by differential binary phase shift keying (DBPSK), and a 2 Mb/s enhanced rate uses differential quadrature phase shift keying (DQPSK). The spreading is achieved by dividing the available bandwidth into 11 sub-channels, each 11 MHz wide, and using an 11-chip Barker sequence to spread each data symbol [Crow1997].

The IR (Infrared) specification recognizes a wavelength range from 850 nm to 950 nm. The IR band is devised for indoor use only and works with non-directed transmissions. The IR specification is created to enable stations to receive line-of-site and reflected transmissions [Crow1997].

2.3.2.2 MAC Sublayer

The *Medium Access Control (MAC)* sublayer has responsibilities for the channel allocation procedures, protocol data unit (PDU) addressing, frame formatting, error checking, and fragmentation and reassembly [Crow1997]. The transmission medium can solely operate in the contention mode. The operation of the transmission medium in the contention mode requires all stations to contend for access to the channel when each packet is transmitted. The medium can also alternate between the contention mode and non-contention mode, known as the *contention period (CP)* and a *contention-free period (CFP)* respectively [Crow1997]. While the medium is in the CFP, the medium usage is mediated by the AP, thus removing the need for stations to content for channel access. The IEEE 802.11 provides three different types of frames, which are management, control, and data. The management frames are utilized for station association and disassociation with the AP, timing and synchronization, and authentication and de-authentication [Crow1997]. Control frames are used for handshaking during the CP, for positive acknowledgements during the CP, and to end the CFP. Data fames are used for the transmission of data during the CP and CFP, and can be combined with polling and acknowledgements during the CFP [Crow1997].

Figure 2.8 illustrates the standard IEEE 802.11 frame format [Crow1997]. Note that if the optional Wired Equivalent Privacy (WEP) protocol is used, the frame body (MSDU) is variable-length field consisting of the data payload and 7 octets for encryption and decryption. The IEEE Standard 48-bit MAC addressing is for identifying a station [Crow1997].

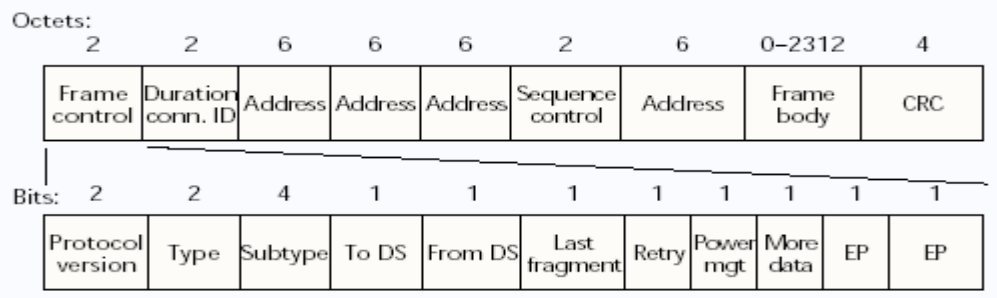

Figure 2.8: Standard IEEE 802.11 Frame Format.

The 2 duration octets present the time (in microseconds) the channel will be allotted for successful transmission of a MAC protocol data unit (MPDU). The type bits indicate the frame as control, management, or data. The subtype bits also show the type of frame such as Clear to Send control frame, etc. Error detection is checked by a 32-bit cyclic redundancy check (CRC) [Crow1997].

2.3.3 Basic Access Method: CSMA/CA

The *Distributed Coordination Function (DCF)* is a basic channel access protocol for asynchronous data transmission in the contention period based on a *Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)* mechanism [Brenner1997, Sheu2002, Issariyakul2003]. As defined in the standard, all stations must support the DCF [Crow1997]. The DCF operates exclusively in the ad hoc network, and either works solely or works with the PCF in an infrastructure network. The MAC architecture is described in Figure 2.9 that shows the DCF sitting directly on top of the physical layer and supporting contention services [Crow1997].

Contention services imply that each station with an MSDU queued for transmission must contend for access to the channel. If the MSDU is transmitted, the station must contend again for access to the channel for all following frames. Contention services support fair access to the channel for all stations [Crow1997].

A CSMA protocol works in the following way. A station wishing to transmit senses a medium. If the medium is busy (e.g. some other station is transmitting), then the station will defer its transmission to a later time. If the medium is sensed free, then the station is permitted to transmit [Brenner1997].

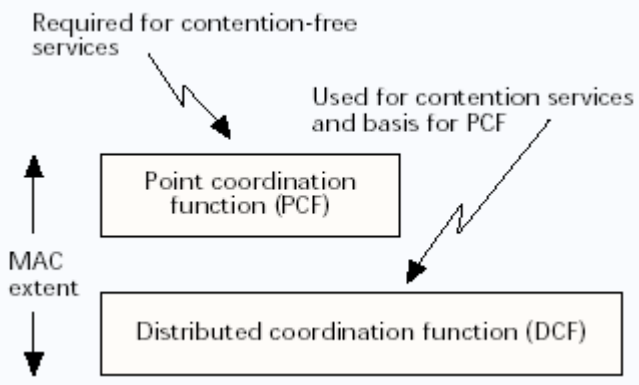

Figure 2.9: MAC Architecture.

These kinds of protocols are very effective when network loads are not heavy because it allows stations to transmit with minimum latency. However, there is always a chance of occurrence of collision when more than one station transmits simultaneously, because stations sense the medium free and decide to transmit at one. These collision situations must be identified so that the MAC layer can retransmit the packet by itself and not by upper layers, which may cause significant delays [Brenner1997].

In the Ethernet case, this collision is identified by the transmitting stations, which enter a retransmission stage based on an exponential random backoff algorithm [Brenner1997]. Even though the Collision Detection is a good mechanism on a wired LAN, they cannot be applied to a Wireless LAN environment, due to the following two reasons [Brenner1997]:

- Implementation of a Collision Detection Mechanism would require the implementation of a Full Duplex radio, enabling transmitting and receiving at once, which is an approach increasing the price significantly.
- Different from the wired LAN environment using the Collision Detection mechanism, the assumption that all stations hear each other is not true on a wireless environment. That is, the fact that a station wishes to transmit and senses the medium free does not necessarily mean that the medium is free around the receiver area.

In order to solve these problems, the IEEE 802.11 provides a Collision Avoidance mechanism combined with a positive acknowledgement scheme, as follows [Brenner1997]:

A station is willing to transmit and senses the medium. If the medium is busy, then the station defers the transmission. If the medium is free from the specified time (called DIFS,

Distributed Inter Frame Space, in the standard), then the station is allowed to transmit. The receiver will check the CRC of the received packet and transmit an acknowledgement packet (ACK). The receipt of the acknowledgement means to the sender no collision occurred. If the sender does not receive ACK, it will resend the same fragment until it receives the ACK or throw away after a certain number of retransmission [Brenner1997].

In IEEE 802.11 [Crow1997], carrier sensing is done in terms of two layers, which are *physical carrier sensing* at the air interface and *virtual carrier sensing* at the MAC sublayer. *Physical carrier sensing* recognizes the existence of other IEEE 802.11 WLAN users by analyzing all detected packets, and also identifies activity in the channel through relative signal strength from other source [Crow1997].

A source station performs *virtual carrier sensing* by transmitting Request To Send (RTS), Clear To Send (CTS), and data frames containing MPDU duration information in their headers. An MPDU, a complete data unit being passed from the MAC sublayer to the physical layer, contains header information, payload, and a 32-bit CRC. The duration information indicates the amount of time (microseconds) after the end of the current frame that the channel will be used to complete the successful transmission of the data or management frame [Crow1997]. Stations in the BSS utilize the duration information to modify their network allocation vectors (NAVs), indicating the amount of time that has to elapse until the present transmission session is done and the channel can be checked out again if it is idle or not. The channel is indicated as busy if either the physical or virtual carrier sensing mechanisms say the channel is busy. The wireless medium is accessed based on priority controlled through the utilization of Inter Frame Space (IFS) time intervals between the transmissions of frames. The IFS intervals are required periods of idle time on the transmission medium. There are three IFS intervals specified in the standard as follows [Brenner1997, Crow1997]:

- Short Inter Frame Space (SIFS): This interval is the smallest IFS, which is followed by PIFS and DIFS respectively. When a station is only required to wait a SIFS, it has priority access over those stations that are required to wait a PIFS or DIFS before transmitting. Therefore, SIFS has the highest-priority for accessing to the communications medium.
- Point Coordination Function IFS (PIFS): This interval is used by the Access Point (AP) to obtain access to the medium before any other station. This interval value is a SIFS plus slot time, for example 78 microseconds.
- DCF-IFS (DIFS): This interval is IFS used for a station wishing to start a new transmission, which is calculated as PIFS plus one slot time, for instance 128 microseconds.

For the basic access mechanism, when a station senses the idleness of the channel, the station waits for a DIFS period and samples the channel again. If the channel is still idle, the station sends an MPDU. The receiver calculates the checksum and decides the correctness of the received packet. If the received packet is correct, the receiver waits a SIFS interval and transmits a positive acknowledgement frame (ACK) back to the sender station for the successful transmission [Crow1997]. Figure 2.10 illustrates a timing diagram explaining the successful transmission of a data frame [Crow1997].

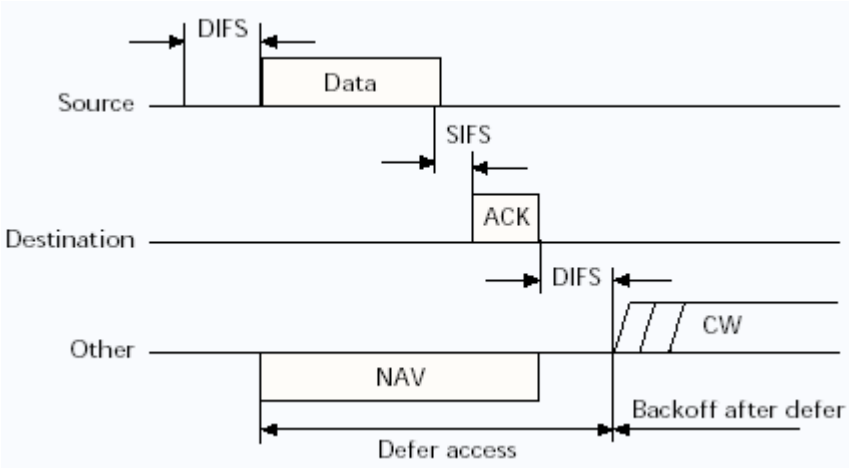

Figure 2.10: Transmission of an MPDU without RTS/CTS.

When the data frame is transmitted, the duration information of the frame is utilized to inform all stations in the BSS how long the medium will be busy. All stations hearing the transmission of the data frame adjust their NAVs according to the duration field value containing the SIFS interval and the ACK of the data frame. When a collision occurs, a source station continues transmitting the complete MPDU because the source station in a BSS cannot hear its own transmissions. If the MPDU is larger, for example 2300 octets, a lot of channel bandwidth is wasted because of a corrupt MPDU. The station can use RTS and CTS control frames in order to reserve channel bandwidth before it transmits an MPDU so that the amount of bandwidth wasted is minimized when a collision occurs [Crow1997]. RTS and CTS control frames are relatively small like 20 octets for RTS and 14 octets for CTS compared to the maximum size of MPDU such as 2346 octets. After successfully contending for the channel, the source station transmits the RTS control frame first with a data or management frame queued for transmission to an indicated destination station. All stations in the BSS hearing the RTS packet read the

duration field and adjust their NAVs based on the duration information contained in the RTS, as previously shown in Figure 2.8 [Crow1997]. The destination station transmits a CTS packet in response to the RTS packet after spending an SIFS idle period. Stations hearing the CTS packet update their NAVs based on the duration field value of the received CTS packet. After the source station receives the CTS successfully, the source station virtually confirms that the medium is stable and reserved for the successful transmission of its MPDU. Note that stations can update their NAVs based on the RTS from the source station and CTS from the destination station, which is helpful to combat the "hidden terminal" problem. Figure 2.11 illustrates the transmission of an MPDU using the RTS/CTS mechanism [Crow1997].

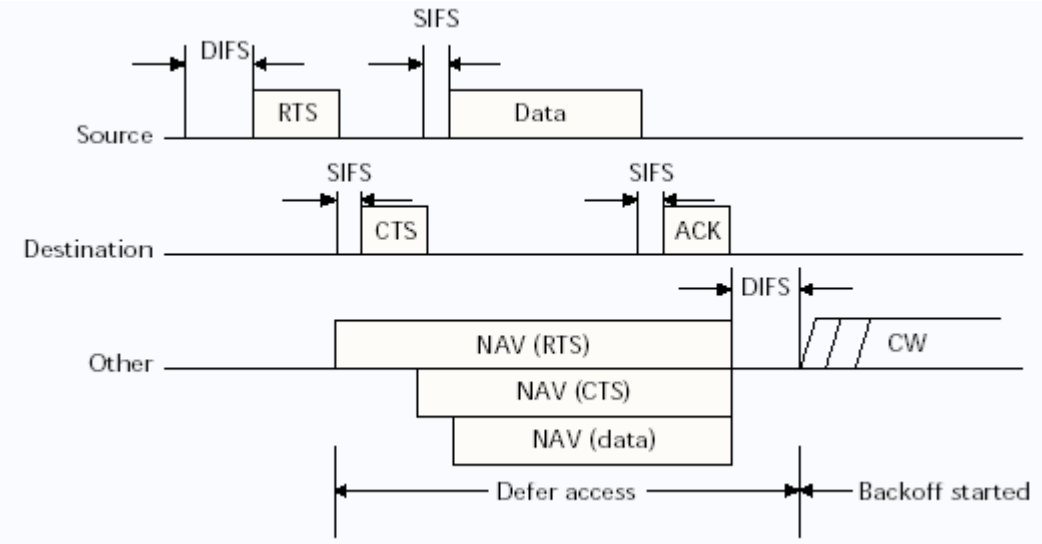

Figure 2.11: Transmission of an MPDU using RTS/CTS.

The RTS/CTS mechanism is optional that stations can use RTS/CTS always or use RTS/CTS or not based on the RTS_Threshold (RT: manageable parameter). So, stations use RTS/CTS when the MSDU exceeds the value of RT. Otherwise, they do not use the RTS/CTS. If a collision occurs with RTS or CTS MPDU, bandwidth wasted will be far less when compared to a large data MPDU [Crow1997]. However, if there is a slightly loaded medium, additional latency is imposed by using the RTS/CTS frames causing the extra overhead. Large MSDU passed from the LLC to the MAC may require fragmentation to support reliability. The fragmentation is performed based on the manageable parameter Fragmentation_Threshold [Crow1997].

A random backoff procedure is used to perform the collision avoidance portion of CSMA/CA. If a station wants to transmit a frame and senses the channel is busy, the station waits for a DIFS until the channel becomes idle, and then calculates a random backoff time [Crow1997]. In the IEEE 802.11, time is slotted in time period that is correspondent to a Slot_Time. The Slot_Time in the IEEE 802.11 is much smaller than an MPDU and is utilized to set the IFS interval and decide the backoff time for stations in the CP [Crow1997]. The random backoff time is an integer value corresponding to a number of time slots. At first, the station calculates a backoff time between 0 and 7. After the medium becomes idle after a DIFS period elapsed, stations decrease their backoff timers until the medium becomes busy again or the time reaches 0. If the time has not decremented to 0 and the status of medium becomes busy, the station freezes its timer. When the timer is reached zero finally, the station transmits its frame. If more than one station reach zero simultaneously, a collision will occur, and each station should generate a new backoff time between 1 and 15 [Crow1997]. The idle time after a DIFS period is called the *contention window* (CW). The benefit of this channel access method is that it encourages fairness among stations. However, it has weakness in that it probably could not support a distributed time-bounded service (DTBS), which is for traffic bounded by specified time delays to achieve an acceptable quality of service such as packet-sized voice and video [Crow1997]. Fairness is maintained because each station has to contend again for the channel after every transmission of an MSDU. All stations have equal chances of obtaining access to the channel after each DIFS interval. With DCF, there is no mechanism to guarantee minimum delay to stations supporting time-bounded services [Crow1997].

In order to support time-bounded services, the IEEE 802.11 standard provides the optional use of the point coordination function (PCF) [LaMaire1996, Crow1997]. In the PCF, a point coordinator or PCF station always has a priority control of the medium. PCF station in the PCF means an access point. Hence, the use of PCF and support of time-bounded services are limited to networks with infrastructure mode [Crow1997]. Thus, this PCF will not be covered in details because this document focuses on the ad hoc mode operation of the IEEE 802.11 WLAN.

2.4 Related Work

This Section reviews previous studies on the performance evaluation of the DSR protocol when compared with other routing protocols, and the testing of optimization features of DSR, particularly in terms of modifications of conventional cache structure in DSR.

2.4.1 Comparative Studies

There are various kinds of comparative studies regarding the performance of the DSR protocol and other routing protocols in ad hoc networks [Camp2002, Raju2002, Jiang2001, Broch1998]. In addition, some researchers compare the traditional DSR protocol with the modified DSR protocol [Zhong2003]. This Section will introduce these comparative works done by other researchers in the past. In particular, when comparative studies are described, the performance of DSR is mainly addressed, rather than other details about the performance of other routing protocols in ad hoc networks.

Camp and her colleagues [Camp2002] studied performance characteristics of two location-based routing protocols for ad hoc networks, namely, *Location-Aided Routing (LAR)* and *Distance Routing Effect Algorithm for Mobility (DREAM)*. They compared the performance of these two protocols with the DSR protocol and a protocol flooding all data packets. They simulated 50 moving nodes based on the random waypoint model by using the *ns*-2 simulator. In the random waypoint model, each node moves from one destination to a next destination with randomly selected locations, speed (speed of mobility of each node), and pause time (time for each node to stay at each random destination). They used four performance metrics, namely, protocol overhead, network-wide data load, end-to-end delay, and data packet delivery ratio. In particular, for the performance analysis of the DSR protocol, the result shows that data packet delivery ratio decreases as the speed increases at a short pause time because the random waypoint model produces a dynamic network topology at a high speed and short pause time. Further, the end-to-end delay increases, as speed increases at a short pause time. More control packets are used when the speed increases at a short pause time. Also, the data packet delivery ratio (the number of data packets delivered over the number of data packet transmitted) decreases at relatively high speeds and short pause times. Finally, data packet load (the number of data byte transmissions per data packet delivered) remains constant as the speed increases at a short pause time because of the behaviors of flooding of DSR in order to find a new route. Additionally, the paper reports that location information and promiscuous mode operation improve the performance of the DSR protocol.

Zhong and Yuan [Zhong2003] also studied the performance of the DSR protocol in special scenario using location information. They compared the efficiency of the modified DSR protocol with that of the traditional DSR protocol. The modified DSR is the combined protocol

of the traditional DSR features in special scenarios with location information. Each node contains local information rather than global information in the routing of wireless ad hoc networks. They integrated the DSR protocol with location information given by the two special scenarios, which are cross and cycle circumstances. Thus, when a node sends a packet, it checks its location table, and then chooses an optimal route to reach its destination. Once the path is defined, the packet is immediately transmitted with a way of multi-hops. The result of these experiments with 8 participating nodes shows that the modified DSR protocol performs more efficiently than the traditional DSR protocol in the two specified scenarios.

Raju and Garcia-Luna-Aceves [Raju2001] wrote a paper titled *Scenario-based Comparison of Source Tracing and Dynamic Source Routing Protocols for Ad Hoc Networks*. Their paper proposes source tracing as a new workable method to routing in ad hoc networks where routers communicate the second-to-last hop and distance in preferred routes to destination nodes. They used two source-tracing algorithms. One is *Bandwidth Efficient Source Tracing (BEST)*, which is a table-driven protocol by which routers maintain routing information for all destinations. The other is the *Dynamic Source Tree (DST)* protocol, which is an on-demand routing protocol by which routers maintain routing information for only those destinations to whom they need to forward data. The DSR protocol is used for simulation experiments to compare those two protocols because DSR has been shown to use less control overhead packets than other on-demand routing protocols. The results of the simulation experiments are that DST demands far less control packets to obtain comparable or better average delays and percentage of packet delivered than DSR, and that BEST accomplishes comparable results to DSR while maintaining routing information for all destinations.

Jiang and Garcia-Luna-Aceves [Jiang2001] conducted a study on performance comparison of three protocols for ad hoc networks. They compared the performance of three protocols, which are *Source Tree Adaptive Routing (STAR)*, *Ad Hoc On-Demand Distance Vector (AODV)*, and *Dynamic Source Routing (DSR)*. They tested the performance of three protocols using the *GloMoSim* simulation environment. The metrics used to measure the performance of three routing protocols for ad hoc networks are control overhead, amount of data delivered, and average latency in packet delivery. In particular, the amount of data delivered means the number of data packets delivered with same amount of data flow. For example, when data flow is 20, if the amount of data delivered is 60, 40 data packets are delivered redundantly. They indicate that DSR has the lowest data delivery rate compared to the other two protocols. In

addition, DSR has the lowest control overhead in comparison with the other two protocols. However, for the latency in packet delivery, DSR does not perform better. Sometimes, DSR takes more latency to deliver a data packet compared to the other two protocols.

Finally, we want to introduce the study of Broch and his colleagues [Broch1998]. Their paper shows the results of a detailed packet-level simulation with the comparison of four multi-hop wireless ad hoc network routing protocols covering a range of design choices, such as *Destination-Sequenced Distance Vector (DSDV), Temporarily-Ordered Routing Algorithm (TORA), Dynamic Source Routing (DSR),* and *Ad Hoc On-Demand Distance Vector (AODV)*. They extended the *ns*-2 network simulator to correctly model the MAC and physical-layer behavior of the IEEE 802.11 wireless LAN standard, containing a realistic wireless transmission channel model, and demonstrated the results of simulations of networks consisting of 50 mobile nodes. As a movement model, they used the random waypoint model. They used three performance metrics, which are packet delivery ratio, routing overhead (regarding every transmission of the routing packet as one transmission), and path optimality (the difference of used paths and length of the shortest path physically existed). They varied pause time between 0 second and 900 seconds in two different speeds, 1 m/s and 20 m/s. Also, they tried the same experiments with different number of participating nodes, such as 10, 20, 30 nodes during the fixed simulation time, 900 seconds. The result shows, focusing on the performance of the DSR protocol, that DSR performs better than the rest of three protocols. For the data delivery ration, DSR provides above 95 percent delivery ratio of the data packets sent under different scenarios consisting of different number of nodes, various pause times, and two different speeds. In addition, DSR uses much less number of routing packets compared to the other three protocols. Furthermore, DSR optimally utilizes paths it takes. The result regarding the path optimality represents that DSR uses the shortest path in most of times.

2.4.2 Testing Optimization Features of the DSR Protocol

Some researchers tried to improve the performance of the DSR protocol in ad hoc networks by testing its optimization features. In particular, this Section will introduce some studies on the effects of improved cache structures on the performance of the DSR protocol in ad hoc networks [Lou2002, Wu2002, Hu2000, Hu2002]. In addition, the study on the optimization of the DSR protocol by reducing Route Discovery Reply packets will be described [Seet2003].

Lou and Yuguang [Lou2002] studied the effects of different cache organizations on the performance of the DSR protocol in ad hoc networks. They tested two types of cache organizations, namely, a *path cache* and a *link cache*. A path cache is structured that each cache entry represents an entire path from a source to all destinations, while a link cache has each individual link to one destination. Thus, a link cache has the potential to use the acquired route information more efficiently. However, if the cache is not updated properly, the out-of-date cache information may cause more route errors. Then, it leads more expensive operation, such as Route Discovery. In their paper, they studied two types of link update mechanisms, which are a *static timeout mechanism* and an *adaptive timeout mechanism*. The static timeout mechanism uses the expiration time of a link using a statistically assigned lifetime, while the adaptive timeout is proposed as the link timeout interval to diverse node mobility levels on the basis of the estimation from a moving average of real link lifetime statistics. They uses three metrics to measure the performance of cache organizations in the DSR protocol for ad hoc networks, such as routing overhead, packet delivery ratio, and end-to-end delay. The results shows that a link cache organization without a proper stale link removal method could have severe performance degradation due to the large number of route error messages occurred. However, using a proper timeout method, the link cache organization reduces the routing overhead significantly and outperforms the path cache when the network load is heavy. Also, they recommended that switching between two types of cache organizations dynamically in response to the network load condition could improve the overall performance of the DSR protocol.

Another study is done to improve the performance of the DSR protocol by optimizing cache structures [Wu2002]. The paper shows the implementation of the extended DSR protocol by maintaining two nodes disjoint path as a consequence of respective Route Discovery process. The paper explains that the efficiency of the DSR protocol largely depends on the “hit ratio” of route cache. That is, the high efficiency occurs when the route from a source to a destination exists in source’s cache. Thus, when the “miss” of route cache occurs, the source cannot avoid invoking the relatively expensive Route Discovery process. Therefore, the paper proposes an approach to reduce the occurrence of Route Discovery process. As a result, the paper explains the effectiveness of multi-path construction process as a way for the reduction of additional overhead caused by the expensive Route Discovery process due to the lack of paths in cache.

Hu and Johnson [Hu2000] presented the effects of different design choices for the cache structure in the DSR protocol for wireless ad hoc networks in their paper. They insist that caching is a significant component in order to reduce the overhead of performing a new Route Discovery before sending each data packet. However, caching can also bring risks in maintaining routing information in a cache after the information is not valid any more because of a changed network topology. They divided the problem into choices of cache structure, cache capacity, and cache timeout. Using detailed simulations of wireless ad hoc networks of 50 mobile nodes, they studied a larger number of different caching algorithms using a range of design choices, and simulated each cache mainly over a set of 50 different movement scenarios derived from 5 different types of mobility model. Their evaluations includes packet delivery ratio, routing packet overhead, packet delivery latency, and path optimality relative to the shortest path, obtained by each caching algorithm. The result shows that a cache structure based on a graph representation of individual link performs better than that based on complete paths through the network. Moreover, they found that there are some subtle relationships between cache timeout policies and cache capacity limits. Further, caches of unlimited capacity or with no cache timeout limits perform significantly worse.

Hu and Johnson [Hu2002] conducted another study to improve the performance of the DSR protocol by reducing overhead caused by stale information in the route cache of each node. They explains that stale information in the cache of each node substantially increase the risk of cache cross-pollution because stale routing information in one node's cache does not exist any longer. Even when a node has actually learned the invalid route, it is still possible for the node to hear the stale information again. In their paper, they present a new mechanism called *epoch numbers* to reduce this problem of cache staleness, by stopping the re-learning of stale information of a link after knowing that the link is not valid. Their scheme allows a node having heard both of a broken link and a discovery of the same link to sequence the two events in order to decide whether the link break or the link discovery occurred before the other. Their paper reported that their solution performs well in that stale information generally cannot override fresh information and their solution does not increase packet overhead. Hu and Johnson [Hu2002] present that their solution works by providing exactly the information needed to sequence discovery of new network links and notification of broken links, thus preventing nodes from re-learning stale cache information.

As another way of optimizing the DSR protocol in ad hoc networks, Seet et al. [Seet2003] propose a new way to reduce network traffics. Many works have been done for the optimization of the DSR protocol in reducing the routing packet overhead, especially from a large saving on route requests, which is usually called Route Discovery process. This paper suggests a new way of optimization of the DSR protocol in terms of decreasing route discovery reply packets, which are known as the reply packets in response to the Route Discovery packets. The authors explain why the route reply process is more expensive to transmit than route requests. Since the route replies are unicast packets (while the route requests are broadcast packets), they use additional control packets for RTS/CTS exchanges in the 802.11 MAC. Moreover, the route discovery reply packets will be larger in size because they have to carry the whole source route obtained by the route request. Therefore, a large number of route discovery reply packets can still consist of a significant portion of the wireless bandwidth. In addition, the device power is needed to process them. Both bandwidth and power are the most precious resources in mobile ad hoc networks. Thus, in terms of the efficiency of resources in mobile ad hoc networks, a large portion of cached routes from route discovery reply packets can also be more harmful than beneficial to the routing performance. In particular, the large amount of route discovery reply packets will be detrimental when nodes are highly mobile, because routes acquired become out-of-date more frequently, making their utilization less effective. Therefore, this paper proposes an optimization version of the DSR protocol by minimizing the number of cached route discovery reply packets. Since a source only uses the shortest path in its cache, the longer route than the current route will not be useful when the route discovery reply packet containing the longer route is received in the source. Hence, a destination node only sends the route discovery reply packets when the route request packet contains a shorter route. Otherwise, the destination node discards the route request packet. The authors conclude that the proposed optimization of the DSR protocol reduces the number of Route Discovery overhead at higher node mobility (shorter pause time), while it does not have much benefit at lower node mobility (longer pause time).

2.4.3 Other Studies

There are also other types of papers discussing about different aspects of the performance of the DSR protocol for wireless ad hoc networks. This Section will introduce some studies. One is a study focused on improving the Quality of Service (QoS) in respect to data transmission of DSR [Leung2001]. Another study introduces a way to reduce energy consumption in the

DSR protocol for ad hoc networks [Xu2001]. In addition, one proposes a new way to improve security of the DSR protocol [Ghazizadeh2002].

According to the study of Leung et al. [Leung2001], the lack of Quality-of-Service (QoS) is indicated as one of weaknesses of the existing best-effort routing protocols, such as DSR not supporting QoS in regard to the data transmission. They suggest a new QoS metric, which supports end-to-end reliability, in order to select end-to-end routes to give more stability and reliability to routes. Thus, they present a distributed *multi-path dynamic source routing (MP-DSR)* protocol for wireless ad-hoc networks to increase QoS feature in regard to end-to-end reliability. Their simulation study demonstrates the high effectiveness of the presented protocol particularly, in terms of the higher achievement of successful packet delivery ratio as compared to the current DSR protocol.

One study introduces a *geographical adaptive fidelity (GAF)* algorithm to reduce energy consumption in wireless ad hoc networks [Xu2001]. According to the paper of Xu et al. [Xu2001], GAF saves energy by identifying nodes, which are equal from a routing perspective and by turning off unnecessary nodes while maintaining a constant level of routing fidelity. GAF controls this policy using application and system-level information. Nodes that source or sink data remain turned-on and intermediate nodes watch and balance the use of energy. GAF does not depend on the underlying ad hoc routing protocol. Xu and his colleagues simulated GAF with unmodified *Ad-hoc On-Demand Distance Vector (AODV)* and *Dynamic Source Routing (DSR)*. The result shows that the simulated GAF can consume 40% to 60% less energy than an unmodified ad hoc routing protocol. Furthermore, the simulations of GAF demonstrate that network lifetime increases proportional to node density. For instance, a four-fold increase in node density brings network lifetime increase from 3 to 6 times, depending on the mobility pattern. More generally, GAF is a good example of adaptive fidelity, a method proposed for enlarging the lifetime of self-configuring systems by employing redundancy to conserve energy while keeping application fidelity.

Ghazizadeh and his colleagues [Ghazizadeh2002] presented *Security Aware Adaptive Dynamic Source Routing Protocol (SADSR)*, which is a secure routing protocol for mobile ad hoc networks. They explain that SADSR verifies the routing protocol messages using digital signatures on the basis of asymmetric cryptography. The basic ground on SADSR is to have multiple paths to each destination and store a local trust value for each node in the network.

Each path has an assigned trust value based on the trust values of the nodes, which happen on that path. The preferred paths for routing are selected based on higher trust values. They implemented their approach in the *ns*-2 simulator and measured the performance of SADSR and DSR. Their results prove that in the existence of malicious nodes, SADSR performs better than DSR in the packet delivery ratio with an acceptable network load.

2.5 Summary

This Chapter has given an overview and knowledge backgrounds of this thesis work in the design, implementation and analysis of an ad hoc messenger application in mobile ad hoc networks.

Section 2.1 discussed wireless ad hoc networks. A wireless ad hoc network consists of a collection of mobile hosts without the aid of any pre-existed infrastructure or centralized administration. Main characteristics of wireless ad hoc networks are described, such as a dynamic network topology, bandwidth-limited and fluctuating capacity links, low-power and resource-limited operations, constrained physical security, and decentralized network control.

Section 2.2 described main features of the DSR protocol. The DSR protocol is a simple and efficient routing protocol for multi-hop wireless ad hoc networks using mobile devices. The DSR protocol uses source routing for a packet from a source to be delivered in the target node. Instead of flooding a periodic broadcast packet to obtain routing information, this DSR protocol transmits ROUTE REQUEST message to a source's neighboring nodes located within the wireless transmission range of the source node totally based on a node's need. Thus, each node does not have to transmit any type of periodic route advertisement or neighbor detection packets. The DSR protocol consists of two main mechanisms, *Route Discovery* and *Route Maintenance*. Route Discovery is initiated when there is no available route in the Route Cache of the source node. When a target node received the Route Discovery request, the target node returns route discovery reply packet containing the route found to the initiator of the Route Discovery. Then, the obtained route from the Route Reply sent by the target node is used to transmit subsequent data packets. Route Maintenance is the mechanism where a source node detects link failure, while it sends a packet to the target node using the source route found by Route Discovery. When the route in use is broken, the source node tries to use alternative routes in its Route Cache. If any alternative route is not available in its Route Cache, a Route Discovery is initiated to find a new route again.

Section 2.3 gave a brief overview of the IEEE 802.11 MAC protocol for readers to understand the analyses of results more easily because our experiments are based on wireless transmission between each node using the IEEE 802.11b technology. The IEEE 802.11 defines three physical layer implementations, which are Frequency Hopping Spread Spectrum (FHSS), Direct Sequence Spread Spectrum (DSSS), and IR. Both FHSS and DSSS operate on the 2.4 GHz ISM band. The IR band is devised for indoor use only and works with non-directed transmissions. The Distributed Coordination Function (DCF) is a basic channel access protocol for asynchronous data transmission in the contention period based on a Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism. All stations must support DCF, which mainly operates in an ad hoc network. The CSMA protocol is used for each node to avoid collision by deferring its transmission to a later time when the channel is busy. After each node waits for a certain period of time until the medium is free, a node in DCF mode wishing to transmit a packet can gain access to the medium by using four-way handshake consisting of RTS/CTS (Request-To-Send/Clear-To-Send) packet. The sending node sends RTS packet to the receiving node. If the receiver accepts the request, it sends CTS to the sender of RTS. If no collisions occur, the sender can begin transmitting its data packet.

Finally, Section 2.4 reviewed some related works that have done previously. Some papers discussing the performance of the DSR protocol by comparing it with other routing protocols in ad hoc networks were introduced. In addition, some studies testing some optimization features of the DSR protocol were discussed. Finally, some studies to improve DSR with different methods were also surveyed.

In the next Chapter, we will describe design details to realize our system, such as hardware and software components used, network environments, specific design features, metrics to evaluate the performance of the DSR protocol, etc.

Chapter 3

Methodology

This Chapter describes the methodology for the implementation our multi-hop ad hoc instant messenger application using Pocket PCs and IEEE 802.11b.

3.1 Environment of the Application Development

3.1.1 Network Testing Environment

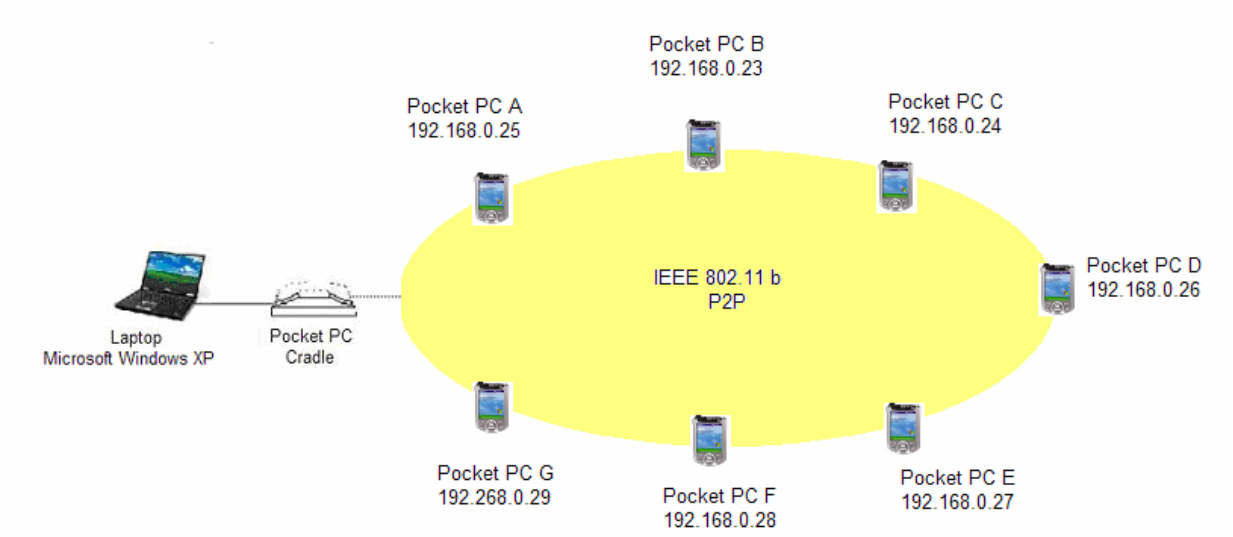

Figure 3.1: Network Testing Environment for the Application Development.

Figure 3.1 shows the network environment for our application development, modified based on Ye's independent study project done at Virginia Polytechnic Institute and State University in Fall 2002 [Ye2002]. Ye previously worked on this ad hoc messenger project. As Figure 3.1 shows, in our network environment for the application development, no access point or infrastructureis not required to set up the wireless ad hoc network. The compiled final executable files can be installed on the Pocket PCs through Microsoft ActiveSync and a cradle,

either using a serial port or USB port. Multiple Pocket PCs can be synchronized with the laptop through setting up a partnership. During the experiments, Pocket PCs communicate with each other through the Compaq wireless WLAN card's 802.11b P2P capability. The total number of participating Pocket PCs in this wireless ad hoc network is 7. For the convenient use of Pocket PCs, each Pocket PC is named and its IP address is assigned manually as follows:

Pocket PC ID	Pocket PC IP Address
A	192.168.0.25
B	192.168.0.23
C	192.168.0.24
D	192.168.0.26
E	192.168.0.27
F	192.168.0.28
G	192.168.0.29

Table 3.1: ID and IP Address of Each Pocket PC.

3.1.2 Software Components

The development software components are as follows:

For the Application Development:

- Microsoft Visual Studio .NET (C#) 2003: Main programming tool
- Microsoft Pocket PC 2002 SDK: Pocket PC platform development API SDK
- Microsoft ActiveSync Version 3.5: Synchronization software for Pocket PC
- Microsoft .Net Compact Framework 2003: Pocket PC .NET platform runtime

For the Network Driver:

- Compaq WLAN Driver: Pocket PC 802.11 network driver

For the Document Development:

- Remote Control Display: Tool to obtain screen shots for the user interfaces of developed applications. Downloaded from

http://www.microsoft.com/windowsmobile/resources/downloads/pocketpc/remotedsp_eula.mspx.

- ZedGraph: Graph Class Library in C#. Tool to draw graphs in the Statistical Analyzer that is an application developed for generating result graphs according to a set of selected x and y parameter. Downloaded from http://www.codeproject.com/csharp/ZedGraph.asp.

3.1.3 Hardware Components

The development hardware components are as follows:

- 1 Toshiba Satellite Mobile Intel® Pentium® 4 – M CPU 2.00GHz, 496 MB of RAM: Main development environment.
- 7 Compaq iPAQ 3870: Targeted platform
- 7 WL 110 11 Mbps Wireless LAN PC Card: Wireless network for P2P communication
- 7 Compaq Dual Slot PC Card Expansion Pack: Connect a Pocket PC with a WLAN card

The following is the main technical specification of Compaq iPAQ 3870 Pocket PC:

CPU	206 MHz Intel Strong ARM
Memory	64 MB SDRAM 32 MB Flash ROM
Resolution	320 x 240
Screen Type	Color
Operating System	Microsoft Pocket PC 2002

Table 3.2: Compaq iPAQ 3870 Technical Specification.

Figure 3.2: Pocket PC Hardware (Source: www.compaq.com).

The Compaq WL 110 Wireless PC card [Compaq2001] has the following specifications:

Form Factor	WL110 Wireless PC Card Type-II Extended	
Dimensions	(L x W x H)	117.8 x 53.95 x 8.7 mm
Weight	45 gram (PC Card)	
Temperature and Humidity (non condensing)		
Operation	0 ° to 55 ° C	Maximum humidity 95%
Transit	-20 ° to 70 ° C	Humidity 15 to 95%
Storage	-10 ° to 60 ° C	Humidity 10 to 90%

Table 3.3: Physical Specification of WL110 Wireless PC Card.

Compatibility	• IEEE 802.11 Standard for Wireless LANS (DSSS) • Wi-Fi (Wireless Fidelity) Certified by the Wireless Ethernet Compatibility Alliance (WECA)	
Host Operating System	Microsoft Window® 95: • NDIS3 Miniport Driver	
	Microsoft Window® NT v4.0: • NDIS4 Miniport Driver	
	Microsoft Window® 98/ME and 2000: • NDIS5 Miniport Driver	
Media Access Protocol	CSMA/CA (Collision Avoidance) with Acknowledgement (ACK)	
Data Rate	High	11 Mb/s

	Medium	5.5 Mb/s
	Standard	2 Mb/s
	Low	1 Mb/s
	The cards use an automatic Transmit Rate Select mechanism.	

Table 3.4: Networking Characteristics of WL110 Wireless PC Card.

R-F Frequency Band	**2.4 GHz (2400-2500 MHz)**			
Supporting sub-channels	Channels vary between 1 and 11 (2412 MHz – 2462 MHz). Default is 10 (2457 MHz)			
Modulation Technique	Direct Sequence Spread Spectrum (DSSS) CCK 11 & 5.5 Mb/s, DQPSK for 2 Mb/s and DBPSK for 1 Mb/s			
Spreading	11-chip Barker Sequence			
Bit Error Rate (BER)	Better than 10^{-5}			
Normal Outlet Power	15 dBm			
Encryption	128-bit (RC4), also supports 64-bit RC4/WEP (Wired Equivalent Privacy)			
Range/Transmission Rate	**11 Mb/s**	**5.5 Mb/s**	**2 Mb/s**	**1 Mb/s**
Open Office	160 m (525 ft.)	270 m (885 ft.)	400 m (1300 ft.)	550 m (1750 ft.)
Semi-Open Office	50 m (165 ft.)	70 m (230 ft.)	90 m (300 ft.)	150 m (375 ft.)
Closed Office	25 m (80 ft.)	35 m (115 ft.)	40 m (130 ft.)	50 m (165 ft.)
Receiver Sensitivity	-83 dBm	-87 dBm	-91 dBm	-94 dBm
Delay Spread (FER of <1%)	65 ns	225 ns	400 ns	500 ns

Table 3.5: Radio Characteristics of WL110 Wireless PC Card.

The range of the wireless signal is associated with the Transmit Rate of the wireless communication. Communications at lower transmit rate will travel larger distances [Compaq2001].

Experiments in our work are done with channel 10, which is a default value.

3.2 Application System Design

3.2.1 Basic Implementation of Client and Server

Our main tool to develop this multi-hop ad hoc instant messenger is Microsoft .Net Compact Framework 2003 in C#. In C#, the *UdpClient* class is used to create a *Client* and a *Thread* is created as a *Server* to listen to arrivals of packets. The *Client* handles all the client functionalities including sending packets. UDP is chosen to send and receive packets, and its size is set to 512 bytes.

3.2.2 Implementation of the DSR Protocol

This Section describes the features of the DSR protocol [Johnson2001] implemented in our ad hoc messenger application.

3.2.2.1 Initial Route Discovery

The initial Route Discovery process uses the same procedures as the regular Route Discovery process. In our application, the following packets are used for the initial Route Discovery:

- *Start Chat Packet* (SCP): This packet is broadcast to all neighbors by a source to find a new route. When this packet reaches a destination node, the destination node will be asked to join the conversation with the source node.
- *Accept Start Chat Packet* (ASCP): This packet is sent from the destination node to the source node when the destination node agrees to chat with the source node. In the normal Route Discovery process, this packet takes the role of an acknowledgement (ACK) packet to the Start Chat Packet.
- *Reject Start Chat Packet* (RSCP): This packet is sent from the destination node to the source node when the destination node disagrees to chat with the source node. When this packet is received in the source node, the chat session will be stopped. In the normal Route Discovery process, this packet acts as an ACK to the Start Chat Packet.

When the destination node agrees to chat with the source node, all routes found through the initial Route Discovery packets (Start Chat Packets) will be stored in the source's cache. When the current route is broken, the source will try alternative routes in its route cache to retransmit the data packet. Only when there is no alternative route available in the route cache of a source node, will a new Route Discovery be performed. The new Route Discovery process is described in more details in the following subsection.

Note that the first route from a source to a destination always exists when the ACM (*All nodes are Connected with Multi-hops*) selection is turned on by the application. The ACM selection is an option allowing the application to be tested in a network environment in which all nodes are connected with multiple hops and will be explained with more details later.

3.2.2.2 Data Packet Transmission

After the initial Route Discovery is done and a chat session begins, the source node or destination node can transmit data packets containing conversation contents by using the following packets:

- *Data Packet* (DP): This packet includes conversation contents in its payload. A data packet can be sent only when a route is found. A data packet contains a route in its header. An intermediate node will just look up the route contained in the data packet, and then forward the data packet to the next node. When the source node sends a data packet, a copy of the data packet is stored in the *Send Buffer* of the source node. This copy of the original data packet is deleted when an ACK for the data packet is received from a target node.
- *Route Reply Packet*[1] (RRP): This is the reply packet (ACK) for a data packet. After the source node sends a data packet, the destination node will send a Route Reply Packet (RRP) to the source node as an ACK. The destination node uses the reverse path of the route contained in the data packet to send a RRP. Also, when the destination node receives a data packet from the source node, if the data packet is received out of order, it saves a copy of the received data packet in the *Receiver Window Buffer* with an ordered sequence number to receive and display data packets in order. By default, Sliding Window Size (SWS) is 1. That is, a *stop-and-wait* policy is used. Thus, only after a RRP is received in the sender, will

[1] Note that "a route reply packet" is different from "a reply route discovery packet". The former is an ACK with respect to a data packet, while the latter is a reply packet with respect to a route discovery packet. In the thesis, we use the term "a reply route discovery packet" interchangeably with the term "a route discovery reply packet," both referring to a reply packet to a route discovery packet.

the next data packet be transmitted. For the more general case that SWS is larger than 1, a data packet (which is not a data packet with the next sequence number that the destination node expects to receive) may be received out of order by the receiver. In this case, the receiver stores the out-of-order data packet received in its *Receiver Window Buffer*, and waits until the next data packet with the expected sequence number is received. After missing data packets (if any) have arrived, the destination node will display the chat content in order followed by the removal of these data packets from the *Receiver Window Buffer*. Therefore, users are guaranteed to see chat contents in the same order as the sender transmits. When the source node receives a RRP, the copy of data packet stored in the *Send Buffer* is removed. Note that the use of SWS of various sizes is a design alternative to be evaluated in the thesis.

- *Timeout for Two End-To-End Nodes*: There is a timeout period between the source and destination nodes, which is set as 5 seconds. In our application, a one-way latency for delivering a data packet between two nodes was 0.4 second and the round-trip latency was 0.8 seconds on average. Since we have 7 nodes (Pocket PCs) in our experiments, the timeout in the worst case (i.e. A-B-C-D-E-F-G) needs to cover 12 hops. Thus, we decided to set the end-to-end timeout period as 5 seconds (12*0.4 second = 4.8 seconds). The source node will wait for the end-to-end timeout period after it sends a data packet to the destination node. If the source node does not receive any ACK (through a Route Reply Packet) from the destination node within the end-to-end timeout, it will retransmit the same data packet. This process repeats three times, after which the source will conclude that a route error has occurred and will enter a Route Maintenance stage. In the next Section, we explain how a broken route is repaired in the Route Maintenance stage.

3.2.2.3 Route Maintenance

- *Receipt Packet* (RP): This packet is used to ensure that a data packet is received in the next node safely. That is, a Receipt Packet is used as an ACK between two neighbor hops[2]. Specifically, after a node forwards a data packet to the next node, if it does not receive a receipt packet from the next node within a hop-by-hop timeout period (1 second), a node

[2] A hop-by-hop receipt packet at the network layer may not be needed if the underlying link-layer subsystem can inform the sender node that a packet cannot be delivered to the next hop because the link between the sender node and the next hop breaks. The thesis does not consider this option because for the case we evaluate the system with a network topology simulator (to be described later), the simulator does not have link-layer mechanisms, and for the case without the use of a network topology simulator we do not have access to a link-layer API for IEEE 802.11 to allow the link layer subsystem to inform a sender node that a packet cannot be delivered to the next hop.

would consider that the link is broken or the next node has failed. Any intermediate node detecting such an error will inform the source node by means of a Route Error Packet (REP), after which the source node will perform a route repairing process (to be explained later). Receipt packets are used only when data packets are sent. The use of receipt packets is a design alternative to be evaluated in the thesis.

- *Route Error Packet* (REP): This packet is used by an intermediate node to inform the source node of a route error. After the source node receives a REP, the route in use is considered not valid any longer and will be removed from the source's cache. Then, the source node invokes a route repairing process to repair the broken route.
- *Route Repairing Process*: A route repairing process includes two route-repairing mechanisms. First, when there is any alternative route available in the route cache of a source node, the source node will use the route to resend the lost data packet (the source has not received an ACK for the data packet sent from the target) until there is no more available route in its route cache. If any of alternative routes works fine by receiving an ACK for the retransmitted data packet, the route will be used continuously to send subsequent data packets. Second, if there is no alternative route available in the route cache of the source node, the source node will perform a Route Discovery to find a new route. More details of Route Discovery stage will be explained later.

3.2.2.4 Use of Route Cache Structure

- *Route Cache*: A route cache is used to store all available routes from a source node to a destination node. When a Route Discovery is initiated, a Route Discovery packet is broadcast by the source node. Since there may be more than one route between the source node and the destination node, the source node may receive multiple route discovery reply packets from the destination node, in which case the source node stores all available routes in its cache. When the source needs to find a route to deliver data packets, it uses the shortest route among all routes it maintains in its cache. The use of cache structure is a design alternative to be evaluated in the thesis.

3.2.2.5 Route Discovery

The Route Discovery is performed when initially there is no route available, or there is a route error detected and no cached route is available. The following packets are employed to perform a new Route Discovery in repairing a broken route when no cached route is available.

- *Route Discovery Packet* (RDP): This packet is used by a source node to find a new route. This Route Discovery Packet (RDP) is flooded to all neighbors of the sender. An intermediate node, who does not find itself in the route record of the RDP and is not the destination node, will rebroadcast the RDP to its own neighbors. Finally, when the target node receives the RDP, it stores the accumulated route record (the route found by the RDP) in its route cache, and replies to the initiator of the RDP by sending a Reply Route Discovery Packet (RRDP) as an ACK.
- *Reply Route Discovery Packet* (RRDP): When a target node receives a RDP, it uses the reverse route of the route record contained in the received RDP to send a Reply Route Discovery Packet (RRDP) as an ACK to the source. Once the source receives the RRDP, it saves the route contained in the RRDP in its route cache.

3.2.2.6 Hang-Up

The Hang-Up stage is the final stage when a chat session is ended. The operation of Hang-Up Packet (HUP) and Reply Hang-Up Packet (RHUP) is the same as Data Packet (DP) and Route Reply Packet (RRP). The following packets are utilized as follows:

- *Hang-Up Packet* (HUP): This packet is sent when a user wants to end a chat session. In our prototype ad hoc chat program (*UserChat*), after a Hang-Up Packet (HUP) is sent, the interface is brought back to the first screen that contains the buddy list to start a new chat session with a new buddy. After the buddy user receives this packet, she or he will see the message that the source buddy ends this chat session. Then, the buddy also goes back to the buddy list screen to start a new chat session with a new buddy.
- *Reply Hang-Up Packet* (RHUP): This packet is an ACK for a HUP. When a source node sends a HUP to its buddy in the chat session, the destination buddy node returns a RHUP to the initiator of the HUP. After then, the destination buddy node is reset to the beginning state. Also, after the source node receives a RHUP from its buddy destination node, it is also reset to the beginning state. Therefore, when any node wants to start a new chat session with any other node, a new route should be established because all nodes in the ad hoc

network do not hold any route information obtained from the previous sessions at the beginning.

3.2.2.7 Other Components

- *Sliding Window Size* (SWS): The SWS parameter is used to specify a *Sender Window Size* and a *Receiver Window Size*. We set Sender Window Size and Receiver Window size to be of the same size specified by the SWS parameter. By default, SWS is 1 representing the *stop-and-wait* policy [Peterson2000]. In this case, a source node cannot send the next data packet until it receives a Route Reply Packet (RRP) as an ACK from the destination. One of our experiments will measure the effect of SWS on the performance of our application. In that experiment, SWS varies between 1 and 6.
- *All nodes are Connected with Multi-hops* (ACM): In our experiments, we simulate movements of nodes in the mobile ad hoc network using the Random Waypoint Model (RWM) and generate a dynamic topology configuration file (conf.xml) as a function of time to be followed by all nodes in a single simulation run. In some cases, the dynamic network generated may partition a node from all other nodes. That is, there may be cases in which there is no route between a source and destination as time progresses, which makes it difficulty to test the effectiveness of Route Discovery and Route Maintenance mechanisms in DSR and design alternatives used to implement our instant messenger application. Therefore, we allow the ACM selection option by which a route always will exist to connect two nodes at all times despite movements of nodes in the network. Two rules are used to achieve the condition. First, all nodes are connected without any node being isolated from the network. During the generation of a dynamic topology configuration file, if we find a node is being isolated from the network, we force it to connect to some neighbor nodes. In that case, a node having the smallest number of neighboring nodes among all participating nodes is selected to connect with the isolated node. Second, all nodes should be connected with multi-hops. This is implemented by limiting the number of neighbor nodes around each node not to exceed one half of the participating nodes, i.e., 3 (7/2 = 3 as an integer) in our experiments. Therefore, when a node is connected with more than one half of the participating nodes, we artificially remove some links around it, starting with a node having the largest number of neighbors. In short, in order for all nodes to be properly connected through multi-hops, ACM modifies the network topology by forcing an isolated node to be connected, and a node having too many neighbors to cut some of its links. Table 3.6 shows an

example of unadjusted and adjusted network topologies generated at a time instant based on the random waypoint model (GEN application) with speed = 80 m/s and pause time = 0 second for the case in which the range for wireless transmission is 40 m, the simulation area is 200m (height) by 200m (width), seed is 3, and the total number of participating nodes is 7.

Neighbors of Unadjusted Network Topology		Neighbors of Adjusted Network Topology	
A		A	G, D
B	D	B	D, C
C		C	B, F
D	B	D	B, A
E	F	E	F
F	E	F	E, C
G		G	A

Table 3.6: Example of Adjusted Network Topology by ACM.

Figures 3.3 and 3.4 show network topologies for both unadjusted and adjusted network topologies obtained from the GEN application based on the Random Waypoint Model (RWM), corresponding to Table 3.6.

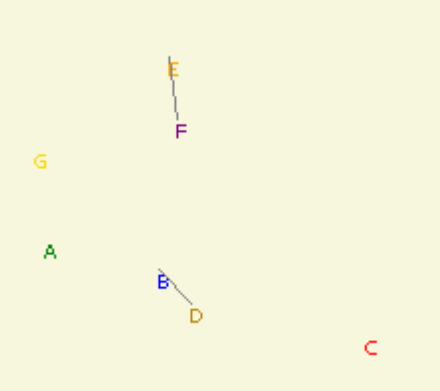

Figure 3.3: Unadjusted Network Topology.

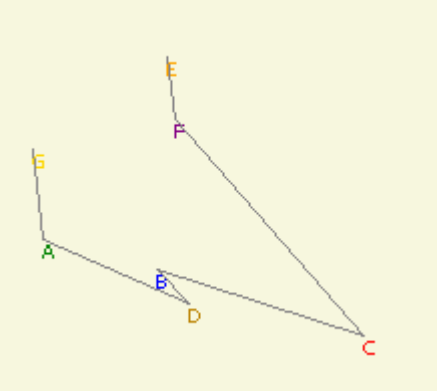

Figure 3.4: Adjusted Network Topology by ACM.

3.2.3 XML-Based Packet Components

This Section illustrates each component of a packet used in our multi-hop ad hoc instant messenger.

3.2.3.1 Regular Packet Format

The regular packet format is used for communication between a source node and a destination node. A regular packet contains 10 fields (or components) as follows:

- *Type*: Type of packet. There are 11 types of packets used in our application using the regular packet structure, as shown in Table 3.7.

Type of Packet	Usage
Start Chat Packet (SCP)	This packet is sent when a user wants to chat with a buddy node.
Accept Start Chat Packet (ASCP)	This packet is sent as an ACK for the Start Chat Packet when a receiver agrees to joining the chat session.
Reject Start Chat Packet (RSCP)	This packet is sent as an ACK for the Start Chat Packet when a receiver disagrees to joining the chat session.
Data Packet (DP)	This packet is sent when a data packet containing conversation content is transmitted.
Route Reply Packet (RRP)	This packet is sent as an ACK to the initiator of a data packet.
Route Discovery Packet (RDP)	This packet is sent when a new route needs to be found.
Reply Route Discovery Packet (RRDP)	This packet is sent as an ACK to the initiator of a Route Discovery Packet (RDP).
Hang-Up Packet (HUP)	This packet is sent when a user wants to end a chat session.
Reply Hang-Up Packet (RHUP)	This packet is sent as an ACK for Hang-Up Packet (HUP).
Route Error Packet (REP)	This packet is sent by an intermediate node to inform a source node about a route error.
On Calling Packet (OCP)[3]	Since a node only engages in one chat session at a time, a node currently in a chat session will reject a new chat session request made by another buddy node by sending an On Calling Packet (OCP) to it. For example, when node **A** and node **C** are chatting, node **B** or other nodes cannot talk

[3]While in real situations, a user can chat with multiple users simultaneously, our wireless ad hoc messenger only allows one node to chat with another node due to lack of screen space in a Pocket PC.

	with **A** or **C**. When other nodes try to contact **A** or **C**, **A** or **C** sends an OCP to the nodes in order to let the other buddy nodes know they are on a chat session. However, among other nodes that are not on a chat session, they can chat with each other. In the above example, other nodes except **A** and **C** can have a chat session. For instance, node **B** and node **D** can have a chat.

Table 3.7: Types of Regular Packets.

- *Initiator*: Source's ID.

- *Target*: Destination's ID.

- *Request ID*: Unique ID to distinguish each packet.. The Request ID is comprised of a four-tuple as follows:

  < *Session Number, Source ID, Destination ID, Sequence Number* >

Session Number: Randomly selected unique number to distinguish each chat session.
Sequence Number: When a data packet is transmitted, a sequence number is used to process each data packet in order at the receiver. The sequence number starts from 1 and is incremented by one as more data packets are sent. When a Route Discovery packet is transmitted, a different sequence number is used. The sequence number for Route Discovery packets starts from – 4 and decremented by 1 as more Route Discovery packets are sent [4]. This sequence number for Route Discovery packets is utilized to measure latency to find a new route.

- *Route Record*: Historical record of route, which is passed from a source to a destination node, as explained in the previous chapter regarding the description of the DSR protocol (Chapter 2). From the source node to the destination node, the previously traversed node's ID is accumulated to the Route Record slot. For example, if there is a route from **A** (source) to **E** (destination) such as **A-B-C-D-E** in the Route Discovery stage, the Route Record slot will be

[4] A negative sequence number is used for a route discovery packet to distinguish from the sequence number of a data packet; -4 used as the initial sequence number for route discovery packets is an arbitrary choice.

added with **A** in a source node, **A: B** in node **B**, **A: B: C** in node **C**, **A: B: C: D** in node **D**, and **A: B: C: D: E** in node **E**. By doing this, node **E** will have a complete path from **A** to **E** and can send an ACK by using the reverse route of the Route Record back to the initiator of Route Discovery packet, **E: D: C: B: A** in this example.

- *Current ID*: Current node's ID. This information is used when a receipt packet is sent to the previous node.

- *Payload*: Conversation content in a data packet.

- *Route Cache*: Route obtained from the Route Cache of a source node. If a cached route is available, a source node adds the cached route in this Route Cache slot of a data packet. Since this slot includes a whole path from a source to a destination, intermediate nodes just can forward a data packet to the indicated next node based on this Route Cache information. Route Cache returns the shortest route whenever a node requests a route.

- *Receipt ID*: ID for Receipt Packet to differentiate each receipt packet from all incoming receipt packets. This Receipt ID is used to remove timeout condition for each receipt packet. The Receipt ID consists of a three-tuple as follows:

  < Current Node ID, Next Node ID, Unique Integer Number>

  Unique Integer Number: This number is randomly selected, but does not overlap with already selected numbers used in Receipt ID to identify each receipt packet.

- *Time Stamp*: Time at which a sender transmits a packet.

3.2.3.2 Receipt Packet Format

The Receipt Packets are used for hop-by-hop packet delivery confirmation. Three fields are used as follows:

- *Type*: Receipt Packet

- *Receipt ID*: Same as Receipt ID in a regular packet.

- *Time Stamp*: Time at which a sender transmits a packet.

3.2.3.3 Examples of a Regular Packet and a Receipt Packet in XML Format

```
<Packet>
<Type>DP</Type>
<Initiator>A</Initiator>
<Target>G</Target>
<RequestID>3452:A:G:1</RequestID>
<CurrentID>D</CurrentID>
<RouteRecord>A:C:D</RouteRecord>
<RouteCache>A:C:D:G</RouteCache>
<ReceiptID>D:G:6785</ReceiptID>
<TimeStamp>18</TimeStamp>
</Packet>
```

Figure 3.5: XML Structure of Data Packet.

```
<Packet>
<Type>RP</Type>
<ReceiptID>D:G:6785</ReceiptID>
<TimeStamp>19</TimeStamp>
</Packet>
```

Figure 3.6: XML Structure of Receipt Packet.

Figure 3.5 shows an example data packet in XML format. The packet type specifies that the packet is a data packet. The "Request ID" field indicates that 3452 is the *session ID* of the chat session, the source node is **A,** the destination node is **G**, and the packet is the first data packet with a sequence number of 1. The current node is **D** and the current route record is updated to include the current node in the partial route found. The "Route Cache" field indicates that the previously Route Discovery process has found a new route **A-C-D-G**. Thus, this data packet will be forwarded to **G** based on the information of Route Cache. The "Receipt ID" field is used to indicate that **D** should receive a receipt packet from the next node, **G**. The time at which node **D** sends this packet to **G** is at T=18 seconds, which is the time elapsed since the session begins.

Figure 3.6 also illustrates the structure of a receipt packet in XML format. This receipt packet is for packet delivery confirmation between **D** and **G** in which 6785 is a unique integer number to distinguish each receipt packet. The time at which node **D** sends this receipt packet to **G** is at T=19 seconds, as indicated in the "Time Stamp" field.

3.2.4 Network Topology Design

Since the actual testing environment for an ad hoc network is difficult to set up, the Random Waypoint Mobility Model has been employed to pre-generate a dynamic network topology as a function of time for nodes to follow in a simulation run. This Section gives a brief overview of the Random Waypoint Model and describes how our GEN application generates simulated network topologies based on different input values.

3.2.4.1 Random Waypoint Mobility Model

The Random Waypoint Mobility Model contains pause time between changes in direction and/or speed [Camp2002b, Bettstetter2003]. Once a Mobile Node (MN) begins to move, it stays in one location for a specified pause time. After the specified pause time is elapsed, the MN randomly selects the next destination in the simulation area and chooses a speed uniformly distributed between the minimum speed and maximum speed. Then, the MN continues its journey toward the newly selected destination at the chosen speed. As soon as the MN arrives at the destination, it stays again for the indicated pause time before repeating the process [Camp2002b].

Camp et al. give a demonstration of an example traveling pattern of an MN generated by the Random Waypoint Mobility Model, as shown in Figure 3.7 [Camp2002b]. The pattern starts at a randomly selected point (133, 180) with the speed of the MN uniformly selected from 0 m/s to 10 m/s.

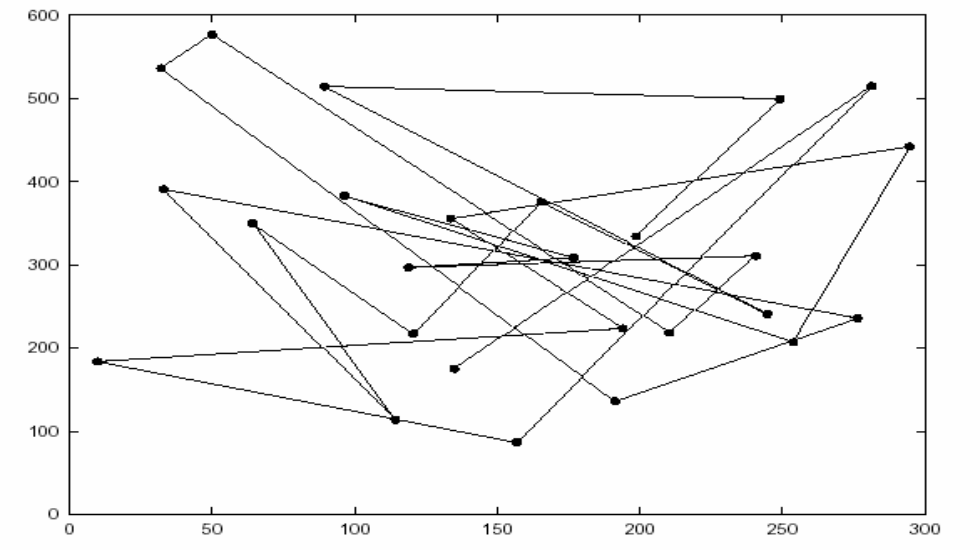

Figure 3.7: Traveling Pattern of a Mobile Node Using the Random Waypoint Model.

When MNs move based on the Random Waypoint Model, there is a complicated relationship between node speed and pause time [Camp2002b]. For instance, a scenario with a higher speed and longer pause time generates a more stable network than a scenario with a lower speed and shorter pause time. Long pause times, for example over 20 seconds, generate a stable network topology, such as few link changes per MN, even at high speeds, as demonstrated in Figure 3.8. In other words, even though speed is very high, if pause time is over 20 seconds, this random waypoint model produces a very stable network. Thus, in order to set up the very fluctuating network scenario by mainly controlling speed, high speed and very short pause time like 0 second are required.

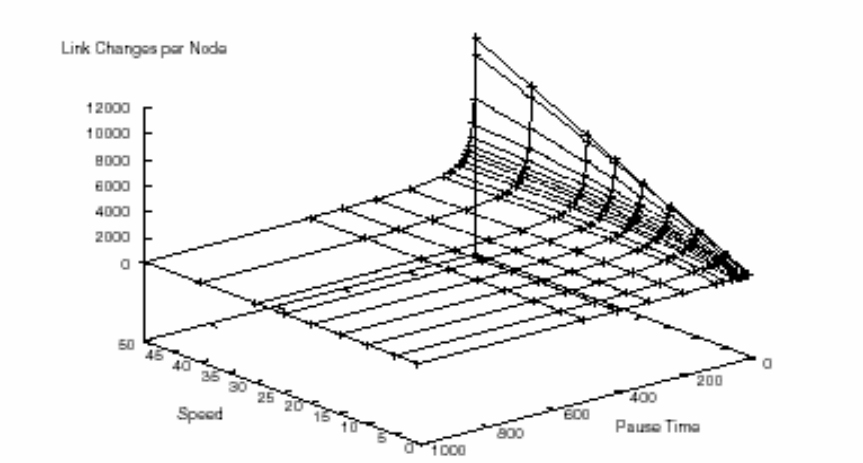

Figure 3.8: Link Breakages vs. Speed vs. Pause Time.

3.2.4.2 GEN Application

We developed a GEN application to generate a dynamic topology configuration file to be followed by all nodes based on the random waypoint mobility model. This Section discusses how GEN creates a configuration file to simulate a dynamically changing network and describes how we use XML to describe the configuration file (output of GEN) and the basic information file (input to GEN).

3.2.4.2.1 User Interfaces of the GEN Application

The GEN program is to be loaded in each Pocket PC. According to the scenario of each experiment, the same configuration file will be loaded in each Pocket PC. The following shows the user interfaces of the GEN program:

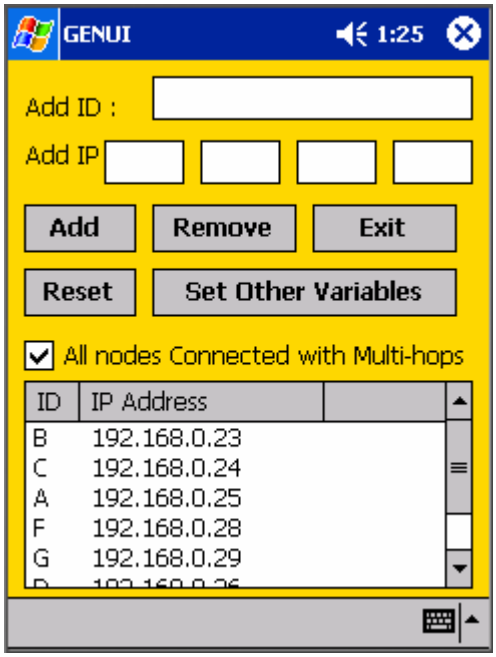

Figure 3.9: User Interface of the GEN (1).

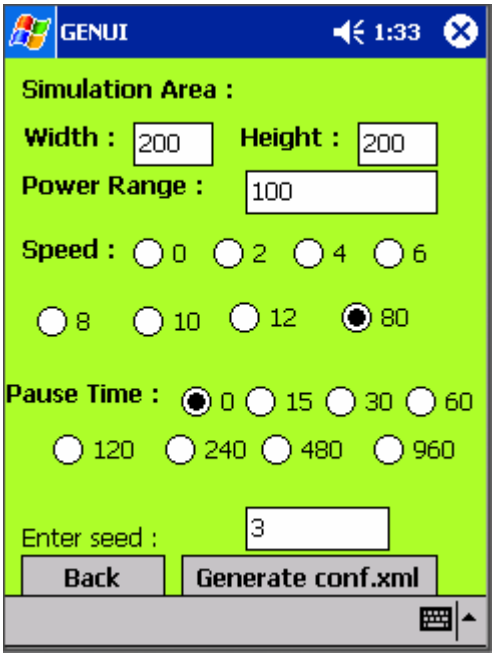

Figure 3.10: User Interface of the GEN (2).

Figure 3.9 is the first user interface we can see when the GEN program is loaded. A user can add or remove or reset a node's ID and IP address. After appropriate information is added and the user wants to set other variables, such as a simulation area, power range, speed, and pause time, etc., the user may click the button named the "*Set Other Variables*." The "*Set Other Variable*" button brings the second interface shown in Figure 3.10. Simulation area and power range (transmission range) can be entered manually. A user can also manipulate speed and pause time. The specified pause time and speed will be applied to all randomly selected destinations, which are randomly selected positions within the chosen simulation area.

After proper input values are chosen, the "*Generate conf.xml*" button shows the third interface, illustrated in Figure 3.11. Figure 3.11 shows the moving pattern of each mobile node based on the random waypoint model. The forth interface of the GEN program in Figure 3.12 demonstrates network topology at a specified time (T) equal to 3 seconds, which means that 3 seconds are elapsed since the session begins. Also, ACM is used for all nodes to be connected with multi-hops as a default condition.

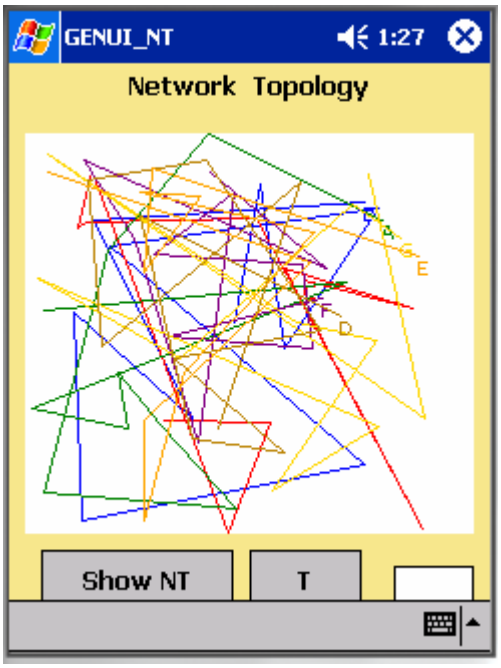

Figure 3.11: User Interface of the GEN (3).

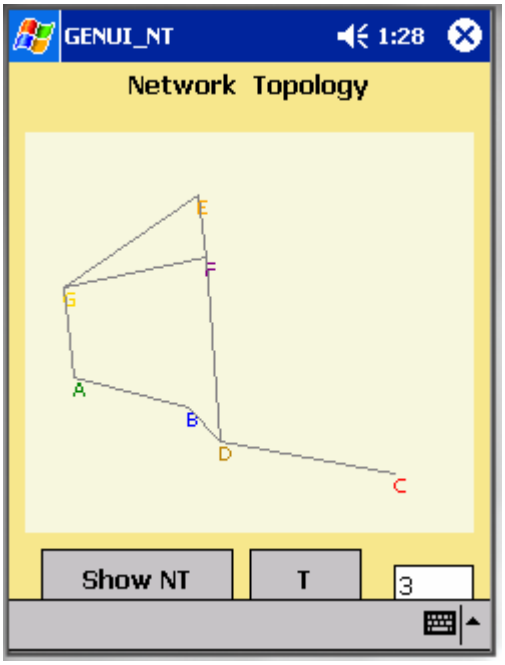

Figure 3.12: User Interface of the GEN (4).

3.2.4.2.2 Network Configuration File in XML Format

The network configuration file (conf.xml) will contain randomly selected 10 destination positions (i.e. (x, y)) each with speed and pause time. Instead of selecting each random destination whenever each node moves as explained in Section 3.2.4.2 based on the Random Waypoint Mobility Model, 10 destination positions are pre-generated. After each node moves all 10 destinations, then it returns to the first destination and continues to move to the next destination.

As explained in the User Interface of the GEN Application in previous subsection, a user can manipulate speed and pause time in the configuration file. Each node (Pocket PC) will see the same configuration file and move in the same simulated network. In the simulated network, each node stays at one destination location for specified pause time, and then moves to the next destination location with the specified speed. Since our experiments will be implemented based on various speed and pause time, a user can control speed and pause time by entering them as input values.

Figure 3.13 shows an example network configuration file in XML format based on the random waypoint model. This configuration file is generated by the GEN application. Appendix C.1 gives a true example of the network configuration file (conf.xml).

```
<Configuration>
  <Node ID="B" IP="192.168.0.23" StartPt="(169,34)">
    <Sequence>
      <Hop>
        <Destination>(34,43)</Destination>
        <Speed>12</Speed>
        <Pause>0</Pause>
      </Hop>
      <Hop>
      .
      .
      9 more hops are generated.
      .
      .
      </Hop>
    </Sequence>
  </Node>
  .
  .
  6 more nodes are generated.
  .
  .
</Configuration>
```

Figure 3.13: Example Configuration File for Simulated Network Topology Based on the Random Waypoint Model.

```
<GEN_BasicInfo>
  <NumOfNodes>7</NumOfNodes>
  <SimulationArea>
    <Width>200</Width>
    <Height>200</Height>
  </SimulationArea>
  <PowerRange>100</PowerRange>
  <Speed>12</Speed>
  <PauseTime>0</PauseTime>
</GEN_BasicInfo>
```

Figure 3.14: Example Basic Information File of the GEN.

3.2.4.2.3 Basic Information File in XML Format

To describe the input parameter values to the GEN application, another XML file is generated based on the chosen input values from the second interface shown in Figure 3.10.

Figure 3.14 illustrates an example XML-based file containing basic information selected for the GEN application. A true example of this XML file is given in Appendix C.2.

3.2.5 User Interfaces of the Chat Application

The DSR protocol is implemented in our prototype ad hoc messenger (*UserChat*) with the following default conditions:

- Use of Cache Structure: On

- Use of Configuration File for the Network Topology: On
- Use of Receipt Packet: Off
- ACM (All nodes are Connected with Multi-hops): On
- SWS (Sliding Window Size): 1

Note that the use of the configuration file for simulating a dynamic network topology can be on or off based on the user's preference in the real chatting prototype.

The prototype application of multi-hop ad hoc messenger using Pocket PCs has the following two user interfaces:

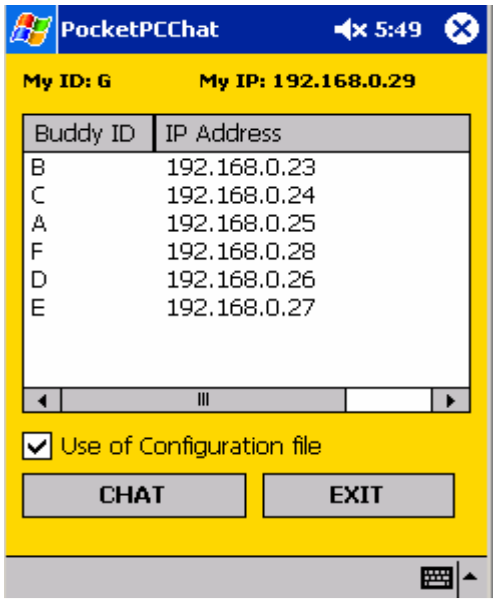

Figure 3.15: User Interface of the Pocket PC Chat - (1) Start Chat Stage.

Figure 3.16: User Interface of the Pocket PC Chat - (2) Chat Stage.

Figure 3.15 shows the first interface a user sees when the application is loaded. The "Buddy List" displays IDs and IP addresses of all other participating nodes except its own. Figure 3.16 is the second interface when a chat session begins.

A use scenario is as follows:

1. When node **A** (source) wants to chat with node **G** (destination), **A** selects **G** in the Buddy List.
2. **G** has the option to accept or reject the chat request from **A**, as displayed in Figure 3.17.

3. **A** may or may not have a chat session by receiving "*Accept*" message or "*Reject*" message from a destination node, **G**, as Figure 3.18 and Figure 3.19 show.
4. If **G** accepts the chat request from **A**, **A** and **G** can have a conversation, as Figure 3.20 and Figure 3.21 demonstrate.
5. When **A** or **G** want to end the conversation, either of them can click the "*Hang-Up*" button. Then, the buddy node will have the message that the conversation is ended, as Figure 3.22 illustrates.
6. After the chat session is closed, both parties automatically go back to the first interface showing a buddy list to chat with a new buddy.

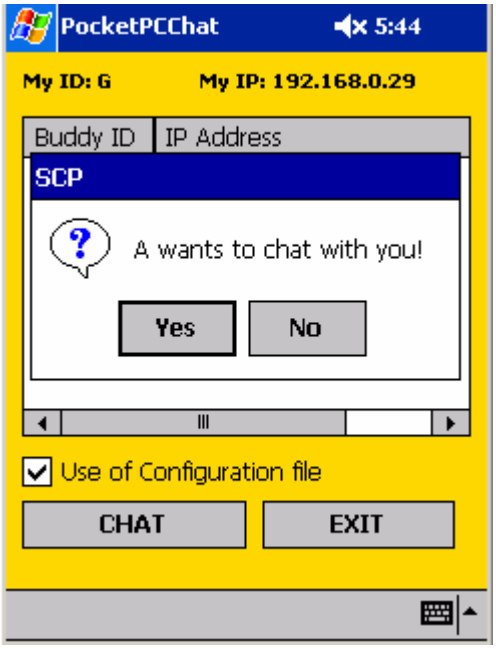

Figure 3.17: User Interface of the Pocket PC Chat – (3) Accept or Reject the Chat Request from the Source in the Destination.

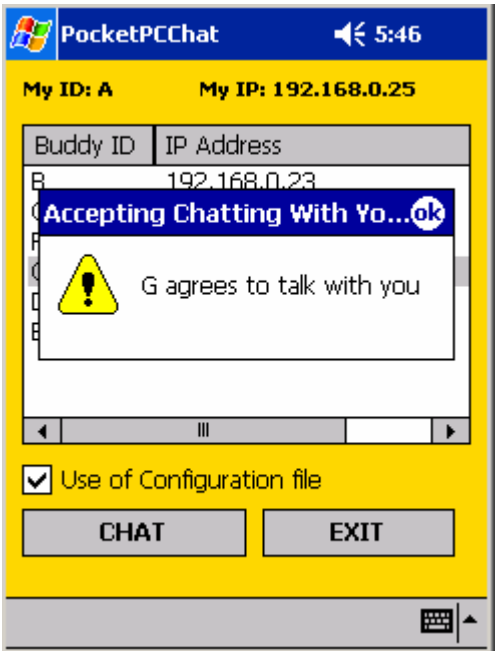

Figure 3.18: User Interface of the Pocket PC Chat – (4) Chat Request Accepted.

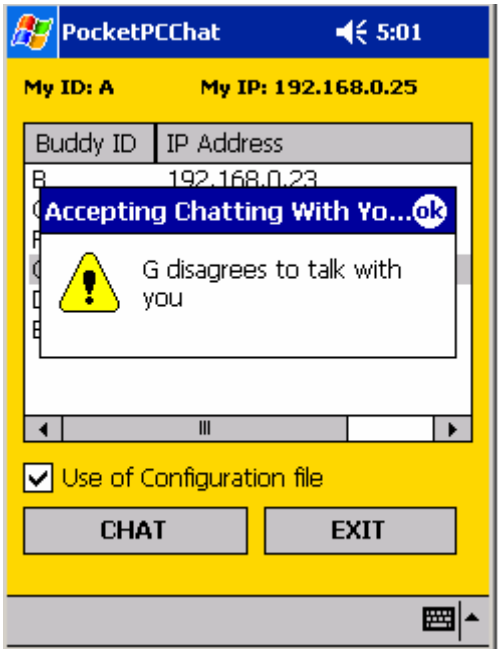

Figure 3.19: User Interface of the Pocket PC Chat – (5) Chat Request Rejected.

Figure 3.20: User Interface of the Pocket PC Chat – (6) Chat Session in A.

Figure 3.21: User Interface of the Pocket PC Chat - (7) Chat Session in G.

Figure 3.22: User Interface of the Pocket PC Chat – (8) Chat Session Ended.

3.2.6 Design for the Automatic Data Collection Application

To facilitate automatic data collection, another version of the wireless ad hoc messenger application is created. The underlying design and implementation is exactly the same as the prototype ad hoc messenger discussed earlier except that it is augmented with automatic testing and data collecting capabilities. The Automatic Data Collection application provides a user interface to allow a tester to specify scenario conditions. Figures 3.23 and 3.24 illustrate a use scenario:

1. A source node selects a buddy randomly by clicking the "*Select buddy*" button.
2. All other conditions can be selected using check boxes, such as ACM (All nodes are Connected with Multi-hops), use of cache structure, and use of receipt packet.
3. A user can enter Sliding Window Size (SWS) and the total number of data packets to be transmitted (for the chat contents) in each text box.
4. When the "*Start*" button is clicked, one packet containing all selected options and input values is broadcast to all nodes including its own. All nodes set the same scenario conditions selected and synchronize their time clock [5]so that all nodes follow the same dynamic network topology based on the configuration file generated by the GEN at the same time.
5. After clock synchronization, the source node starts sending data packets until all are transmitted.
6. After all data packets are transmitted and the source node receives all ACKs for the transmitted data packets, the session is closed and proper result files are generated to measure the performances of our ad hoc messenger under the specified scenario.
7. The result files generated in each Pocket PC are transmitted through a program named the "Send And Receive Files," to a laptop. A program named the "Statistical Analyzer" running on the laptop then takes all the result files from each node and produces X-Y graphs to analyze the impact of the selected environmental or design condition on the performance of our application.

[5] Note that the synchronization process here refers to all nodes starting an experiment simultaneously. There may be clock drifts that exist in different mobile devices, which we do not consider in the thesis. Also we do not move mobile devices in an experiment, but sit them all in the same room with each of them reading in the same configuration file and following the mobility pattern defined in the configuration file to emulate node movements in an experiment. Therefore, the actual physical distance separating two mobile nodes is mush shorter than the virtual distance separating the same two nodes defined in the configuration file.

Figures 3.23 and 3.24 show snapshots of the user interface provided by the Automatic Data Collection application. More detailed user interfaces for the Send And Receive Files and Statistical Analyzer for achieving wireless file transfer and analysis of result files are listed in Appendix B.1 and Appendix B.2, respectively.

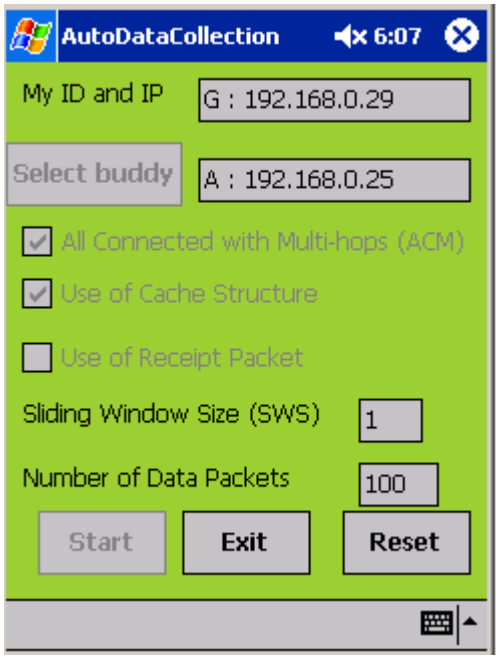

Figure 3.23: User Interface of the Automatic Data Collection - (1) Destination G.

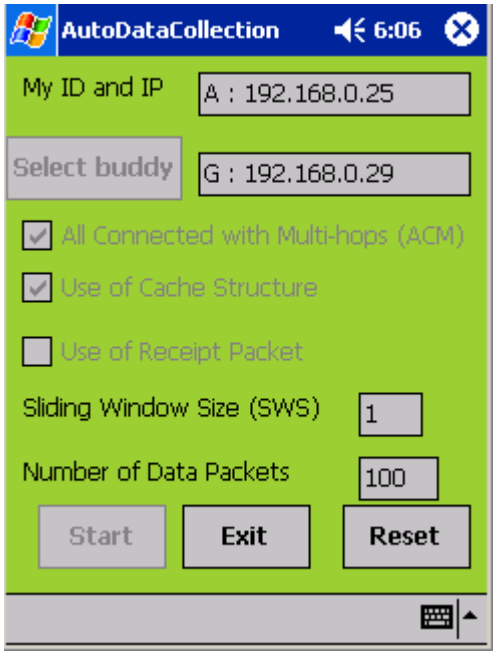

Figure 3.24: User Interface of the Automatic Data Collection - (2) Source A.

3.3 Performance Evaluation: Input Parameters and Metrics

This Section describes input parameters to both the GEN and Automatic Data Collection applications, and defines performance metrics to evaluate the performance of our wireless ad hoc messenger application.

3.3.1 Input Parameters

For the GEN Application:

- *Pause Time*: In the random waypoint model, each node stays at a location for a pause time period. Our application uses the following values to manipulate pause time:
  5, 30, 60, 120, 240, 480, 960 (second).

The default pause time is 60 seconds.

- *Speed*: In the random waypoint model, each node moves to a next location with a specified speed. Speed can be selected in our application among the following values:

  0, 2, 4, 6, 8, 10, 12, 80 (m/sec).

  The default speed is 80 m/s.
- *Simulation Area*: The network area for each node's movement can be entered manually. The width and height of the simulation area should be specified. The default simulation area is 200m (Width) by 200m (Height), which is equal to 40,000 m^2.
- *Power Range* (Transmission Range): Within a single-hop transmission range, nodes are connected with each other. This value can be entered manually. The default power range is 100 m.
- *Seed*: We use different seeds to generate different topology configuration files. We also conduct each experiment for a specific scenario three times (three rounds) to observe statistical variances, so each experiment uses three different seeds to generate three different configuration files.

For the Automatic Data Collection Application:

- *ACM (All nodes are Connected with Multi-hops)*: On or Off
- *Use of Cache Structure*: On or Off
- *Use of Receipt Packets*: On or Off
- *Total Number of Data Packets to Send*: Can manually enter. The default value is 100.
- *SWS (Sliding Window Size)*: Can manually enter. The default value is 1.

3.3.2 Metrics Used

We select five metrics to evaluate the performance of our wireless ad hoc messenger, namely, *average latency to find a new route, average latency to deliver a data packet, delivery ratio of data packets, normalized control overhead, and throughput.*

3.3.2.1 Average Latency to Find a New Route

This metric indicates the average delay experienced to find a new route. This metric is chosen to measure the performance of our application because a short delay to find a new route in the presence of dynamically changing network topologies is a good indication that our application functions well. There are two situations to estimate the average latency to find a new route, as summarized in Table 3.8 and Table 3.9:

- *When receipt packets are used:*

Scenarios	Start Time	End Time
1. When a route error occurs, an alternative route in the source's cache is valid.	When a source receives a Route Error Packet (REP).	When a Route Reply Packet (RRP) is received by the source after it uses an alternative route in its cache to retransmit the data packet.
2. When a route error occurs, alternative routes in the source's cache are not valid.	When a source receives a Route Error Packet (REP).	Since alternative routes in the source's cache are not valid, a new Route Discovery is performed. Then, when a Reply Route Discovery Packet (an ACK for the Route Discovery Packet) is received in the source.
3. When no route is available in the source's cache and the source does not receive any packets, such as a Route Error Packet or Route Reply Packet.	When a source performs a new Route Discovery.	When the source receives a Reply Route Discovery Packet (an ACK for the Route Discovery Packet).
4. When a source has not received any packets, such as a Route Error Packet or Route Reply Packet after sending the same data packet three times with the timeout interval for an end-to-end acknowledgement, the source	When a source sends a data packet using an alternative route in its cache since the data packet is not delivered to the target node after three trials.	When a source receives a Route Reply Packet for the data packet sent using the alternative route in its cache.

has alternative routes in the cache.		

Table 3.8: Scenarios for Average Latency to Find a New Route When Receipt Packets Are Used.

- *When receipt packets are not used:*

Scenarios	Start Time	End Time
1. The source has not received a Route Reply Packet for a data packet sent after three trials. It is recognized as a route error. Then, the source uses a valid route in its cache.	When a source sends a data packet using an alternative route in its cache.	When a source receives a Route Reply Packet for the data packet sent using the alternative route in its cache.
2. There is the same scenario as Scenario 1 except that there are no valid routes available in the source's cache.	When a source performs a new Route Discovery.	When the source receives a Reply Route Discovery Packet (an ACK for the Route Discovery Packet).

Table 3.9: Scenarios for Average Latency to Find a New Route When Receipt Packets Are Not Used.

3.3.2.2 Average Latency to Deliver a Data Packet

This metric indicates the delay experienced to successfully deliver a data packet from a source to a destination. This metric is selected also because it measures how our application functions under dynamically changing network topologies.

More specifically, this average latency to deliver a data packet is the average delay for one-way trip of a data packet from a source to a destination. Therefore, in order to gauge this one-way trip time of a data packet, this latency starts when a data packet is sent by the sender and ends when a data packet is received by the receiver.

3.3.2.3 Delivery Ratio of Data Packets

This metric indicates how many data packets (including retransmission due to errors) are transmitted in order to deliver the same number of data packets. For example, when there are 10 data packets to be delivered but 20 data packets are actually transmitted, then the delivery ratio is

50% since 10 data packets are transmitted but not delivered. This data delivery ratio is selected also to measure the performance of our application when the topology is rapidly changing causing links to be broken easily. This metric is defined by the following function:

Delivery Ratio of Data Packets =

Total number of data packets delivered / Total number of data packets transmitted.

3.3.2.4 Normalized Control Overhead

This metric represents how many control packets are transmitted per data packet delivered. Since constrained resources are an issue in an ad hoc network, using fewer control packets per data packet is desirable to help reduce the use of bandwidth, battery power of each mobile device, etc. The normalized control overhead is computed by the following formula:

Normalized Control Overhead =

Total number of control packets used / Total number of data packets delivered.

Control packets are used for Route Discovery and Route Maintenance. Examples of control packets are Route Reply Packet, Route Error Packet, Start Chat Packet, Accept Start Chat Packet, Reject Start Chat Packet, Route Discovery Packet, Reply Route Discovery Packet, Hang-Up Packet, and Reply Hang-Up Packet. We do not consider receipt packets as control packets because they are just used for hop-by-hop delivery confirmation. In a simulation experiment, we collect control packets sent by each node in order to avoid double counting.

3.3.2.5 Throughput

This metric shows how many data packets are delivered per time unit (second). It is used to measure the degree of parallelism allowed by the system to handle the delivery of multiple data packets simultaneously. The throughput is calculated as follows:

Throughput = Total number of data packets delivered / Total time elapsed.

The *total time elapsed* starts when the first Route Discovery packet (Start Chat Packet) is sent and ends when the ACK of the last data packet (Route Reply Packet for the last data packet sent) is received by the sender. When the throughput is high, it means that more packets can be transmitted per second or less time will be spent to deliver a fixed number of data packets.

3.4 XML Files Used for Describing Results of Experiments

There are three XML files used to describe experimental results from the Automatic Data Collection application as follows: ADC_basicInfo.xml, Result_(NodeID).xml, and DataAnalysis_(NodeID).xml.

Figure 3.25 shows an example file of ADC_basicInfo.xml obtained from the Automatic Data Collection application. This file contains basic information about selected nodes and options, such as source node, target node, use of cache structure, SWS, etc. After this file is moved to the Statistical Analyzer running on the laptop, the information in this file is looked up when result graphs are generated.

```
<ADC_BasicInfo>
  <Source>A:192.168.0.25</Source>
  <Target>G:192.168.0.29</Target>
  <ACM>True</ACM>
  <Cache>True</Cache>
  <SWS>1</SWS>
  <NumOfDataPacket>100</NumOfDataPacket>
  <ReceiptPacket>False</ReceiptPacket>
</ADC_BasicInfo>
```

Figure 3.25: Example of Basic Information File from the Automatic Data Collection.

```
<DataAnalysis ID="A" SourceNode="Yes" TimeNow="1687">
  <XParam>
    <SWS>1</SWS>
    <Speed>80</Speed>
    <PauseTime>15</PauseTime>
  </XParam>
  <Throughput>0.06</Throughput>
  <TotalDataPacketDelivered>100</TotalDataPacketDelivered>
  <AvgLatencyDeliveryingDP>15.19</AvgLatencyDeliveryingDP>
  <AvgLatencyFindingNewRoute>6.08</AvgLatencyFindingNewRoute>
  <DeliveryRatioOfDP>0.34</DeliveryRatioOfDP>
  <AvgNumOfCaCheEntries>0.91</AvgNumOfCaCheEntries>
  <TotalNumberOfCPSent>180</TotalNumberOfCPSent>
</DataAnalysis>
```

Figure 3.26: Example of Data Analysis File in the Source Node from the Automatic Data Collection.

Figure 3.26 displays an example file of DataAnalysis_A.xml generated from the Automatic Data Collection application. This source node's file is mainly looked up by the Statistical Analyzer application in the laptop to produce proper result graphs according to a set of selected x-parameter and y-parameter. An example file in a non-source node is given in Figure 3.27.

```
<DataAnalysis ID="G" SourceNode="No" TimeNow="1688">
  <TotalNumberOfCPSent>372</TotalNumberOfCPSent>
</DataAnalysis>
```

Figure 3.27: Example of Data Analysis File in a Non-Source Node Generated from the Automatic Data Collection Application.

As most information needed for data analysis can be obtained from the source node except the total number of control packets sent, the files from non-source nodes do not have to contain all other information like components of the file from the source node. However, since the total number of control packets sent should be collected from all nodes participating in the network, it is recorded in each DataAnalysis_(NodeID).xml even for non-source nodes.

Figure 3.28 shows an example file of Result_A. xml that includes the history of all packets sent and received by the source node. This file is used as a sanity check that all processes are functioning properly based on the design of the DSR protocol in our application. As illustrated in Figure 3.28, each result packet consists of 7 fields as follows:

- *ComType*: Communication type. "*S*" for a sent packet or "*R*" for a received packet.

- *Type*: Type of packet.
- *Request ID*: Request ID of packet.
- *Route Cache Used*: Route used for communication.
- *Num of Route In Cache*: Number of available routes in the cache.
- *Time Stamp In Sender*: Time at which the current packet is sent.
- *Time Stamp In Receiver*: Time at which the current packet is received.
- *Not Transmitted DP*: Data packet that is not transmitted yet. This field is used to measure latency to find a new route when the current packet type is Route Discovery Packet or Reply Route Discovery Packet or Route Error Packet.

```
<Result NodeID="A" NodeIP="192.168.0.25" TimeNow="1686">
  <Packet ComType="S">
     <Type>SCP</Type>
     <RequestID>83693:A:G:0</RequestID>
     <RouteCacheUsed>N/A</RouteCacheUsed>
     <NumOfRoutesInCache>0</NumOfRoutesInCache>
     <TimeStampInSender>1</TimeStampInSender>
     <TimeStampInReceiver>N/A</TimeStampInReceiver>
     <NotTransmittedDP>N/A</NotTransmittedDP>
  </Packet>
  .
  .
  .
  .
  All sent or received packets are recorded.
  .
  .
  .
  .
  <Packet ComType="R">
     <Type>RRP</Type>
     <RequestID>83693:A:G:100</RequestID>
     <RouteCacheUsed>G:A</RouteCacheUsed>
     <NumOfRoutesInCache>1</NumOfRoutesInCache>
     <TimeStampInSender>1685</TimeStampInSender>
     <TimeStampInReceiver>1686</TimeStampInReceiver>
     <NotTransmittedDP>N/A</NotTransmittedDP>
  </Packet>
</Result>
```

Figure 3.28: Example of Result File in the Source Node Generated from the Automatic Data Collection Application.

3.5 Summary

This Chapter detailed the methodology used for the design, implementation and evaluation of the wireless ad hoc messenger application using Pocket PCs based on the DSR protocol.

First, we discussed our development environment based on seven Pocket PCs with manually assigned IP addresses to evaluate the performance of our wireless ad hoc messenger based on the DSR protocol. For wireless communications between participating nodes, WLAN cards with 802.11b P2P capability are utilized. The main tool to develop the application is Microsoft Visual Studio .Net C# 2003, Microsoft .Net Compact Framework 2003, and Microsoft Pocket PC 2002 SDK. For the communication between Pocket PCs and a laptop, Microsoft ActiveSync is employed.

Second, we discussed our design methodology for the implementation of DSR with basic features and alternative designs including the use of XML to describe data and receipt packets, the use of a dynamic network topology configuration file generated by the GEN application using the random waypoint model to account for node mobility, the interface design for a real ad hoc messenger, and the use of automatic data collection to facilitate testing and performance analysis.

Third, for performance evaluation we described our user interface design that allows a tester to specify input parameters to reflect environmental and operational conditions and test their effect on system performance. We also defined five performance metrics, i.e., average latency to find a new route, average latency to deliver a data packet, delivery ratio of data packets, normalized control overhead, and throughput, as the basis for performance evaluation.

Finally, we discussed our design methodology in using XML to describe experimental results to facility data exchanges between processes that generate data and those that analyze data and display results so that performance analysis after experiments are done can be performed automatically.

In Chapter 4, experimental results will be presented and analyzed in terms of the five selected performance metrics.

Chapter 4
Experimental Results and Analysis

We have performed a series of experiments to evaluate the performance of the ad hoc messenger application with respect to the following factors: speed, pause time, Sliding Window Size (SWS), use of receipt packets, and use of cache structure. These factors are used to recognize the sensitivity of our wireless ad hoc messenger implemented with the DSR protocol under alternative designs and network environments. In particular, the first two parameters (speed and pause time) reflect the network operating environments, while the last three parameters (SWS, use of receipt packets, and use of cache structure) represent design alternatives in implementing our ad hoc messenger application.

First, we vary the speed and pause time to control the stability of the network topology generated by the random waypoint model. In the random waypoint model, the next location to which each node moves is randomly generated. The speed with which to move to the next location is also randomly generated. Then a pause time is also randomly generated specifying the time the node stays at the next location. The cycle repeats. In manipulating the network environment, when the speed varies (changing between 0 m/s and 12 m/s), the pause time will be set as 0 second in order to get rid of its effect on the changing network topology. More specifically, since the random waypoint model produces a somewhat stable network topology even with a very high speed and a relatively long pause time (for example, over 20 seconds), the default pause time is set to 0 second in order to eliminate its effect. Also, when the pause time varies (changing between 15 seconds and 960 seconds), the speed will be fixed at one default value, 80 m/s, in order to eliminate its effect. The reason to select 80 m/s as the default speed is that the pause time alone can control the stability of the network topology when each node moves from a current position to the next position very quickly.

Second, we vary the Sliding Window Size (SWS) parameter to test the system's capability to allow multiple data packets to be transmitted simultaneously. When the network bandwidth is abundant and connectivity is stable, increasing SWS will allow more data packets to be transmitted simultaneously and thus will increase throughput due to parallelism. However, in an ad hoc network where nodes are resource constrained and bandwidth is limited, increasing SWS may cause more errors in packet delivery and may have an adverse effect on throughput. We will evaluate the effect of SWS on the performance of our wireless ad hoc messenger in a dynamically changing network topology.

Third, we examine the effect of using receipt packets so as to detect route errors more quickly for hop-to-hop communication. Even though the IEEE 802.11 MAC protocol tries retransmission for a number of times, a receipt packet is still needed to inform the occurrence of a route error to the source node. However, in a bandwidth-limited and resource-constrained ad hoc network, the use of receipt packets may degrade the system performance needlessly because of a higher traffic load introduced into the network. We will evaluate the effect of receipt packets on the system performance.

Finally, we examine the effect of using cache. Using cache has been reportedly to improve the performance of the DSR protocol. When cache is used, it means that a source node's route cache can store more than one route acquired from Route Discovery Reply Packets returned by the destination node (as replies to a Route Discovery packet). We design an experiment to evaluate if using cache is beneficial or harmful in the presence of a changing network topology.

4.1 Default Values for Input Parameters

4.1.1 Fixed Parameters

The fixed default input values in all experiments are as follows:

- Simulation Area: Width (m) * Height (m) = 200 (m)*200(m) = 40,000 m²
- Power Range (Transmission Range): 100 m
- Total Number of Data Packets to Transmit: 100

4.1.2 Changeable Input Parameters

The default values for changeable input parameters in all experiments are as follows:

- Speed: 80 m/sec.
- Pause Time: 60 sec.
- All nodes are Connected with Multi-hops (ACM): On
- Use of Receipt Packets: Off
- Sliding Window Size (SWS): 1
- Use of Cache Structure: On

The above input parameters are changeable in order to set up a specific scenario for each experiment. More specifically, five input variables are manipulated to establish a specific scenario, including the speed, pause time, use of receipt packets, SWS, and use of cache structure. The speed and pause time are manipulated to control the rate at which a network topology changes. The use of receipt packets, SWS, and the use of cache structure are manipulated as design alternatives to the DSR protocol implemented in our wireless ad hoc messenger.

4.2 Performance of Messenger with respect to Node Speed

This Section reports the effect of speed on the performance of the wireless ad hoc messenger application.

4.2.1 Scenario

In this experiment, all parameters assume their default values except the speed parameter, which changes from 0 m/s to 12 m/s in increment of 2:

- Speed: 0, 2, 4, 6, 8, 10, 12 (m/s)

As explained in Chapter 3 (Chapter 3 on Methodology), the random waypoint mobility model generates a stable network even at a high speed if the pause time is long (i.e. over 20 seconds). In order to eliminate the possibility that a stable network is generated because of a long pause time, the pause time is set as 0 second. Therefore, in this experiment, “speed” is the only parameter that controls the network stability. In general, a low speed will generate a stable network while a high speed will generate a frequently changing topology.

4.2.2 Results

4.2.2.1 Speed vs. Average Latency to Find a New Route

Figure 4.1 shows how increasing the speed affects the average latency to find a new route. Figure 4.1 displays the mean value obtained from three different experiment rounds using three different configuration files for the network topology. Correspondingly, Figure 4.2 displays results for these three different rounds.

As expected, the average latency increases as the speed increases since the random waypoint model produces a very dynamic network with the node speed is high, resulting in a high average latency to find a new route.

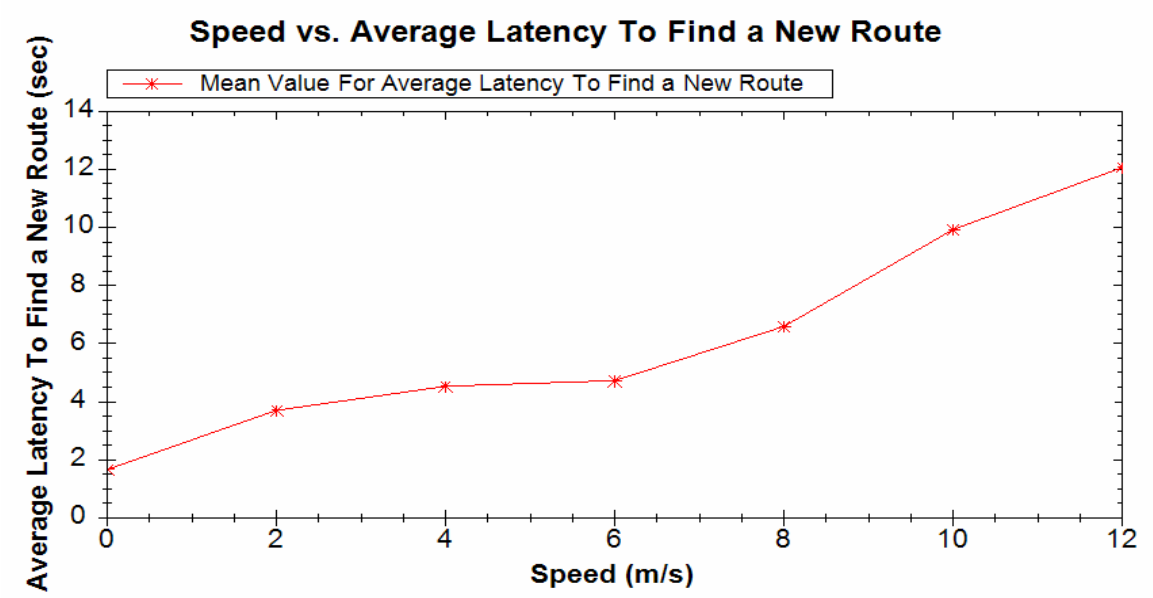

Figure 4.1: Speed vs. Mean Average Latency to Find a New Route.

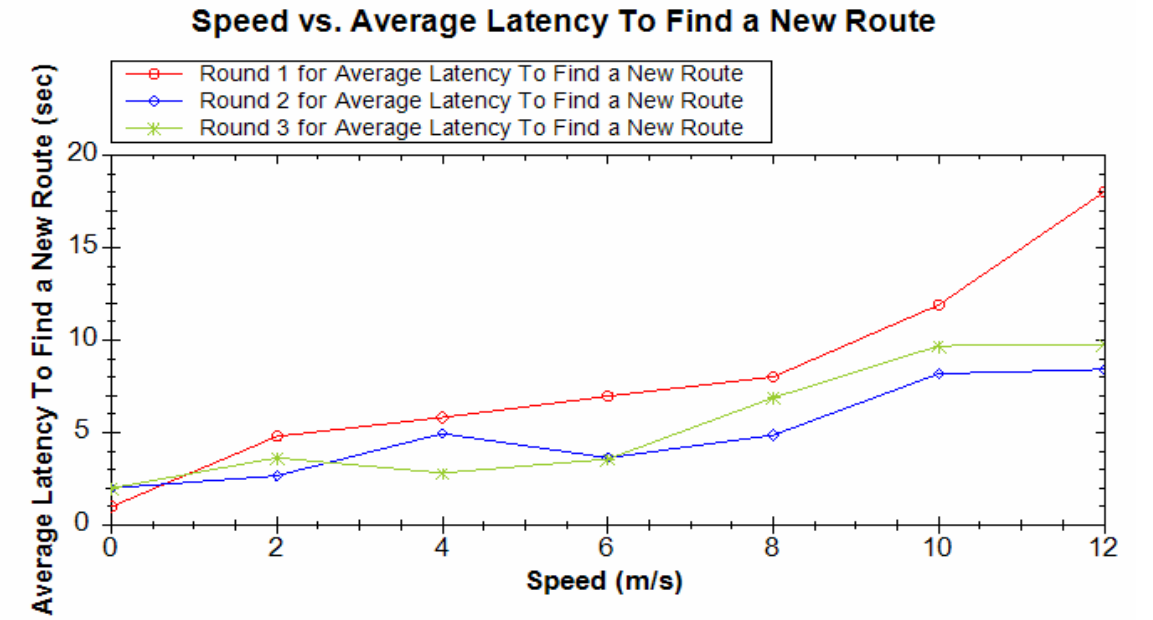

Figure 4.2: Speed vs. Average Latency to Find a New Route for Three Separate Rounds.

4.2.2.2 Speed vs. Average Latency to Deliver a Data Packet

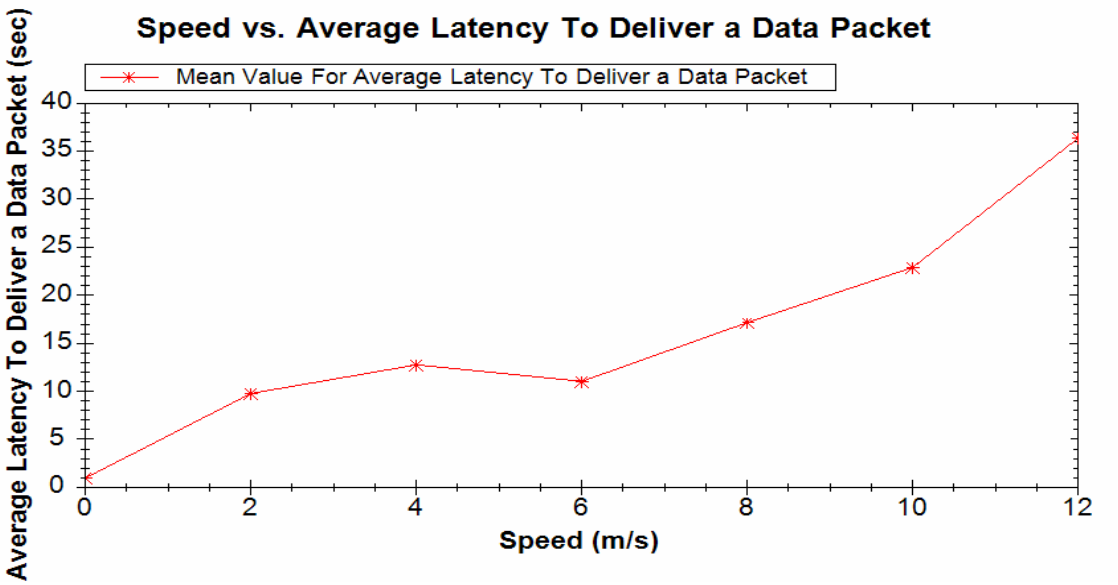

Figure 4.3: Speed vs. Average Latency to Deliver a Data Packet.

Figure 4.3 shows the average latency to deliver a data packet increases as the speed increases. The reason again is due to the fact that the random waypoint mode produces a more fluctuating network at a high node speed. Thus, it takes more time for a data packet to reach the destination node in an unstable network topology.

4.2.2.3 Speed vs. Delivery Ratio of Data Packets

Figure 4.4 shows that a network topology generated with a high node speed and a short pause time negatively influences the delivery ratio of data packets. That is, as the speed increases, the network becomes more and more dynamic, and the delivery ratio of data packets decreases because more packets need to be retransmitted due to errors.

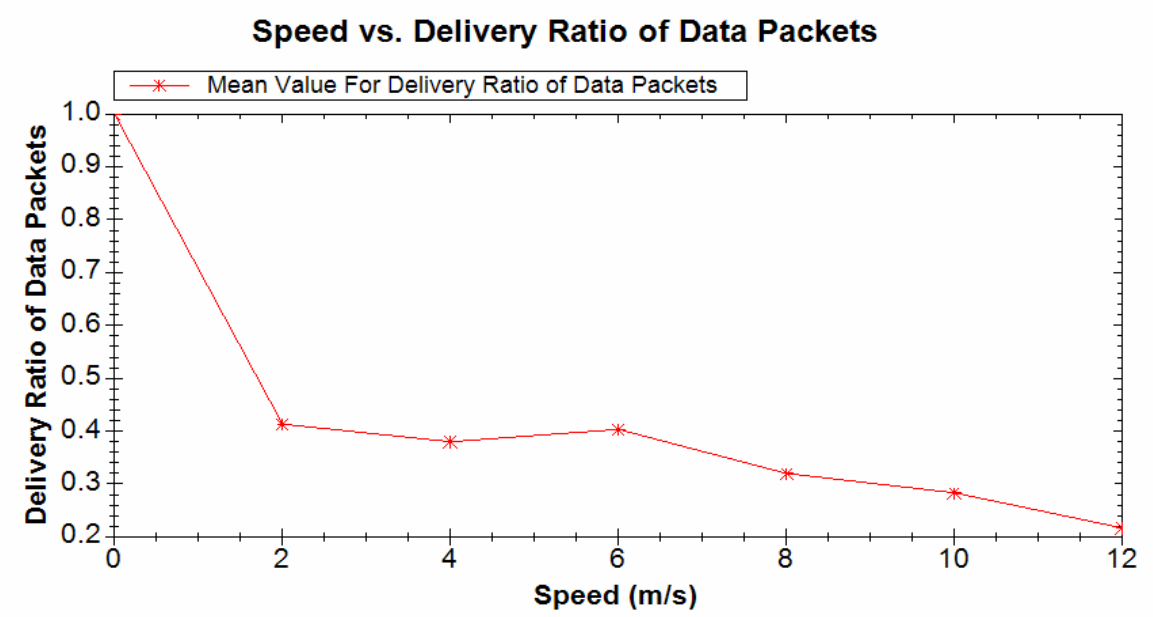

Figure 4.4: Speed vs. Delivery Ratio of Data Packets.

4.2.2.4 Speed vs. Normalized Control Overhead

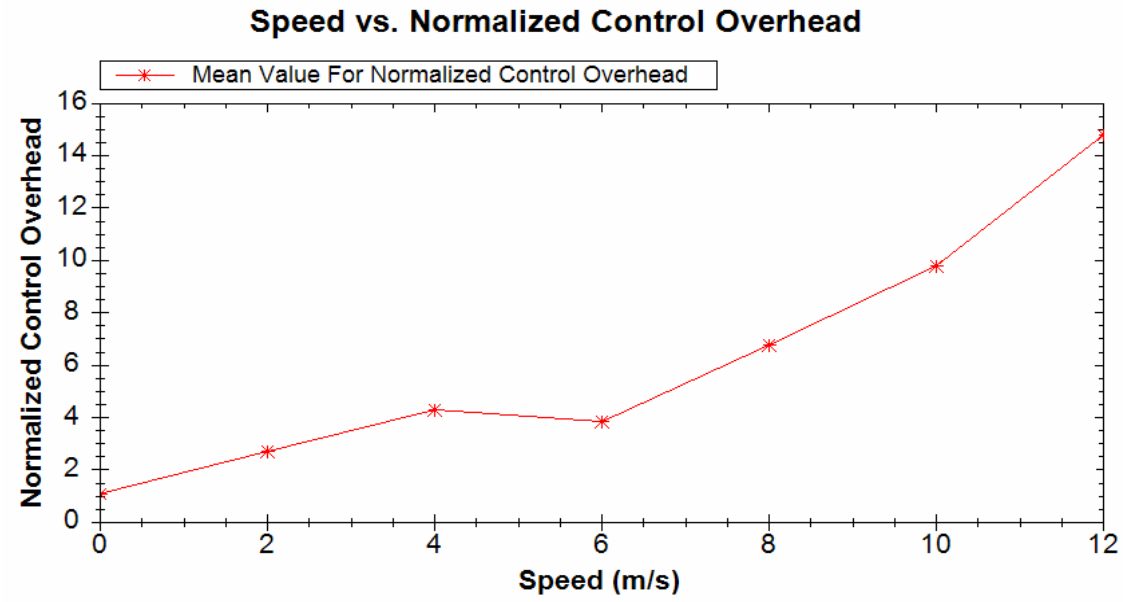

Figure 4.5: Speed vs. Normalized Control Overhead.

Figure 4.5 demonstrates that the normalized control overhead increases as the node speed increases. The reason is again due to the fact that the random waypoint model generates a highly dynamic network with a high node speed and a short pause time, causing more control

packets to be used to deliver the same number of data packets. That is, because there will be more broken routes when nodes move with a high speed, so more control packets, such as Route Discovery packets, may be used to repair broken routes.

4.2.2.5 Speed vs. Throughput

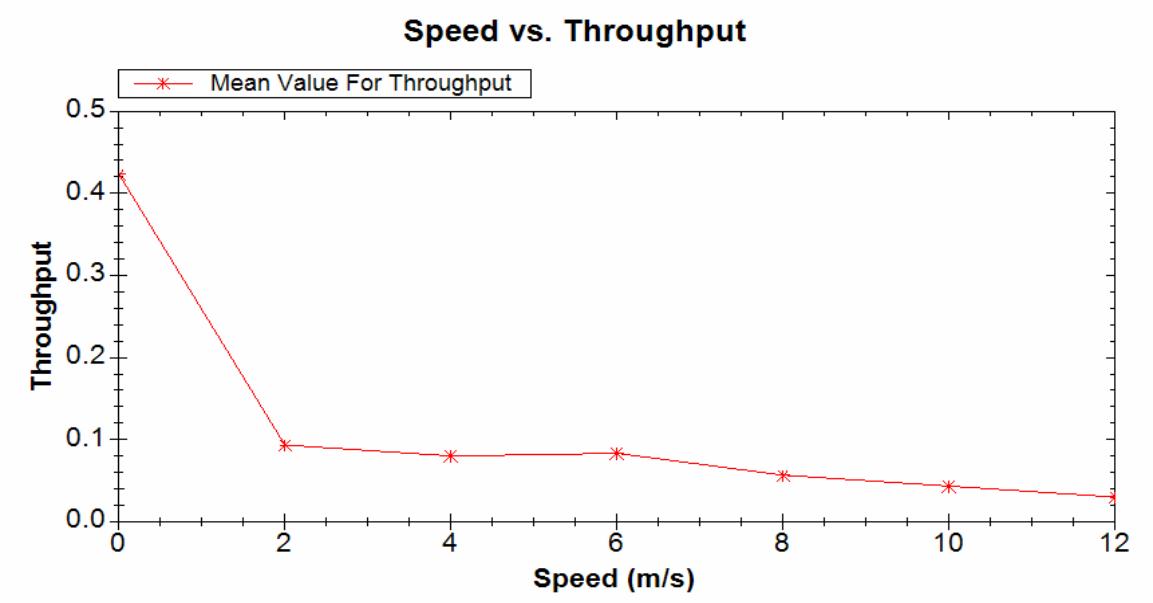

Figure 4.6: Speed vs. Throughput.

Figure 4.6 shows the negative effect of increasing the node speed on system throughput since a network with nodes moving with a high speed can break routes easily, thus causing the number of data packets delivered per second to be decreased.

4.2.3 Summary of Result

The experiment to test the effect of node speed on the performance of our wireless ad hoc messenger shows that our wireless ad hoc messenger performs better when the speed is low. In this experiment, the node speed is the only factor to control the stability of network topology using the random waypoint model. We observe that as the speed decreases, the network generated is more stable and our wireless ad hoc messenger will result in a shorter latency to find a new route, a shorter latency to deliver a data packet, a higher delivery ratio of data packets, a lower normalized control overhead, and a higher throughput.

4.3 Performance of Messenger with respect to Pause Time

4.3.1 Scenario

In this experiment, all parameters assume their default values except the pause time parameter, which varies between 15 seconds and 960 seconds as follows:

- Pause Time: 15, 30, 60, 120, 240, 480, 960 (seconds)

4.3.2 Results

4.3.2.1 Pause Time vs. Average Latency to Find a New Route

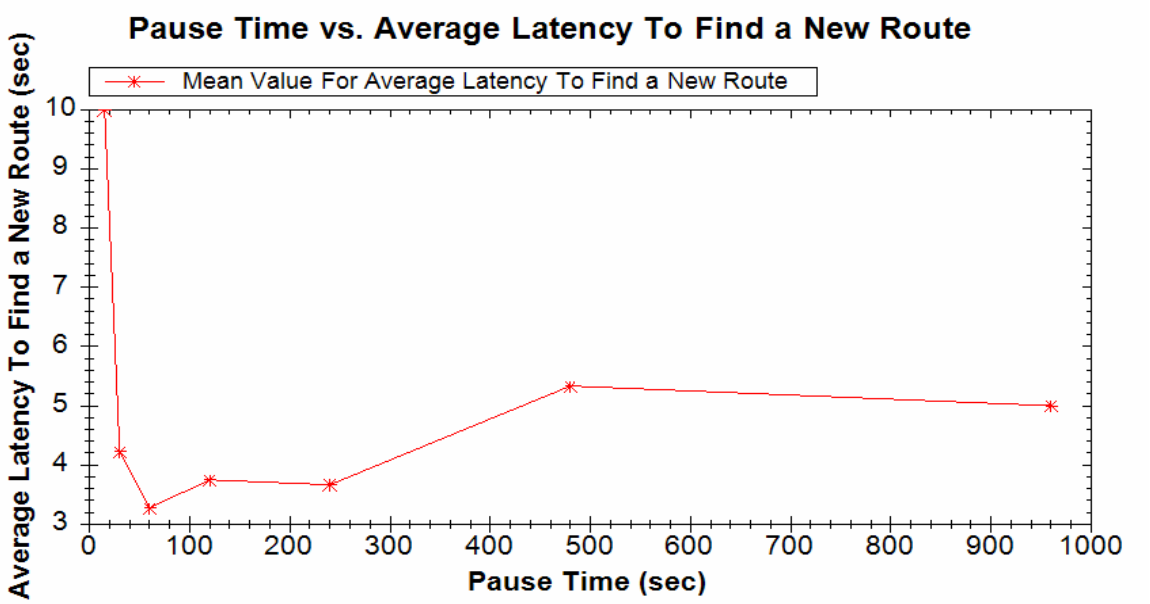

Figure 4.7: Pause Time vs. Average Latency for Finding a New Route.

Figure 4.7 shows the mean average latency obtained from three experiment rounds in which the pause time is the x-parameter and the average latency for finding a new route is the y-parameter. Figure 4.7 illustrates that a source node takes the most time to find a new route when the pause time is short, which is 15 seconds in this experiment. When the pause time increases to 60 seconds, the average latency decreases dramatically. However, the latency increases again when the pause time increases further.

The reason for the latency time behavior is that when the pause time is sufficiently long (>240 seconds in this case), the network generated by the random waypoint model is very stable,

so the latency to find a new route is attributed to the first Route Discovery process without any cache entry being available in the source node. When the pause time decreases to 60 seconds in this experiment scenario, the latency to find a new route is calculated out of the average of that due to the first Route Discovery process, and those due to the use of cache entries. As a result, the average latency is the minimum. As the pause time decreases further to 15 seconds, the network generated is very dynamic for any cache entries to be useful and a new Route Discovery process is required which has to take a long time to find a new route in a dynamic network, resulting in the average latency to be very high.

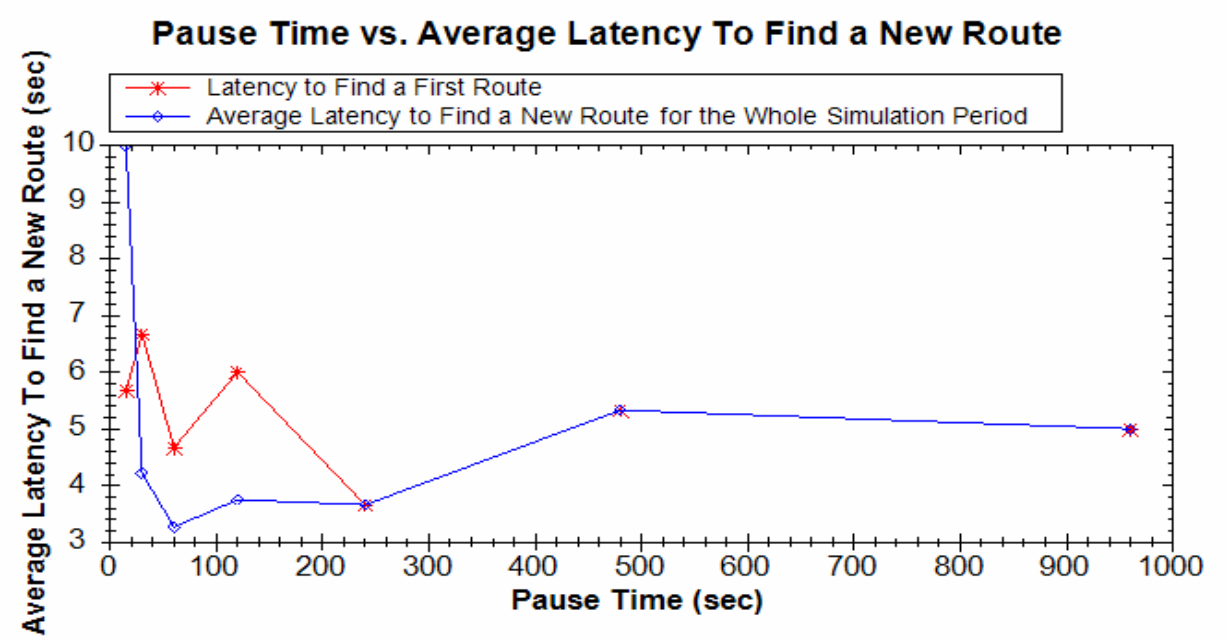

Figure 4.8: Comparison Graph of Pause Time vs. Average Latency to Find a New Route and Latency to Find the First Route.

We verify the reason given above by reporting the latency to find the "first" route as shown in Figure 4.8. When the pause time is 15 seconds, the network topology is very frequently changing, causing all cache entries stored in the source node to be invalid. Thus, when the pause time is 15 seconds, it incurs more time to find a new route than to find the first route because of the extra time used to try invalid cache entries. This is indicated by the very long "average" latency compared with the much smaller latency for finding the first route when the pause time is 15 seconds. When the pause time increases to 60 seconds, we see that the average latency is at its minimum because most cache entries now are valid and can be utilized to decrease the amount of time to find a route when performing a Route Discovery process. This is

also verified in Figure 4.8 where we see the latency of the first route is higher than the average latency when the pause time is 60. Finally, when the pause time is 240 seconds or higher, the average latency to find a new route and the latency to find the first route are the same, as shown in Figure 4.8. Since the Route Discovery process when the pause time is sufficiently high (> 240 seconds) is only performed in the initial Route Discovery stage (because the network topology generated is very stable and not changed throughout the entire experimental period), the source node does not need to perform any more Route Discovery process after the initial Route Discovery.

4.3.2.2 Pause Time vs. Average Latency to Deliver a Data Packet

Figure 4.9 shows the average latency to deliver a data packet decreases as the pause time increases. The average latency to deliver a data packet dramatically drops as the pause time becomes sufficiently high (> 240 seconds) beyond which the network generated becomes very stable so the latency to deliver a data packet becomes very low.

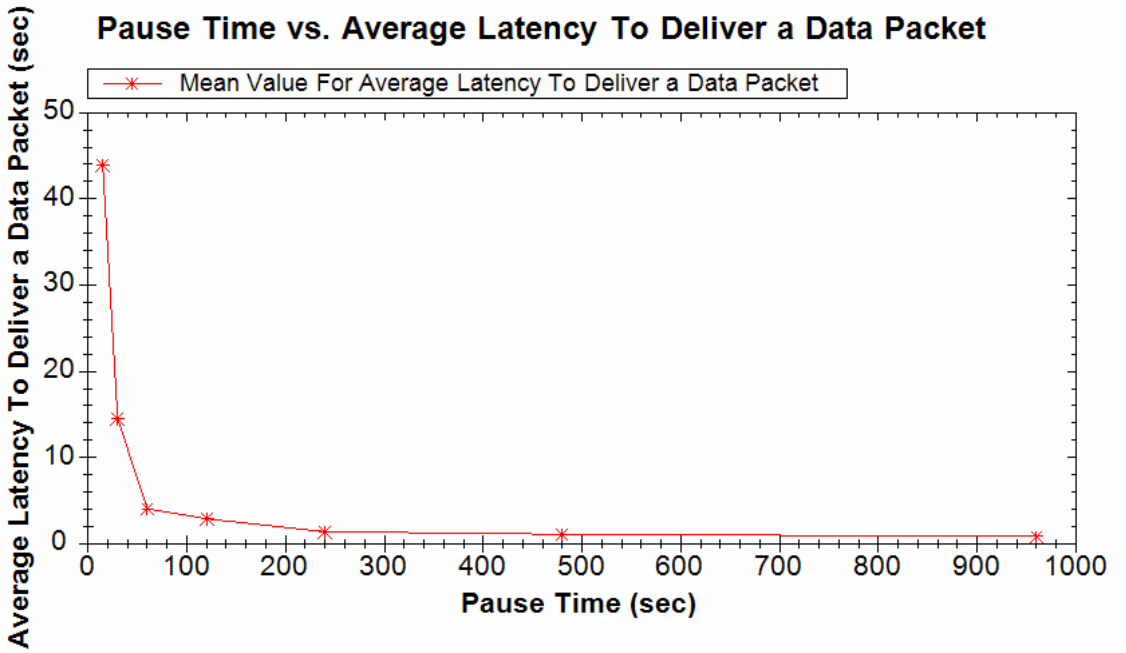

Figure 4.9: Pause Time vs. Average Latency to Deliver a Data Packet.

4.3.2.3 Pause Time vs. Delivery Ratio of Data Packets

Figure 4.10 shows that the delivery ratio of all transmitted data packets noticeably increases as pause time increases. When the pause time is sufficiently high (>240 seconds), the delivery ratio of data packets is almost 100%. That is, when 100 data packets are sent, 100 data packets are successfully delivered without retransmission.

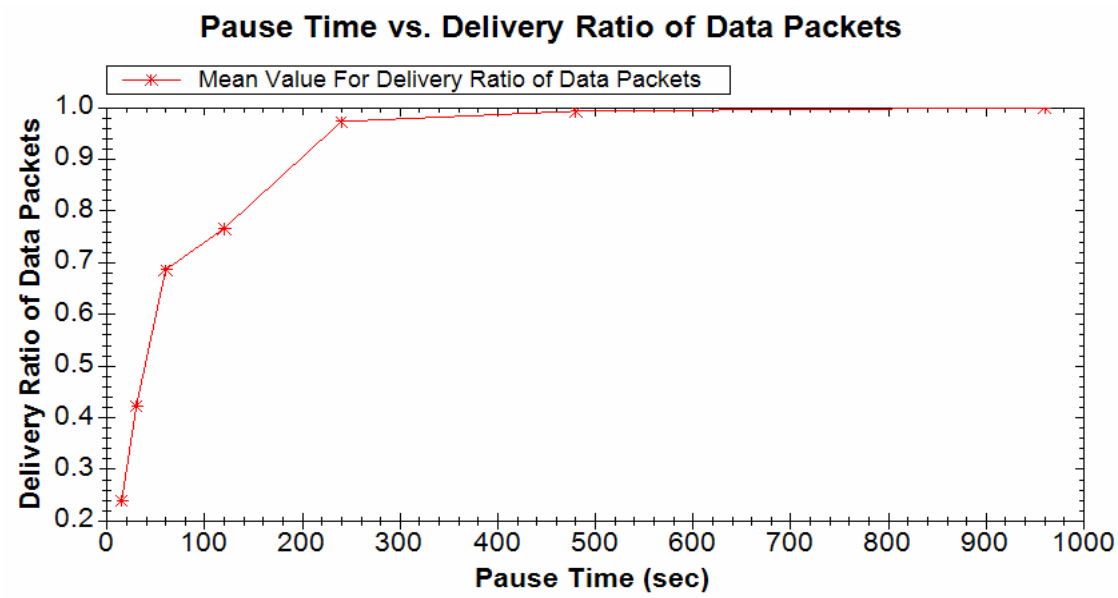

Figure 4.10: Pause Time vs. Delivery Ratio of Data Packets.

4.3.2.4 Pause Time vs. Normalized Control Overhead

Figure 4.11 shows that as the pause time increases the normalized control overhead decreases. This is because more control packets are used for performing Route Discovery and Maintenance processes when participating nodes are highly mobile.

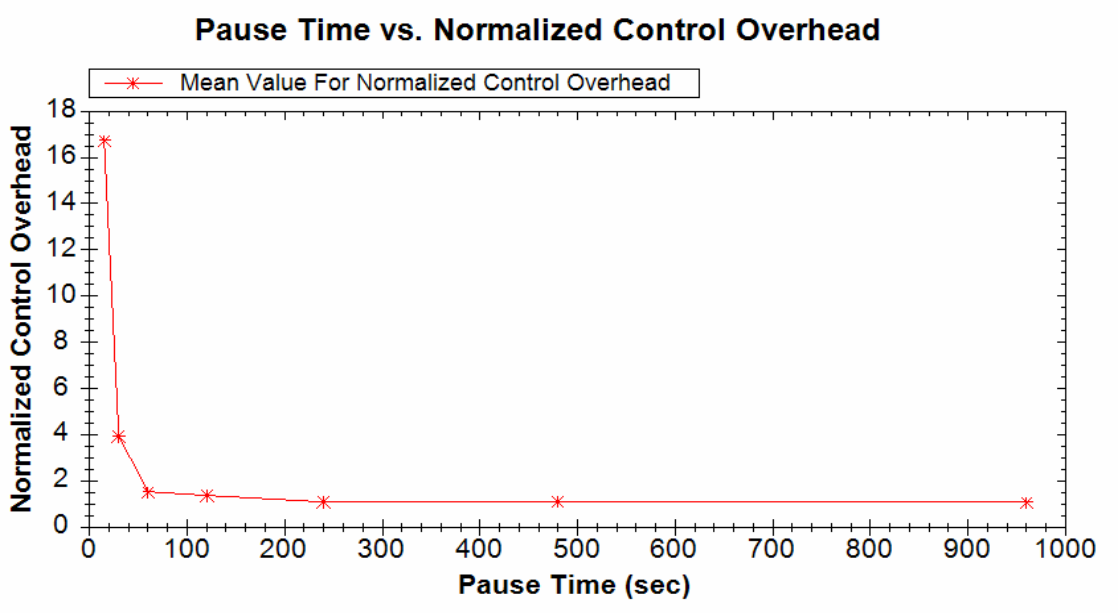

Figure 4.11: Pause Time vs. Normalized Control Overhead.

4.3.2.5 Pause Time vs. Throughput

As expected, Figure 4.12 shows that throughput also increases as the pause time grows. That is, as the network becomes more stable, it takes less time for a source node to send the same amount of data packets.

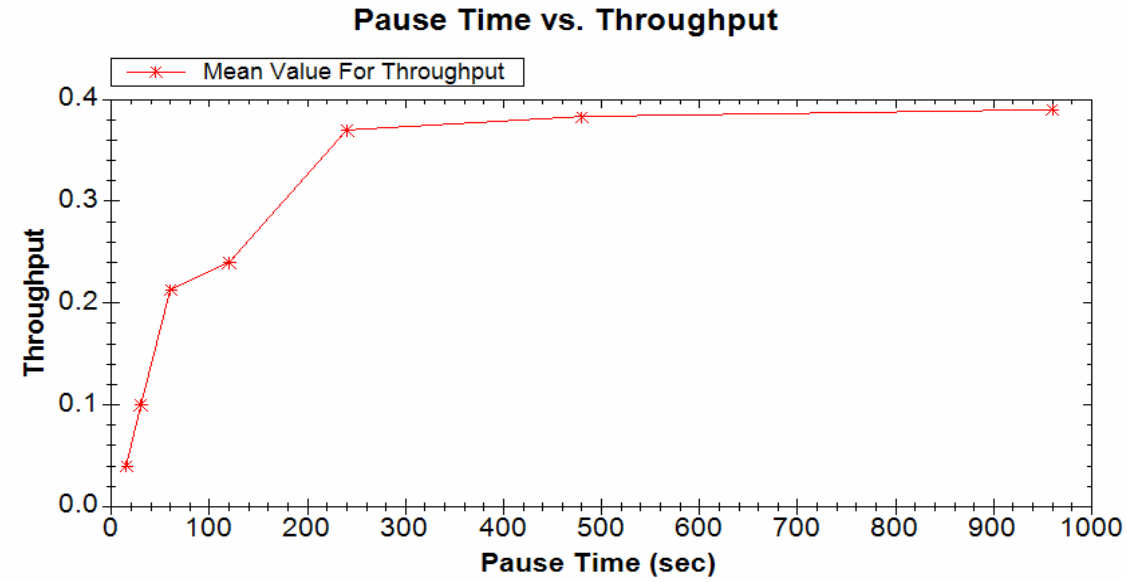

Figure 4.12: Pause Time vs. Throughput.

4.3.3 Summary of Result

In conclusion, our wireless ad hoc messenger using the DSR protocol performs better in terms of the five metrics selected when the pause time is long at which the random waypoint model generates a more stable network. More specifically, when the pause time is sufficiently long (>240 seconds in our experiment), the latency to deliver a data packet is low, the delivery ratio of data packets is high, the number of control packets used per a data packet is small, and the throughput is high. An interesting result is that when the pause time is relatively long, an initial Route Discovery takes more time compared to the latency for Route Discovery over the entire simulation period, because a long pause time creates a stable network so a subsequent Route Discovery can utilize valid cache entries to find valid routes to reduce the search time. On the other hand, when the pause time is very small resulting in a very dynamic network, an initial Route Discovery takes less time compared to the latency for Route Discovery because most cache entries are not valid in this case, thus causing a subsequent Route Discovery process to take more time to try invalid routes stored in the cache. We will examine the effect of using cache more in Section 4.6.

4.4 Performance of Messenger with respect to Sliding Window Size (SWS)

4.4.1 Scenario

All parameters in this experiment assume their default values except the Sliding Window Size (SWS) parameter, which changes its value from 1 to 6 as follows:

- SWS: 1, 2, 3, 4, 5, 6

4.4.2 Results

4.4.2.1 SWS vs. Average Latency to Find a New Route

Figure 4.13 shows how much time is spent to find a new route on average, as SWS grows. When SWS is 1 or 2, our wireless ad hoc messenger can endure the amount of network traffic load and the latency to find a new route is reasonably small.

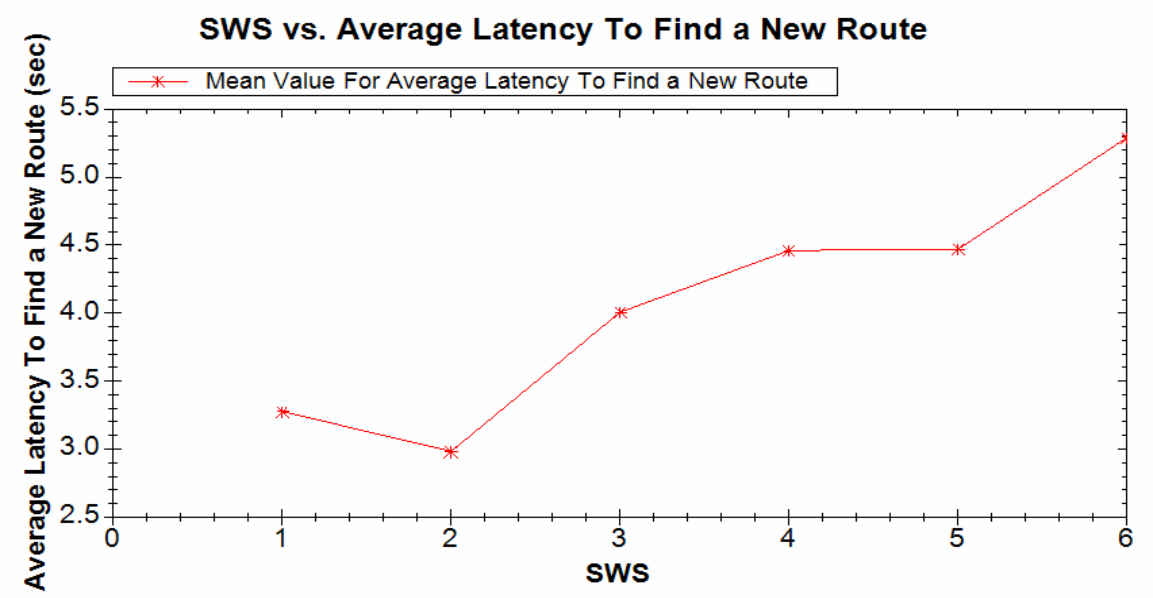

Figure 4.13: SWS vs. Average Latency to Find a New Route.

However, as SWS is larger than 2, the latency to find a new route increases. A possible reason is that the underlying MAC layer protocol (IEEE 802.11) for wireless transmission is based on RTS/CTS with CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance), so a lot of collision may occur when multiple packets are transmitted simultaneously. That is, when multiple packets are sent at the same time, the Collision Avoidance mechanism utilized in IEEE 802.11 may cause a high packet transmission delay when the medium is busy. Furthermore, a packet may be thrown away after a given number of retransmissions, or it may be garbled and not received by the receiver. These reasons contribute to a long delay in finding a new route. In particular, when a packet arrives late at the destination node such that the ACK of the packet is not received by the sender within the end-to-end timeout period, a new Route Discovery would be triggered by the source node, causing a further delay.

4.4.2.2 SWS vs. Average Latency to Deliver a Data Packet

Figure 4.14 shows that as SWS increases, the average latency to deliver a data packet increases. The reason is that as SWS increases, more packets are in transit and the Collision Avoidance mechanism of IEEE 802.11 tends to cause a packet to be delivered late or thrown away after a given number of retransmissions or garbled and not received by the receiver.

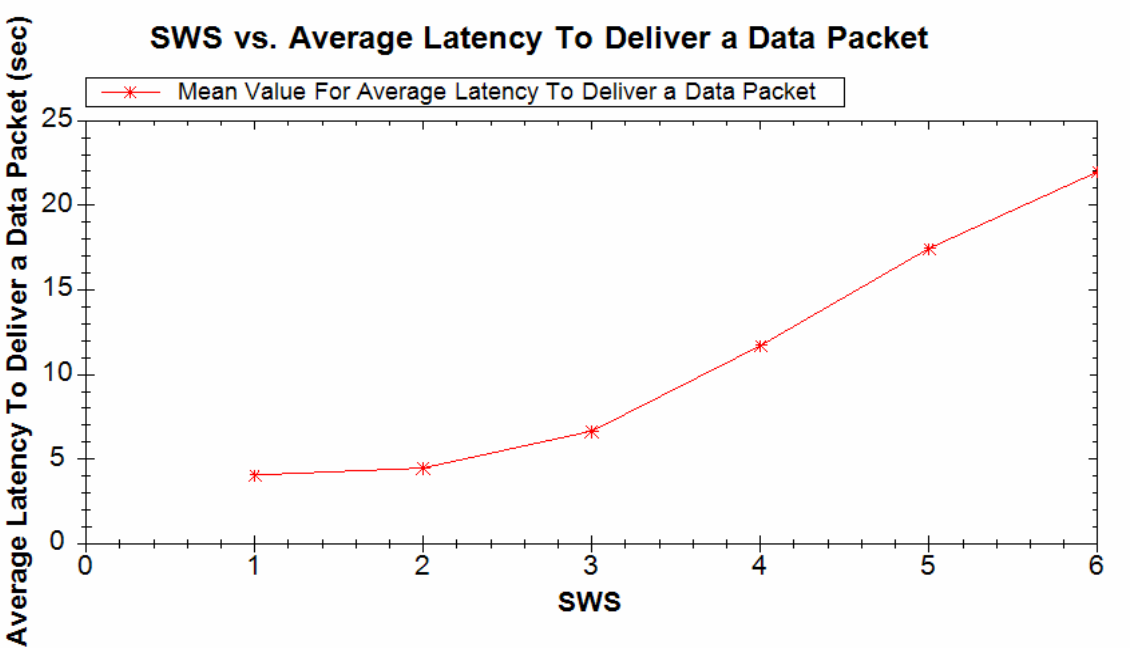

Figure 4.14: SWS vs. Average Latency to Deliver a Data Packet.

4.4.2.3 SWS vs. Delivery Ratio of Data Packets

Figure 4.15 demonstrates the effect of increasing the SWS on the delivery ratio of data packets in our wireless ad hoc messenger. It shows that the delivery ratio of data packets decreases, as SWS increases. The reason is that more packets need to be retransmitted as SWS increases because packets could arrive late or are lost due to a higher level of medium contention. Thus, more data packets are sent than those actually delivered to the receiver.

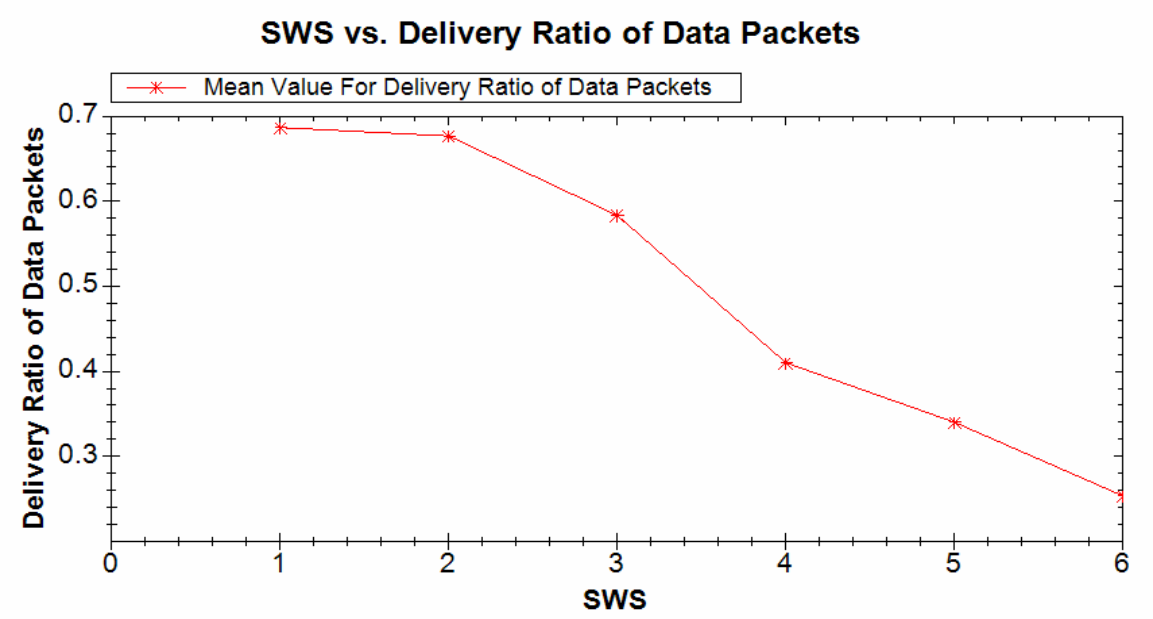

Figure 4.15: SWS vs. Delivery Ratio of Data Packets.

4.4.2.4 SWS vs. Normalized Control Overhead

Figure 4.16 shows the number of control packets incurred per data packet, as a function of SWS. As expected, more control packets are required as SWS increases. Since the RTS/CTS mechanism with CSMA/CA tends to cause more Route Discovery processes to be invoked when the network traffic increases, more control packets (i.e. Route Discovery Packets and Reply Route Discovery Packets) are required as SWS increases.

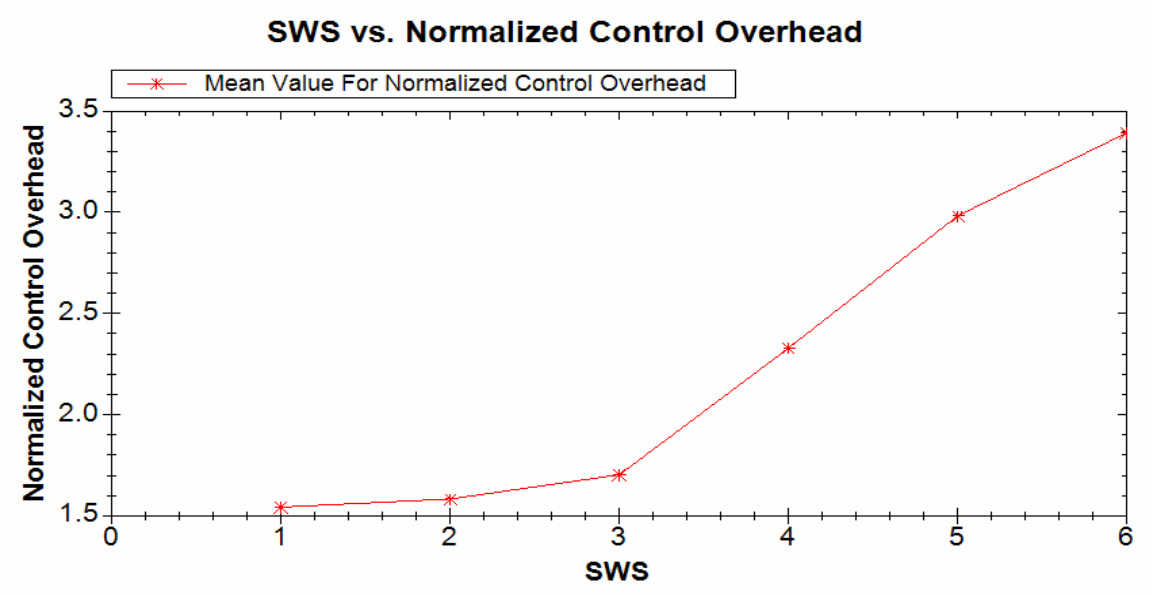

Figure 4.16: SWS vs. Normalized Control Overhead.

4.4.2.5 SWS vs. Throughput

Figure 4.17 shows the number of packets delivered per second as a function of SWS. When SWS is 2, the throughput is the highest, beyond which the throughput decreases as SWS grows. Due to parallelism, the throughout when SWS is 5 is better than the throughput when SWS is 1. However, when SWS is 6, the high latency experienced due to heavy medium contention causes the throughput to fall below the throughput at SWS equal to 1. Thus although data parallelism introduced through the use of SWS > 1 could help improve the throughput, the medium contention effect in IEEE 802.11 could overweigh the benefit and degrade the throughput. As a result, there exists an optimal SWS at which the throughout is maximized.

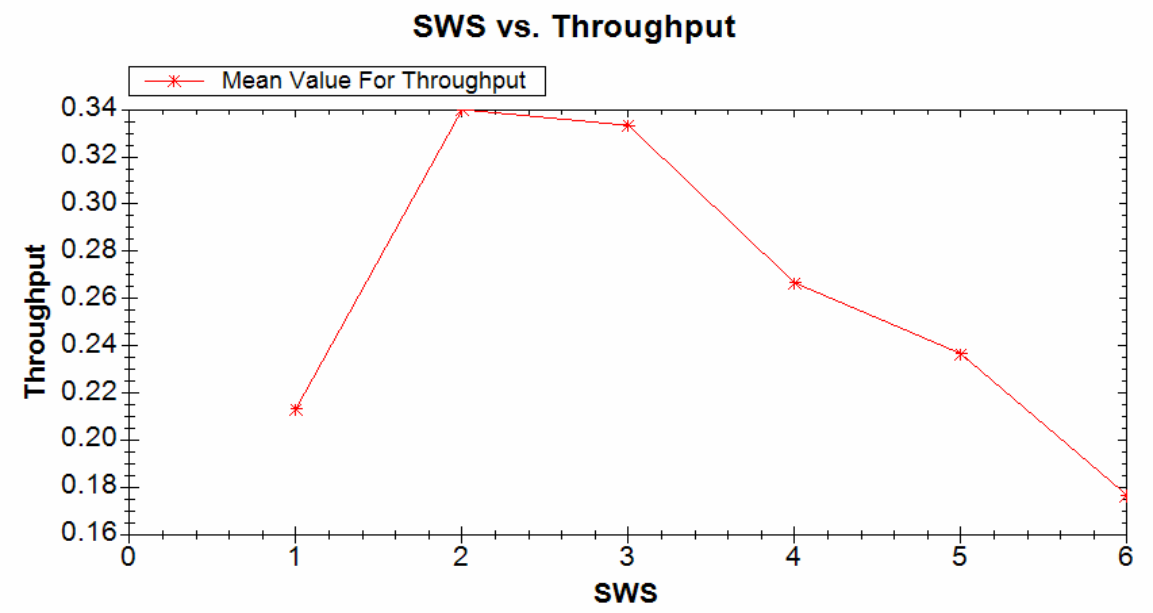

Figure 4.17: SWS vs. Throughput.

4.4.3 Summary of Result

This experiment tested the effect of SWS on the performance of our ad hoc messenger using DSR in terms of the five metrics selected. Since a large SWS encourages parallelism in data processing, our wireless ad hoc messenger application exhibits better performance in throughput when SWS is 2 than when SWS is 1. However, when SWS is larger than 2, the throughput decreases because of the high latency experienced in packet delivery due to the collision avoidance mechanism used in IEEE 802.11 to handle multiple packets being transmitted simultaneously. We also observe that at SWS equal to 2, the latency to find a new route is at its minimum with a similar reasoning applied. The optimal SWS size depends on the system bandwidth capacity; in our experiment we observe the optimal SWS being 2 because the latency to deliver a data packet, the delivery ratio of data packets, and the normalized control overhead all deteriorate badly beyond this optimal SWS value.

4.5 Performance of Messenger with/without Receipt Packets

4.5.1 Scenario

In this experiment, all parameters assume their default values except the pause time and the use of receipt packets. The pause time parameter changes from 15 seconds to 960 seconds and the use of receipt packets is turned On/Off as follows:

- Pause Time: 15, 30, 60, 120, 240, 480, 960 (seconds)
- Use of Receipt Packets: On or Off

4.5.2 Results

4.5.2.1 Pause Time vs. Average Latency to Find a New Route with/without Receipt Packets

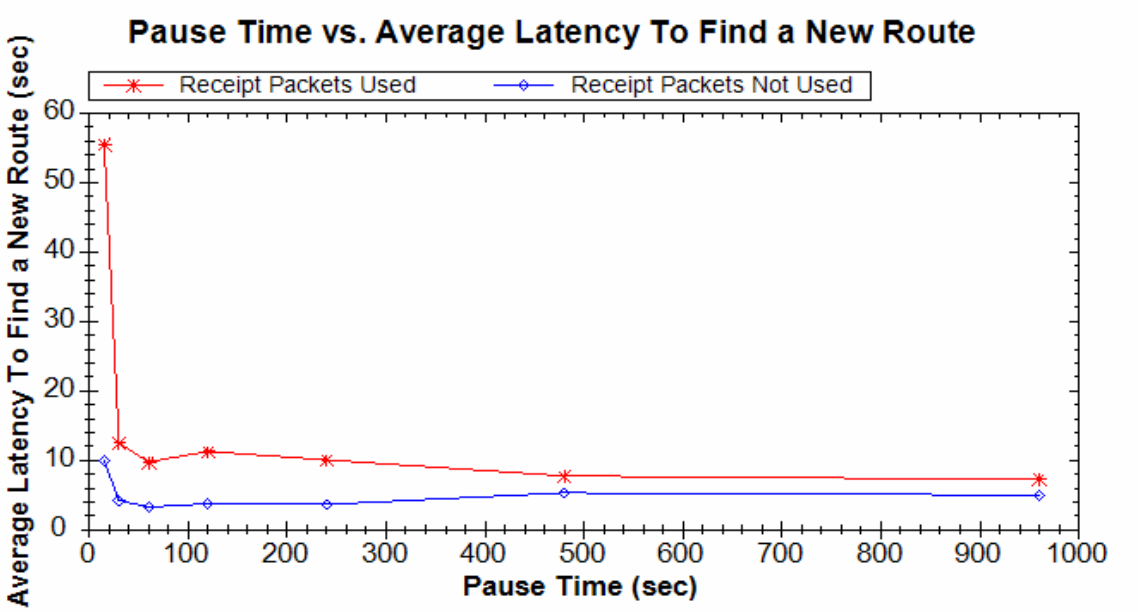

Figure 4.18: Pause Time vs. Average Latency to Find a New Route with/without Receipt Packets.

Figure 4.18 shows two curves for the average latency to find a new route when receipt packets are used or not used, respectively. It shows that not using receipt packets gives a shorter latency to find a new route than using receipt packets. This indicates that using receipt packets may adversely bring higher network traffic and cause each packet to arrive late. Thus, the delayed arrival of each packet leads to a longer latency in finding a new route when Route Discovery is performed.

4.5.2.2 Pause Time vs. Average Latency to Deliver a Data Packet with/without Receipt Packets

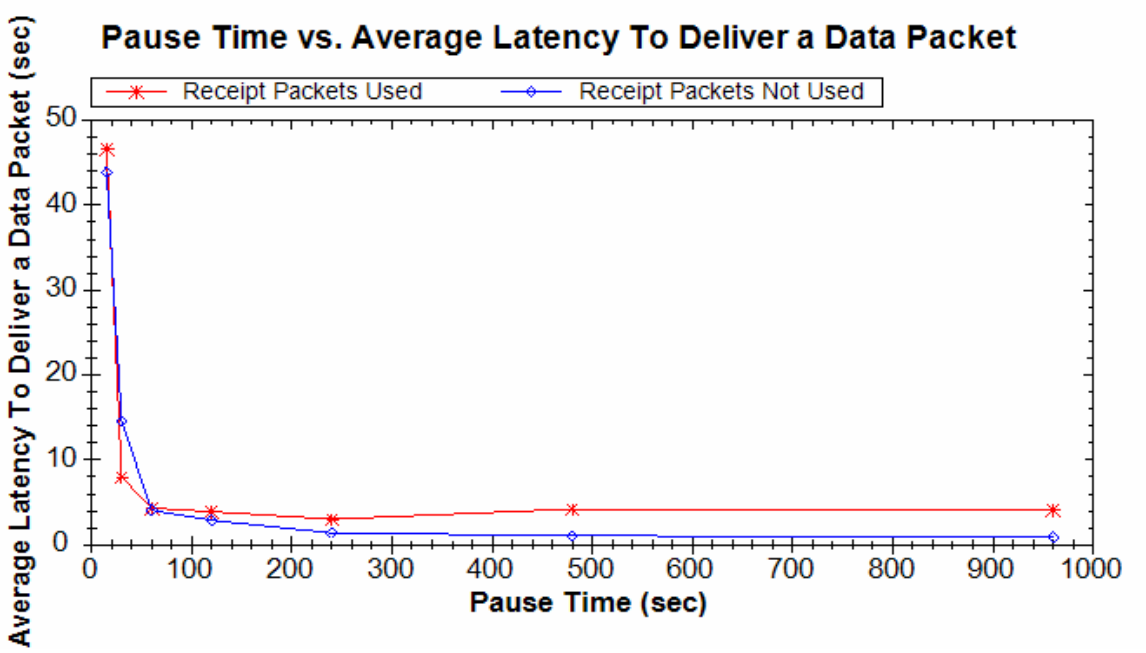

Figure 4.19: Pause Time vs. Average Latency to Deliver a Data Packet with/without Receipt Packets.

Figure 4.19 shows that not using receipt packets gives better performance in terms of the average latency to deliver a data packet in general. However, when the pause time is shorter or the network is very unstable, using receipt packets may give better performance sometimes, i.e., for the case when the pause time is 30 seconds. Also, the performance of two experiments diverges especially when the pause time is 60 seconds. Since the system using receipt packets detects route failure by intermediate nodes more quickly than that of not using receipt packets, the benefit of the rapid detection of route failure may exceed the overhead of using more control packets. When receipt packets are not used, a source node tries to resend the same data packet three times each with an end-to-end timeout interval. After then, if the source node still has not received the ACK of the data packet sent, it performs a Route Discovery process because the source node regards the data packet as lost. Thus, when receipt packets are not used, the amount of network traffic is much less than using receipt packets. However, in the experiment without receipt packets, if there is a route failure, the source node needs to timeout three times before the failure is detected. Therefore, if the network bandwidth can sustain the amount of control

packets generated by using receipt packets, and the network topology is unstable, then using receipt packets can lower the average latency to deliver a data packet.

4.5.2.3 Pause Time vs. Delivery Ratio of Data Packets with/without Receipt Packets

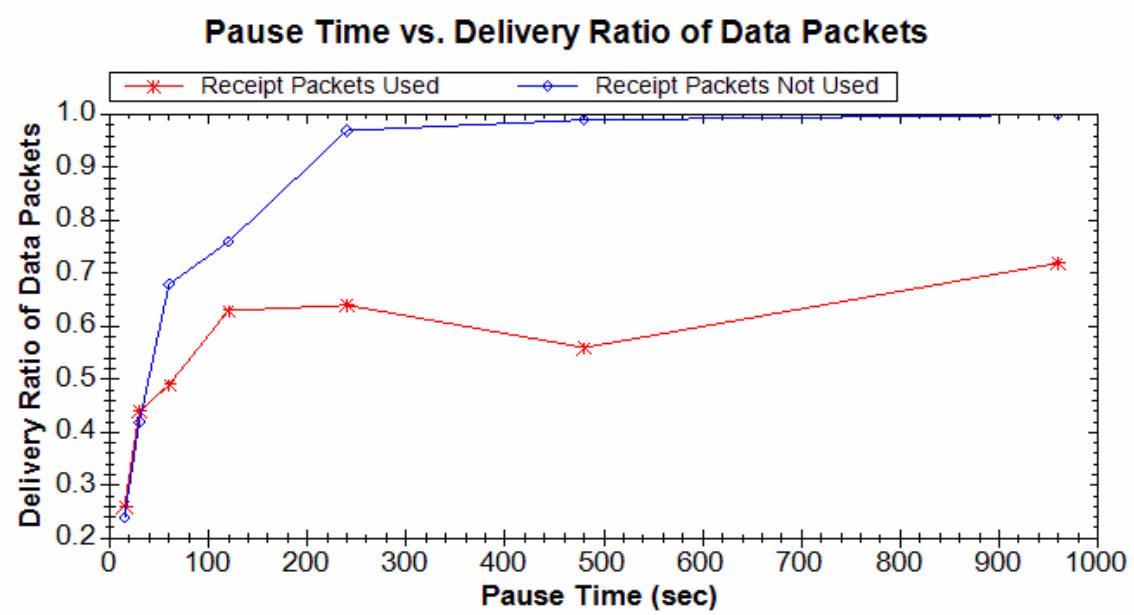

Figure 4.20: Pause Time vs. Delivery Ratio of Data Packets with/without Receipt Packets.

Figure 4.20 shows how the pause time affects the delivery ratio of data packets when receipt packets are used or not, respectively. The two curves diverge when the pause time is at 30 seconds. When the pause time is shorter or the network topology frequently changes, using receipt packets improves the delivery ratio of data packets. However, as the network topology becomes stable, using receipt packets is not useful and can even be harmful as it introduces more control packets. While the use of receipt packets is to detect route errors by intermediate nodes for rapid detection of route failures, it is not useful in a stable network. Consequently, receipt packets should be used only in a network with relatively fast moving nodes so that the benefit of rapidly detecting route errors by intermediate nodes outweighs the disadvantage of more control packets being introduced into the network.

4.5.2.4 Pause Time vs. Normalized Control Overhead with/without Receipt Packets

Figure 4.21 shows two curves for the normalized control overhead when receipt packets are being used or not, respectively. This confirms that our wireless ad hoc messenger using receipt packets generates much more control packets.

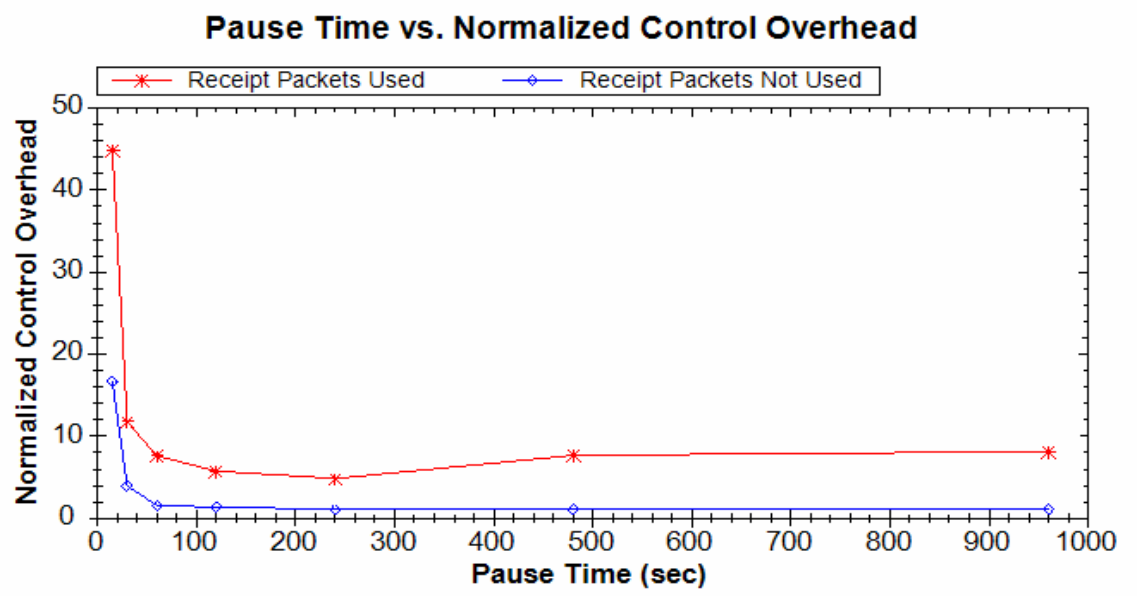

Figure 4.21: Pause Time vs. Normalized Control Overhead with/without Receipt Packets.

4.5.2.5 Pause Time vs. Throughput with/without Receipt Packets

Figure 4.22 shows two curves for the throughput when receipt packets are being used or not, respectively. The result correlates well with the packet delivery ratio result. That is, when the network topology is dynamic, or participating nodes move frequently, using receipt packets detects broken routes quickly by intermediate nodes and, as a result, improves the throughput.

However, in a stable network in which a route error does not occur often, using receipt packets increases the amount of network traffic and causes packets to be delivered out of the timeout period, and, consequently, degrades the throughput.

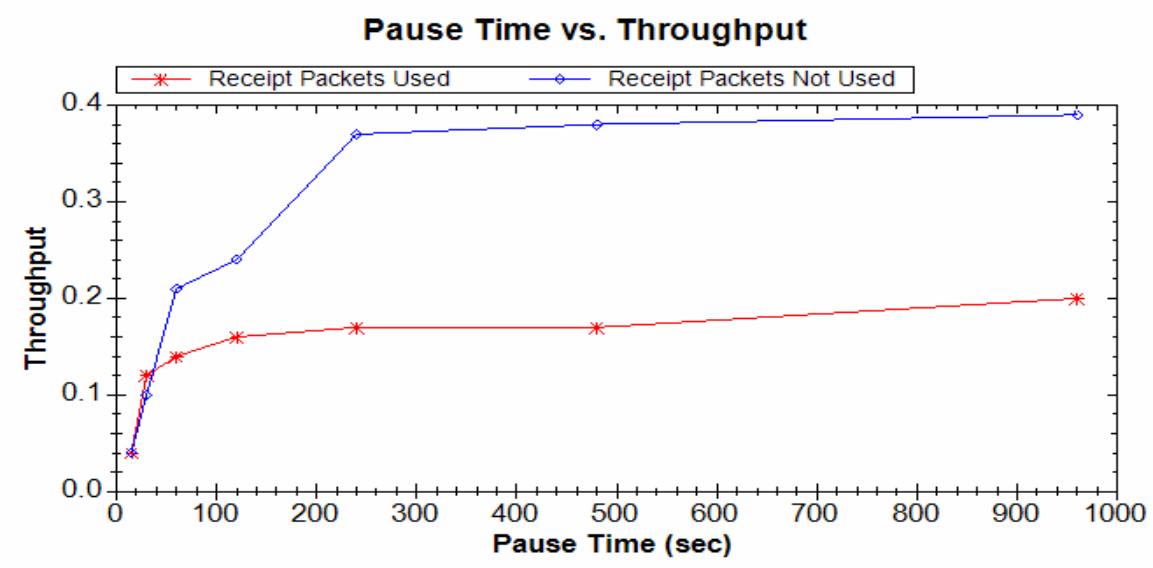

Figure 4.22: Pause Time vs. Throughput with/without Receipt Packets.

4.5.3 Summary of Result

We tested the effect of using receipt packets for detecting route errors more rapidly on the performance of our ad hoc messenger application under stable and dynamic network topologies. The pause time was varied to generate stable or dynamic network topologies based on the random waypoint model. Since using receipt packets generates more network traffic, it may cause more data delivery delay due to the collision avoidance mechanisms of IEEE 802.11. Thus, when the network is stable, using receipt packets is not beneficial and could be harmful because of a higher traffic load being introduced. However, since the use of receipt packets facilitates route failures to be detected more rapidly, the use of receipt packets may be advantageous if the network topology is very dynamically changing.

4.6 Performance of Messenger with/without Cache Structure

By default, our cache structure includes all available routes obtained from Route Discovery. When the current route is broken, it is removed from the cache of the source node. On the other hand, new routes acquired from a new Route Discovery are added to the cache of the source node. When the source node uses a route in its cache, it always selects the shortest route to reach the destination. When the selected route is not valid, the source node will try alternative routes in its cache if they are available.

When cache is not utilized, only one route is available in the source node. Thus, if the currently using route is not valid any more, the source node must invoke a Route Discovery process again to find a new route.

4.6.1 Scenario

In this experiment, all parameters assume their default values except the pause time and the use of cache. The pause time parameter changes from 15 seconds to 960 seconds and the use of cache is turned On/Off as follows:

- Pause Time: 15, 30, 60, 120, 240, 480, 960 (seconds)
- Use of Cache Structure: On or Off

4.6.2 Results

We first report the average number of source-destination routes maintained in the cache of the source node. Figure 4.23 shows that average number of routes maintained is between 1 and 3 for the case in which cache is used, and is between 0 and 1 for the use in which the cache structure is not utilized.

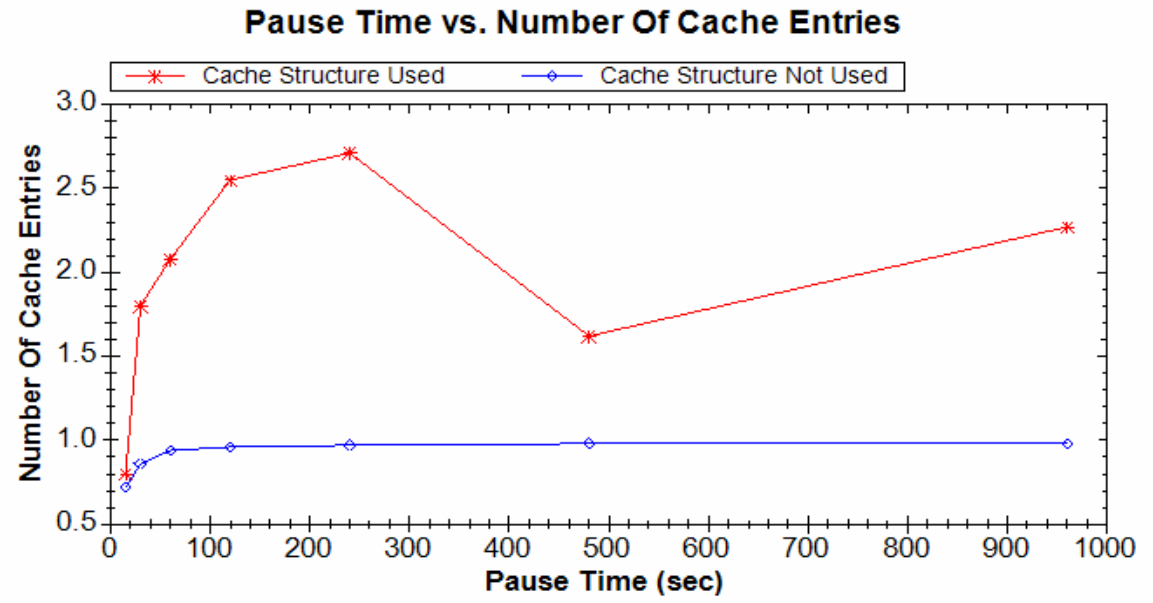

Figure 4.23: Pause Time vs. Average Number of Cache Entries with/without Cache Structure.

4.6.2.1 Pause Time vs. Average Latency to Find a New Route with/without Cache Structure

Figure 4.24 illustrates how the cache structure affects the average latency to find a new route. There is a break-even point at pause time = 60. When the pause time is small (<60) and thus the network is dynamic, the use of cache provides a higher latency. On the other hand, when the pause time is large (>60) and thus the network is stable, using cache provides a lower latency.

The result means that when the network topology is frequently changing, the use of cache does not provide too much benefit and can even degrade performance. The reason is that when the network is very dynamic, cache entries tend to become invalid so it increases the time to find a route since the source will try invalid cache entries before invoking a Route Discovery process to search for a route.

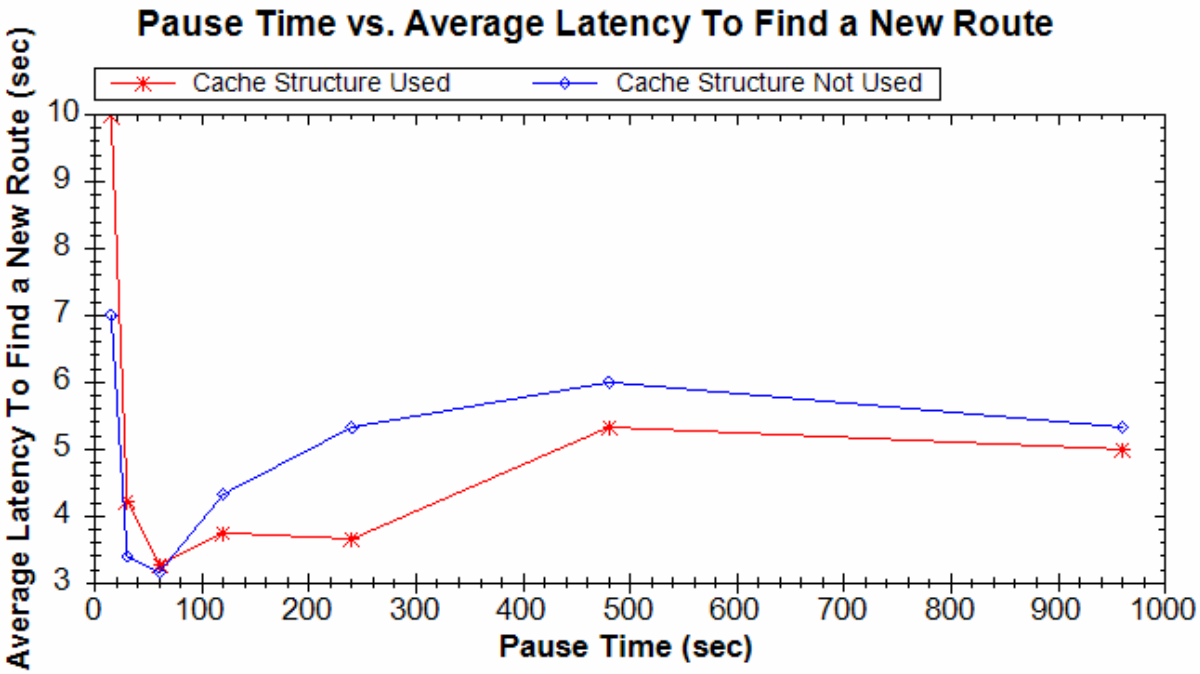

Figure 4.24: Pause Time vs. Average Latency to Find a New Route with/without Cache Structure.

As shown in Figure 4.24, our application performs better with the use of cache when the network is stable. When the network topology does not change often, the alternative routes stored in the cache are helpful to find a new route because the source node does not have to perform an expensive Route Discovery process, and rather simply uses a valid route in its cache.

There are optimization techniques, such as refreshing the route cache periodically, or turning on or off the option of using cache based on the network topology status as reported in [Lou2002, Wu2002, Hu2000, Hu2002]. These optimization techniques were not explored in this experiment and will be left as future work.

4.6.2.2 Pause Time vs. Average Latency to Deliver a Data Packet with/without Cache Structure

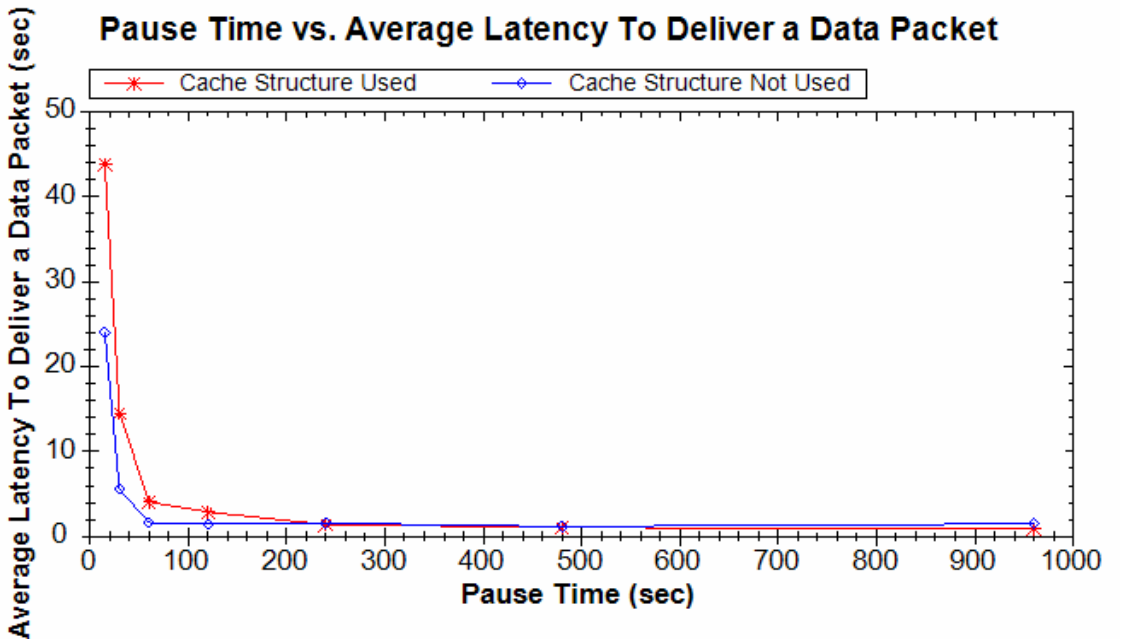

Figure 4.25: Pause Time vs. Average Latency to Deliver a Data Packet with/without Cache Structure.

Figure 4.25 shows how the use of cache structure affects the average latency to deliver a data packet. The trend is the same as before. However, different from the result on the average latency to find a new route, even under a stable network topology, there is no significant difference in the average latency to deliver a data packet with or without the use of cache. This is because the latency to deliver a data packet represents the delay for one data packet to travel from a source node to a destination node, and there would not be much difference if one route used is valid for a long period of time.

4.6.2.3 Pause Time vs. Delivery Ratio of Data Packets with/without Cache Structure

Figure 4.26 shows the effect of cache on the delivery ratio of data packets. Again, when the pause time is small (<240 seconds) and the network is dynamic, the use of cache causes more data packets to be retransmitted because the source uses invalid routes stored in the cache to transmit data packets.

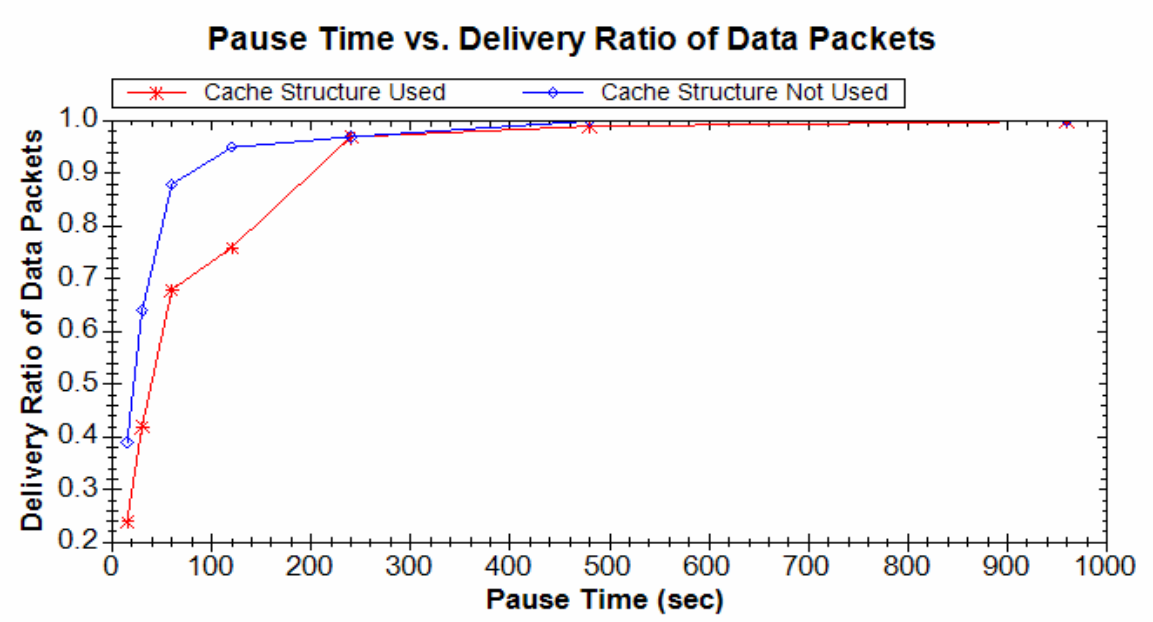

Figure 4.26: Pause Time vs. Delivery Ratio of Data Packets with/without Cache Structure.

4.6.2.4 Pause Time vs. Normalized Control Overhead with/without Cache Structure

Figure 4.27 compares the normalized control overhead of our system when the cache structure is used vs. not used. More control packets are used under high mobility of participating nodes in the network especially when cache structure is used. As explained before, when nodes in the network are moving frequently, the use of cache produces more route error packets. That means that more control packets would be transmitted because of the use of stale route information in the cache.

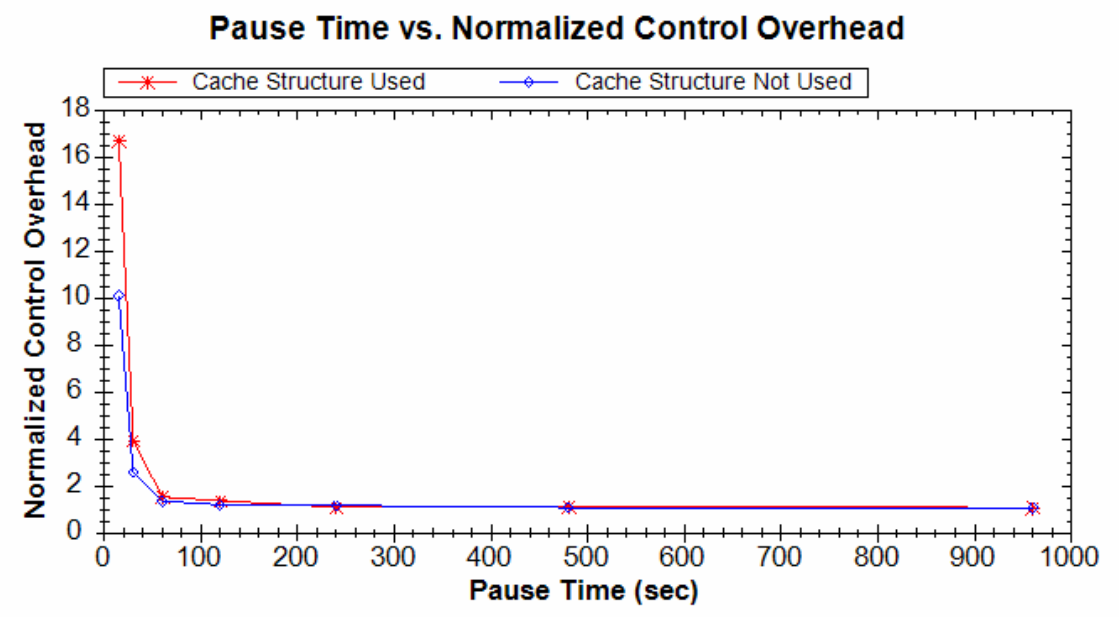

Figure 4.27: Pause Time vs. Normalized Control Overhead with/without Cache Structure.

4.6.2.5 Pause Time vs. Throughput with/without Cache Structure

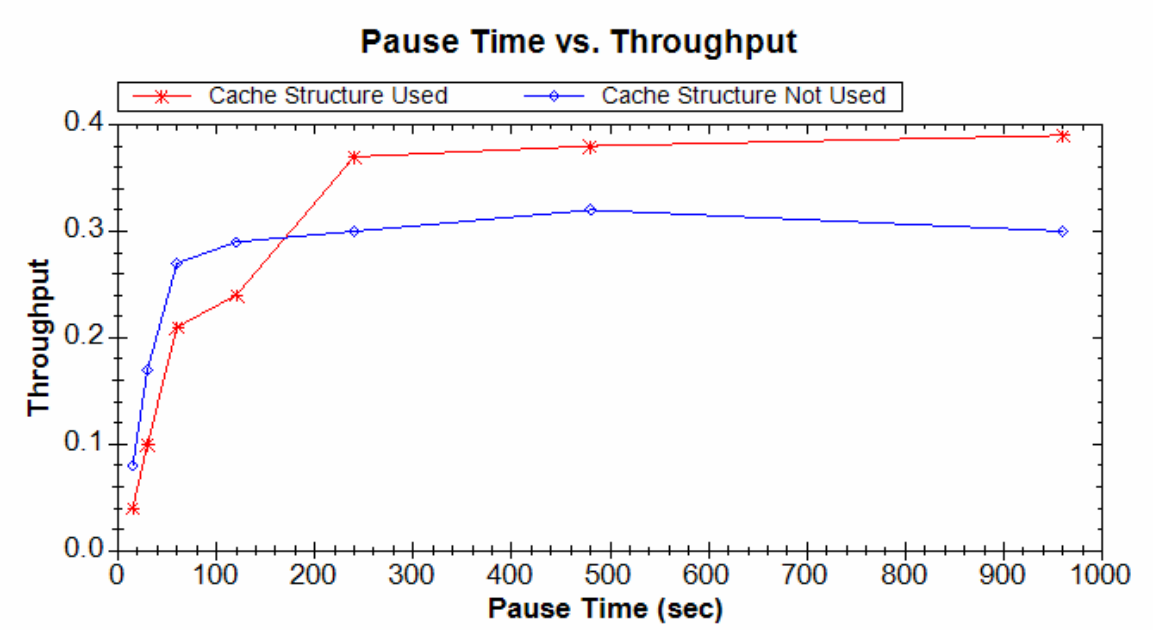

Figure 4.28: Pause Time vs. Throughput with/without Cache Structure.

Figure 4.28 shows the impact of using cache structure on the throughout. A distinct break-even point exists. That is, the use of cache improves the throughput only when the pause

time is relatively long, confirming that using cache is helpful in improving the performance of our application when the network is relatively stable.

4.6.3 Summary of Result

In this experiment, we tested the effect of using cache on the performance of our wireless ad hoc messenger application. The results showed that the use of cache structure as a design alternative in DSR is not helpful when the network topology is unstable and likely to change frequently. That is, when the pause time is relatively short, and thus the network topology is dynamic, using cache may even be harmful to the system performance because of the extra delay incurred through using invalid cache entries to find a route. There exists a break-even point (with respect to the degree of changes in the network topology) at which the use of cache can improve the performance of the system although the improvement obtained is in general small.

4.7 Summary

In this Chapter, we have described the experiment setup and reported results for evaluating the performance of our wireless ad hoc messenger application with physical interpretations given. Below we summarize the experimental results obtained.

First, as the node speed increases, the performance of our wireless ad hoc messenger decreases because as the speed increases, a more dynamic network will result, thus causing routes to be broken more easily.

Second, as the pause time increases, the performance of our wireless ad hoc messenger also increases. The reason is that a long pause time produces a relatively stable network topology.

Third, the increase of SWS degrades the performance of our ad hoc messenger application on the one hand due to a higher level of medium contention between multiple packets being transmitted simultaneously, but improves the performance on the other hand due to a higher degree of parallelism introduced. We discovered that there exists an optimal SWS value under which these two factors are best balanced and the system performance is maximized. In our experiment, we observed that the optimal SWS value is 2, beyond which the system performance degrades as SWS increases.

Fourth, the use of receipt packets does not provide any benefit when the network topology is stable because of extra control packets being introduced into the system. However, when the network topology is changing rapidly, the use of receipt packets may improve the system performance because route errors can be quickly detected by intermediate nodes.

Finally, the use of cache may improve performance if the network topology is relatively stable. However, when the network topology is rapidly changing, the use of cache could be harmful to system performance because most cache entries may be invalid, causing the system to spend more time to search for a route during a Route Discovery or Maintenance process.

Chapter 5
Conclusions and Future Work

5.1 Summary of Thesis

In this thesis, we designed, implemented and evaluated a multi-hop ad hoc messenger using Pocket PCs and Microsoft .Net Compact Framework. Each Pocket PC communicates wirelessly with another using the IEEE 802.11b technology without any aid of infrastructure. The main protocol implemented in this application was the DSR (Dynamic Source Routing) protocol, which consists of two important mechanisms, *Route Discovery* and *Route Maintenance*. Since the DSR protocol operates solely based on source routing and *on-demand* process, it has been selected as the routing protocol to be implemented and tested for our ad hoc messenger application characterized by a source on-demand chat conversation between nodes in a mobile ad hoc network. The mobility behavior of nodes in the application is modeled by the random waypoint model through which random locations to which a node move are generated, and the associated speed and pause time are specified to control the frequency at which the network topology is changed.

To test the performance of our wireless ad hoc messenger using DSR, five standard metrics are evaluated, namely, average latency to find a new route, average latency to deliver a data packet, delivery ratio of data packets, normalized control overhead, and throughput. These metrics tests if the application using DSR functions correctly and efficiently in an ad hoc network.

To provide the testing conditions, five parameters are manipulated reflecting changes in the network topology and design alternatives of the wireless ad hoc messenger application. In order to test the performance of the implemented ad hoc messenger under changing network topologies, the node speed and pause time parameters are manipulated. Three design alternatives, namely, Sliding Window Size (SWS), the use of receipt packets, and the use of cache, are examined. Specifically, SWS in our application is increased to know its effect of parallelism of data processing on the performance of our application. The effect of the use of receipt packets deigned for rapid detection of route errors by intermediate nodes is evaluated. Finally, the effect of using cache structure is tested.

The performance of our wireless ad hoc messenger using DSR has been evaluated against the five metrics selected based on scenarios generated through manipulating the speed and pause

time of the mobility model and design variables (SWS, receipt packets, and cache structure). The results are summarized as follows:

- *The effect of speed on the performance of our wireless ad hoc messenger using DSR*: As the node speed increases such that the network topology becomes more dynamic, the performance deteriorates.
- *The effect of pause time on the performance of our wireless ad hoc messenger using DSR*: As the pause time increases so the network topology becomes more stable, the system performance improves.
- *The effect of increasing SWS on the performance of our wireless ad hoc messenger using DSR*: There exists an optimal SWS value under which the average latency to find a new route would be minimized and the throughput would be maximized. When SWS goes beyond the optimal size, which is 2 in our experiment, the system performance degrades because the delay experienced per data packet in the presence of multiple data packets being transmitted simultaneously in IEEE 802.11 outweighs the benefit of data parallelism as SWS increases.
- *The effect of the use of receipt packets on the performance of our wireless ad hoc messenger using DSR*: Receipt packets are designed to more rapidly detect broken routes by intermediate nodes. We discovered that there is a "break-even" point in term of the frequency of topology changes beyond which the use of receipt packets can improve the performance of the system because the use of receipt packets allows route errors to be detected quickly and reduces the delay caused by redundant retransmissions of data packets through an out-of-data route.
- *The effect of the use of cache structure on the performance of our wireless ad hoc messenger using DSR*: We discovered that the use of cache is not beneficial or even harmful when the network topology is relatively dynamic. This is because route entries store in the cache in a highly dynamic network can become invalid quickly so using them to search for a route during the Route Discovery process can generate more route errors and cause more delays. On the other hand, when the network topology is relatively stable, cache entries are likely to be valid so the use of cache structure can improve the system performance in terms of the average latency to find a new route and the system throughput.

Based on these experimental results, it is clear that the three design alternatives studied in the thesis suggest that the ad hoc messenger application should be implemented in an adaptive

manner such that the SWS value, whether to use receipt packets, and whether to use cache can be determined dynamically based on the current network condition detected through feedback from the system. A possible method is to dynamically measure the performance of system in terms of the five performance metrics introduced in the thesis and fine-tune these three parameters to maximize the system performance.

5.2 Accomplishments

With respect to the purposes of the thesis, the followings goals have been accomplished:

- A real chatting prototype using Pocket PCs has been designed and implemented using IEEE 802.11b wireless cards, Microsoft .Net Compact Framework (C#), and XML technology.
- A real chatting prototype has been implemented based on the DSR protocol, fully implementing its basic two mechanisms, Route Discovery and Route Maintenance.
- XML was utilized to describe data packets and result files to facility data exchanges and analysis.
- A GEN application has been developed to generate a network configuration file modeling changing network topologies based on the random waypoint mobility model.
- An Automatic Data Collection application has been developed to ease the testing and evaluation of our ad hoc chatting application. In particular, the Automatic Data Collection application implemented automatic data transmission, generation of result files using XML technology, and collection of result files using wireless transmission (using the Send and Receive Files application) into a centralized Statistical Analyzer application for generating appropriate result graphs according to a set of selected x-parameter and y-parameter.
- The performance of our wireless ad hoc messenger under various scenarios has been tested and analyzed appropriately against five standard metrics and the results indicate conditions under which design alternatives (SWS, the use of receipt packets, and the use of cache) should be employed in order to improve the performance of the application. Physical interpretations of the results along with suggestions for dynamically adjusting the values of these design parameters so as to optimize the system performance are given.

5.3 Future Work

This work designed, implemented and evaluated a wireless multi-hop ad hoc instant messenger with only 7 Pocket PCs fully implementing the basic features of the DSR protocol along with a few design alternatives. It paves the way for more future work incorporating possibly more nodes and more optimized features of the DSR protocol. For example, an optimized cache structure [Lou2002, Wu2002, Hu2000], as mentioned in the related work of Chapter 2, can be considered and its effect may be tested with more nodes in the system. In addition, other optimized features [Johnson2001, Seet2003] can be implemented and their performances can be evaluated, such as reducing route discovery reply packets for route discoveries performed, packet salvaging, replying route request using cached routes, and so on, as mentioned in the additional Route Discovery and Maintenance features in Chapter 2.

The use of receipt packets at the network layer may not be necessary if the underlying link layer subsystem can return an ACK to indicate if a packet can be delivered successfully over a link between two hops. Exploration of a link-layer API is called for in future work.

The design of the wireless ad hoc messenger can also be improved by separating application-layer from network-layer implementations. The current implementation uses the same port to send/receive application-layer data packets and network layer route discovery and maintenance packets. Future work should separate network-layer protocol implementation from application-layer ad hoc messenger application implementation to follow the design concept of layering and separation of concern. This would allow other applications to be easily built on network-layer implementation.

A more sophisticated design for the GEN program for creating dynamically changing network topologies to test the ad hoc messenger application is also called for. When the ACM option is turned on, the topology generated can reflect various degrees of connectivity of nodes by design, e.g., generating a mesh vs. a tree structure, and to test the effect on system performance of the application.

Furthermore, future work can test performance of other routing protocols, such as AODV, DSDV, OLSR, geographical forwarding [Li2001], etc. to compare against the DSR protocol. In order to optimize the use of constrained resources in an ad hoc network, mobility prediction [Su2001] and battery power conservation techniques [Xu2001] can be developed and experimented to test the effect of these ad hoc routing protocols on a real application, such as the real ad hoc messenger developed in the thesis. Moreover, our small-scale network consisting of only 7 Pocket PCs can be extended to a large-scale deployment with more mobile devices included using Bluetooth or IEEE 802.11 technology in the future [Chen2002].

Bibliography

[Bettstetter2003] Christian Bettstetter, "Topology Properties of Ad Hoc Networks with Random Waypoint Mobility," *ACM SIGMOBILE Mobile Computing and Communications Review,* Vol. 7, Issue 3, pages 50-52, July 2003.

[Brenner1997] Pablo Brenner, "A Technical Tutorial on the IEEE 802.11 Standard" *BreezeCom,* 1997.

[Broch1998] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile and Networking* (MobiCom'98), ACM 1998 October, pages 85-97, 1998.

[Camp2002] Tracy Camp, Jeff Boleng, Brad Williams, Lucas Wilcox and William Navidi, "Performance Comparison of Two Location Based Routing Protocols for Ad Hoc Networks," *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002),* pages 1678-1687. 2002.

[Camp2002b] Tracy Camp, Jeff Boleng, and Vanessa Davies, "A Survey of Mobility Models for Ad Hoc Network Research," *Wireless Communication & Mobile Computing (WCMC): Special Issue on Mobile Ad Hoc Networking: Research, Trends and Applications,* Vol.2, No.5, pp.483-502, 2002.

[Chen2002] Ing-Ray Chen, Denis Gracanin, Shawn Bohner, Luiz DaSiliva and Scott Midkiff, "Wireless Ad Hoc Messenger," Microsoft Research Grant, Virginia Polytechnic and State University, Microsoft RFP, Section 2: Mobile and Wireless, 2002.

[Compaq2001] Compaq Computer Corporation, "WL110 Wireless PC Card Quick Install Guide," 2001.

[Corson1999] S. Corson and J. Macker, "Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations," IETF RFC 2501.

[Crow1997] Brian P. Crow, Indra Widjaja, Jeong Geun Kim, and Prescott T. Sakai, “IEEE 802.11 Wireless Local Area Networks”, *IEEE Communications Magazine*, pages 116-126, September 1997.

[Frodigh2000] Magnus Frodigh, Per Jonasson, and Peter Larsson, “Wireless Ad Hoc Networking – The Art of Networking without a Network,” *Ericsson Review*, No. 4, 2000.

[Garcia-Luna-Aceves1999] J.J. Garcia-Luna-Aceves and Marcelo Spohn, “Source-Tree Routing in Wireless Networks,” *Proceedings of the Seventh Annual International Conference on Network Protocols*, pages 273, October 31- November 03, 1999.

[Gunes2002] Mesut Gunes and Otto Spaniol, “Routing Algorithms for Mobile Multi-Hop Ad Hoc Networks,” *International Workshop NGNT*, pages 20-24, 2002.

[Hekmat2004] R. Hekmat, “Fundamental Properties of Wireless Mobile Ad-hoc Networks,” *KiVI Telecommunicatieprijs*, Netherlands, March 2004.

[Hu2000] Y.C. Hu and D.B. Johnson, “Caching Strategies in On-Demand Routing Protocols for Wireless Ad Hoc Networks,” *ACM 6th International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 231 – 242, August 2000.

[Hu2002] Yih-Chun Hu and David B. Johnson “Ensuring Cache Freshness in On-Demand Ad Hoc Network Protocols,” *ACM Workshop On Principles Of Mobile Computing, Proceedings of the Second ACM international workshop on Principles of mobile computing*, pages 25-30, 2002.

[Issariyakul2003] Teerawat Issariyakul, Ekram Hossain, and Dong In Kim, “Medium Access Control Protocol for Wireless Mobile Ad Hoc Networks: Issues and Approaches,” *Wireless Communication and Mobile Computing*, pages 935-958, 2003

[Jackquet2001] P. Jackquet, P. Muhlethaler, T. Cluusen, A. Laouiti, A. Qayyum, and L. Viennot, “Optimized Link State Routing Protocol for Ad Hoc Network,” *IEEE INMIC*, Pakistan, 2001.

[Jiang2001] Hong Jiang and J.J. Garcia-Luna-Aceves, "Performance Comparison of Three Routing Protocols for Ad Hoc Networks," *Tenth International Conference on Computer Communications and Networks*, pages 547-554, 2001.

[Johnson1994] David B. Johnson, "Routing in Ad Hoc Networks of Mobile Hosts," *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, pages 158–163, December 1994.

[Johnson1996] David B. Johnson and David A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, Chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

[Johnson2001] David B. Johnson, David A. Maltz, and Josh Broch, "DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks," *Ad Hoc Networking*, edited by Charles E. Perkins, Chapter 5, pages 139–172. Addison-Wesley, 2001.

[LaMaire1996] Richard O. LaMaire, Arvind Krishna, and Pravin Bhagwat, James Panian, "Wireless LANs and Mobile Networking: Standards and Future Directions," *IEEE Communications Magazine*, Vol. 34, Issue 8, pages 86-94, August 1996.

[Lee2000] Sung-Ju Lee, William Su and Mario Gerla, "On-Demand Multicast Routing Protocol (ODMRP) for Ad Hoc Networks," *IETF MANET Working Group*, Internet-draft, 2000. Downloaded from http://www.ics.uci.edu/~atm/adhoc/paper-collection/papers.html

[Leung2001] Leung, R., Jilei Liu, Poon, E., Chan, A.-L.C. and Baochun Li, "MP-DSR: a QoS-aware multi-path dynamic source routing protocol for wireless ad-hoc networks," *Local Computer Networks, Proceedings of the 26th Annual IEEE Conference*, pages 132-141, November 2001.

[Li2001] J. Li, J. Jannotti, D.S.J. De Couto, D.R. Karger and R. Morris, "A Scalable Location Service for Geographic Ad Hoc Routing," *ACM 6th International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 120-130, August 2000.

[Lou2002] Wenjing Lou and Yuguang, F., "The Effects of Cache Organizations on the Performance of On-Demand Routing Protocols in Ad Hoc Networks," *MILCOM*, Vol. 1, pages 254-259, October 2002.

[Maltz1999] David A. Maltz, Josh Broch, Jorijeta Jetcheva, and David B. Johnson, "The Effects of On-Demand Behavior in Routing Protocols for Multi-hop Wireless Ad Hoc Networks," *IEEE Journal on Selected Areas in Communications 17(8),* pages 1439-1453, August 1999.

[Maltz2001] David A. Maltz, "On-Demand Routing in Multi-hop Wireless Mobile Ad Hoc Networks," School of Computer Science Carnegie Mellon University, Ph. D. Thesis. 2001.

[Ogier2004] R. Ogier, F. Templin and M. Lewis, RFC 3684, "Topology Dissemination Based on Reverse-Path Forwarding (TBRPF), Network Working Group, 2004.

[Park1997] V. Park, and S. Corson, "Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification," Internet draft. 1997. Downloaded from http://www.ics.uci.edu/~atm/adhoc/paper-collection/papers.html

[Perkins1994] Charles E. Perkins and Pravin Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *Proceedings of the SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, August 1994.

[Perkins1999] Charles E. Perkins and Elizabeth M. Royer, "Ad-hoc on-demand distance vector routing," *Proceedings of WMCSA99: 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90--100, February 1999.

[Peterson2000] Larry L. Peterson and Bruce S. Davie, *Computer Networks*, A System Approach, Second Edition, Morgan Kaufmann Publisher, 2000.

[Raju2001] Raju, J., and Garcia-Luna-Aceves, J.J., "Scenario-based comparison of source-tracing and dynamic source routing protocols for ad-hoc networks," *Communications, IEEE International Conference,* Vol. 6, pages 1919-1924, June 2001.

[Seet2003] Boon-Chong Seet, Bu-Sung Lee, and Chiew-Tong Lau, “Optimization of Route Discovery for Dynamic Source Routing in Mobile Ad Hoc Networks,” IEE, *Electrics Letters Online*, No. 20031049, July 2003.

[Sheu2002] Shiann-Tsong Sheu, Tobias Chen, Jenhui Chen, and Fun Ye, “The Impact of RTS Threshold on IEEE 802.11 MAC Protocol,” *Proceedings of the Ninth International Conference on Parallel and Distributed Systems*, 2002.

[Su2001] William Su, Sung-Ju Lee, and Mario Gerla, “Mobility Prediction and Routing in Ad Hoc Wireless Networks,” *International Journal of Network Management*, Vol. 11, Issue 1, pages 3-30, January-February 2001.

[Wu2002] Jie Wu, “An Extended Dynamic Routing Source Scheme in Ad Hoc Wireless Networks,” *The 35th Annual Hawaii International Conference on System Sciences (HICSS’02)*, Vol.9, January 2002.

[Xu2001] Y. Xu, J. Heidemann and D. Estrin, “Geography-Informed Energy Conservation for Ad Hoc
Routing,” *The 7th ACM International Conference on Mobile Computing and Networking (MobiCom 2001)*, pages 70-84, 2001.

[Ye2002] J. Ye, “Multi-Hop Ad Hoc Wireless Messenger Using Microsoft Net Compact Framework and Pocket PCs,” Independent Study at Virginia Polytechnic Institute and State University, Fall 2002.

[Zhong2003] Yingji Zhong and Dongfeng Yuan, “Dynamic Source Routing Protocol for Wireless Ad Hoc Networks in Special Scenario using Location Information,” *Communication Technology Proceedings, ICCT, International Conference*, Vol.2, pages 1287-1290, April 2003.

[Zhou1999] L. Zhou and Z. Haas, “Securing Ad Hoc Networks,” *IEEE Network Magazine*, Vol. 13, No. 6, pages 24-30, November/December 1999.

Appendices

Appendix A

This appendix gives all abbreviations of terminologies used in this thesis.

A.1 Abbreviations of Terminologies

ACK	Acknowledgement
AODV	Ad hoc On-demand Distance Vector
AP	Access Point
ARQ	Automatic Repeat Request
BEST	Bandwidth Efficient Source Tracing
BSA	Basic Service Area
BSS	Basic Service Set
CFP	Contention Free Period
CP	Contention Period
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CTS	Clear To Send
dBm	"dBm" notation represents a measured power level in decibels relative to 1mW.
DBPSK	Differential Binary Phase Shift Keying
DCF	Distributed Coordination Function
DIFS	DCF-IFS
DREAM	Distance Routing Effect Algorithm for Mobility
DS	Distributed System
DSDV	Destination-Sequenced Distance Vector
DSR	Dynamic Source Routing
DST	Dynamic Source Tree
DSSS	Direct Sequence Spread Spectrum
DTBS	Distributed Time Bounded Service

DQPSK	Differential Quadrature Phase Shift Keying
ESS	Extended Service Set
FCC	Federal Communications Commission
FDDU	Fiber Distributed Data Interface
FEC	Forward Error Correction
FHSS	Frequency Hopping Spread Spectrum
FRC	Forward Error Correction
GAF	Geographical Adaptive Fidelity
HiperLAN	High Performance Radio Local Area Network
IBSS	Independent BSS
ID	Identification
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IR	Infrared
ISM	Industrial, Scientific, and Medical
LAR	Location Aided Routing
LCC	Logical Link Control
MAC	Medium Access Control
MANET	Mobile Ad-Hoc Network
MN	Mobile Node
MP-DSR	Multi-Path DSR
MPDU	MAC Protocol Data Unit
MSDU	MAC Service Data Unit
NAV	Network Allocation Vector
NDIS	Network Driver Interface Specification
ODMRP	On-Demand Multicast Routing Protocol
OLSR	Optimized Link State Routing
PCF	Point Coordination Function
PDA	Personal Digital Assistant
PDU	Protocol Data Unit
PIFS	Point Coordination Function IFS
P2P	Peer-to-Peer
QoS	Quality of Service
RAM	Random Access Memory

RC4	Ron's Code or Rivest's Cipher, a stream cipher designed by Rivest for RSA Data Security.
RTS	Request To Send
RWM	Random Waypoint Model
SADSR	Security Aware Adaptive Dynamic Source Routing Protocol
SIFS	Short Inter Frame Space
STAR	Source Tree Adaptive Routing
SWS	Sliding Window Size
TBRPF	Topology Dissemination Based on Reverse-Path Forwarding
TORA	Temporarily-Ordered Routing Algorithm
WEB	Wired Equivalent Privacy
WECA	Wireless Ethernet Compatibility Alliance
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Network

Appendix B

This appendix B shows user interfaces of the Statistical Analyzer used as a centralized data collector for all result files from each node and as an analyzer to generate appropriate graphs based on a selected set of x-parameter and y-parameter.

B.1 User Interface of the Statistical Analyzer

Each Pocket PC can transmit any file to this Statistical Analyzer running on a laptop. The left text box shows the files arrived at the Statistical Analyzer. The selected Session and Round indicates the specific directory to store the arriving files. The IP address of the Statistical Analyzer is set manually to 192.168.0.10, as Figure B.1 shows.

After the number of Session and Rounds is entered, the "*Generate Sessions and Rounds Directories to Store Files to be Received*" button will bring the generation of proper Sessions and Rounds. In the above example, there are 7 directories for 7 sessions, and 3 directories for 3 rounds within each of a session directory. Note that a user can designate the path that he or she wants to collect results files from all the nodes in the system in the upper left text box called the "*Designate the Place You Want to Collect Result Files,*" as shown in Figure B.2. Thus, the

directories for the selected number of Sessions and Rounds will be created in the designated directory path.

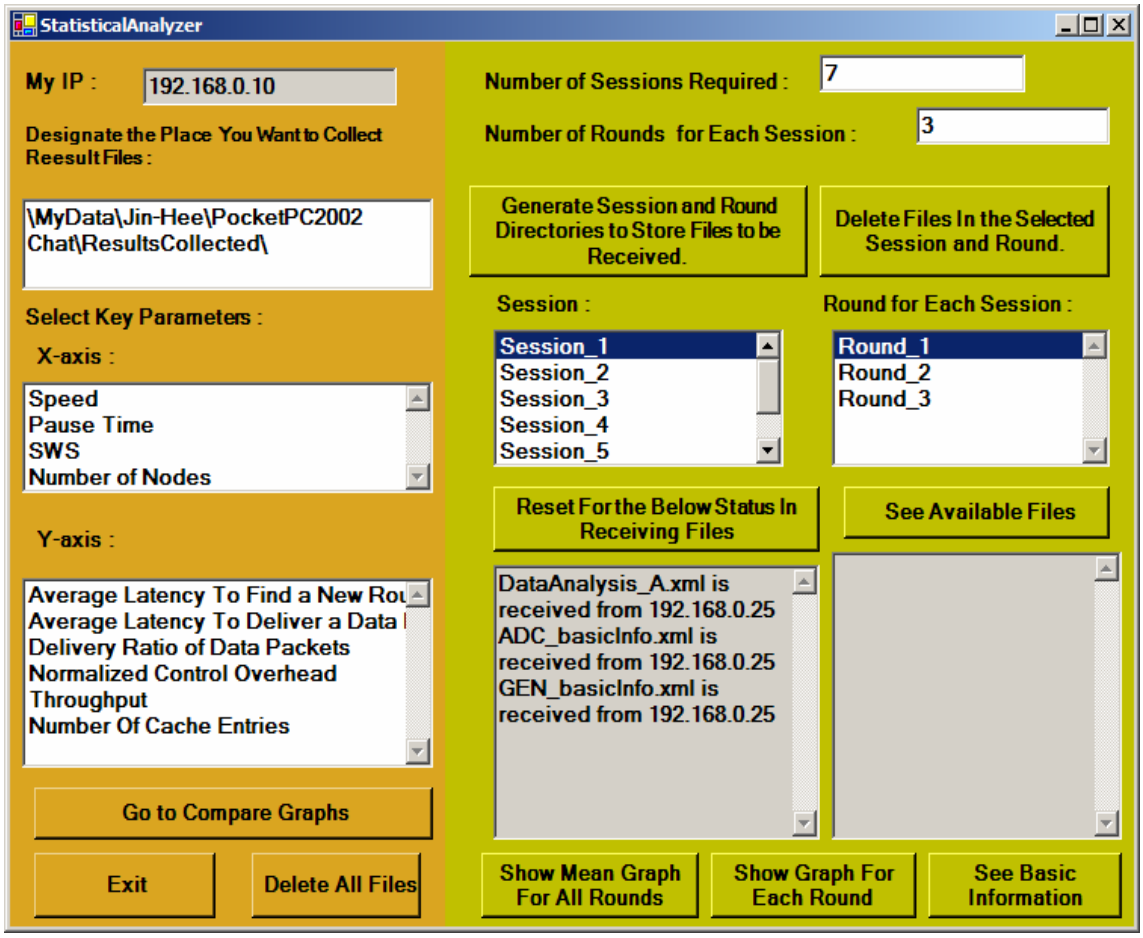

Figure B.1: User Interface of the Statistical Analyzer (1): Wireless File Transmission.

In Figure B.2, the "*See Available Files*" button shows all files arrived under the designated directory in the Statistical Analyzer. Also, a set of x- and y-parameter can be selected on the left side of the interface. X-parameters are the variables that are manipulated to set up a specific scenario. Y-parameters are the five standard metrics to measure the performance of our wireless ad hoc messenger.

The "*Show Mean Graph For All Rounds*" button displays a graph with a mean value of three rounds based on the selected x and y parameter. The "*Show Graph for Each Round*" button demonstrates a graph with a value of each round. Figure B.3 and Figure B.4 show examples of respective graphs, where x-parameter is *Speed* and y-parameter is *Delivery Ratio of Data Packet*.

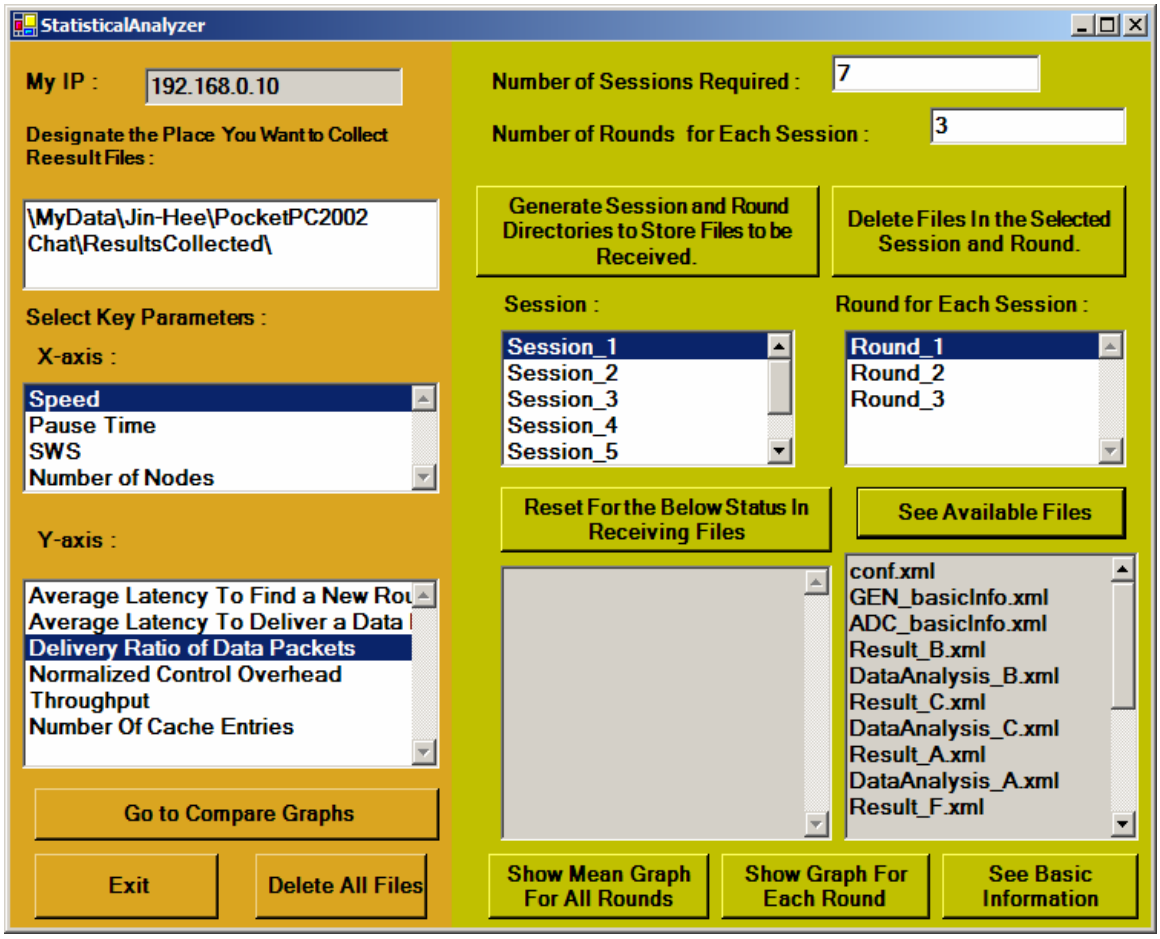

Figure B.2: User Interface of the Statistical Analyzer (2): See Available Files and Select Parameters.

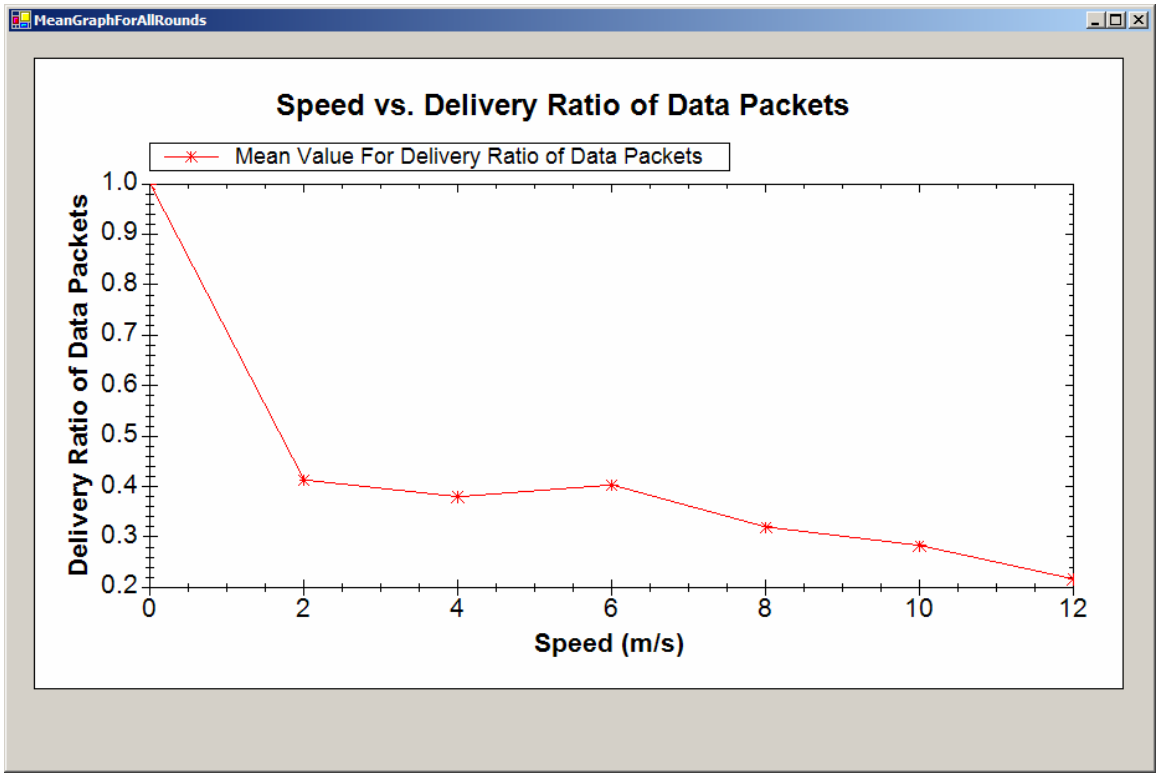

Figure B.3: User Interface of the Statistical Analyzer (3): Show Mean Graph for All Rounds.

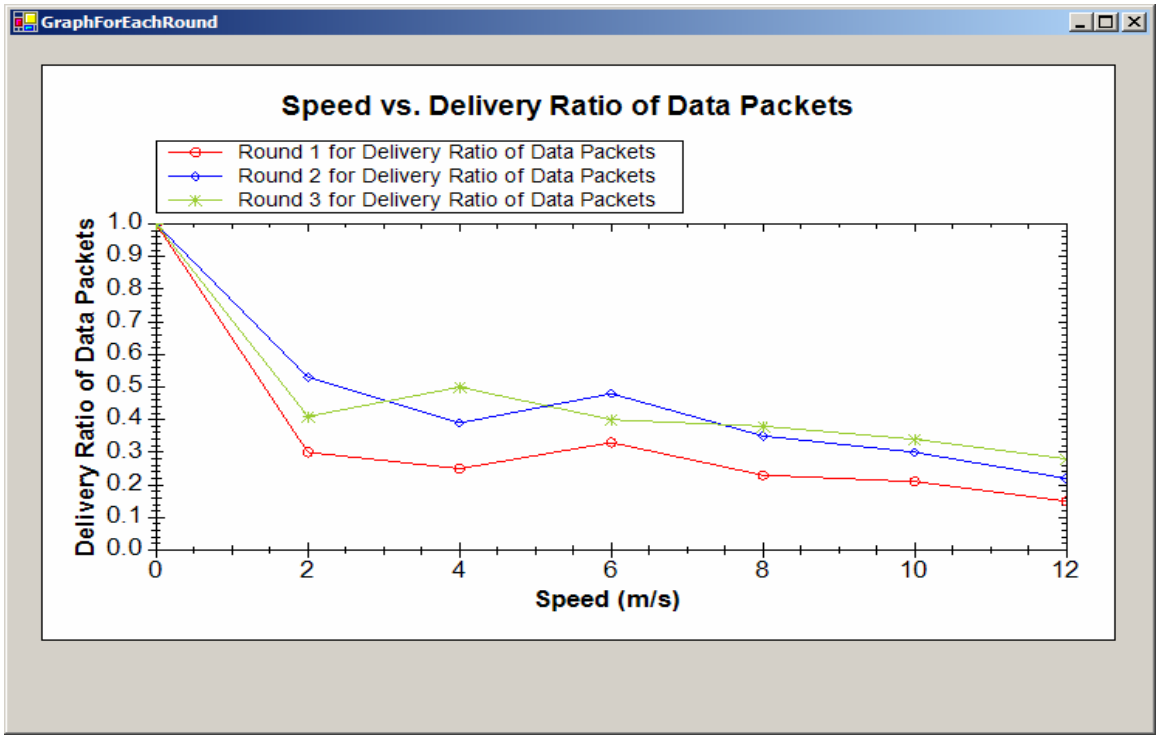

Figure B.4: User Interface of the Statistical Analyzer (4): Show Graph for Each Round.

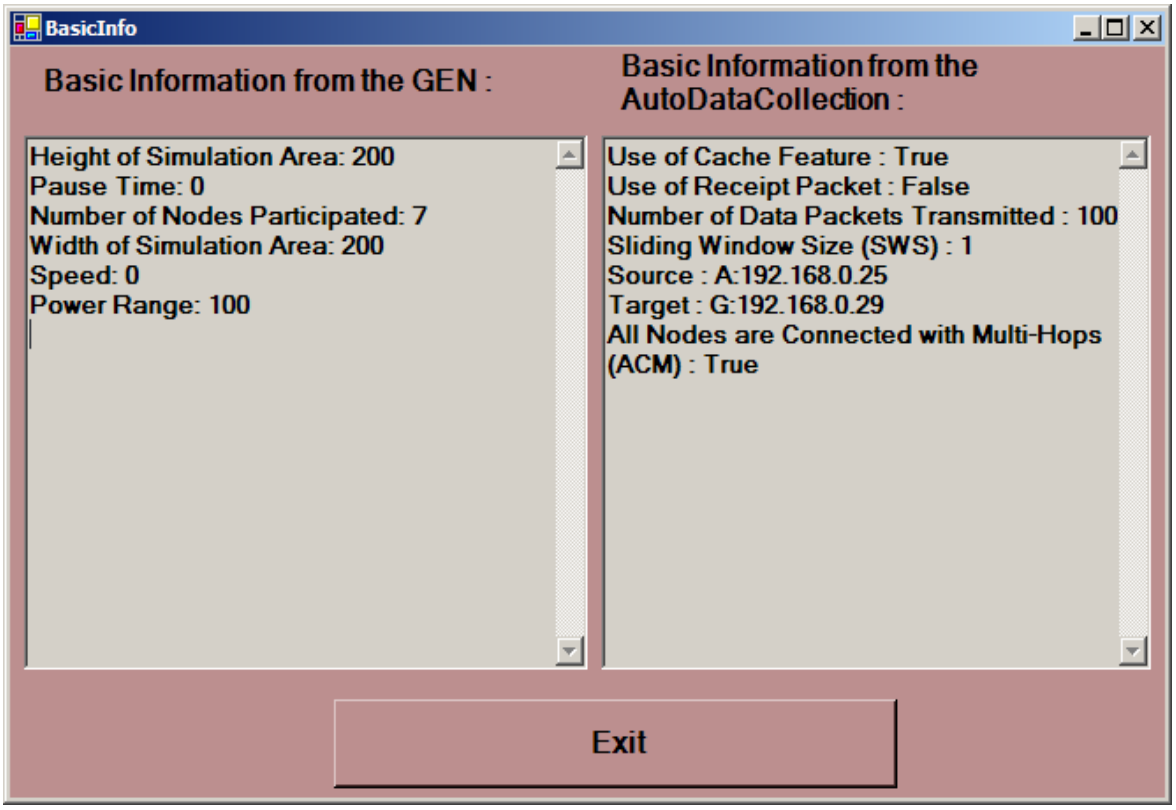

Figure B.5: User Interface of the Statistical Analyzer (5): See Basic Information.

The "*See Basic Information*" button, located at the right bottom in Figure B.1, is used to display the basic information obtained in GEN_basicInfo.xml and ADC_basicInfo.xml files, as shown in Figure B.5.

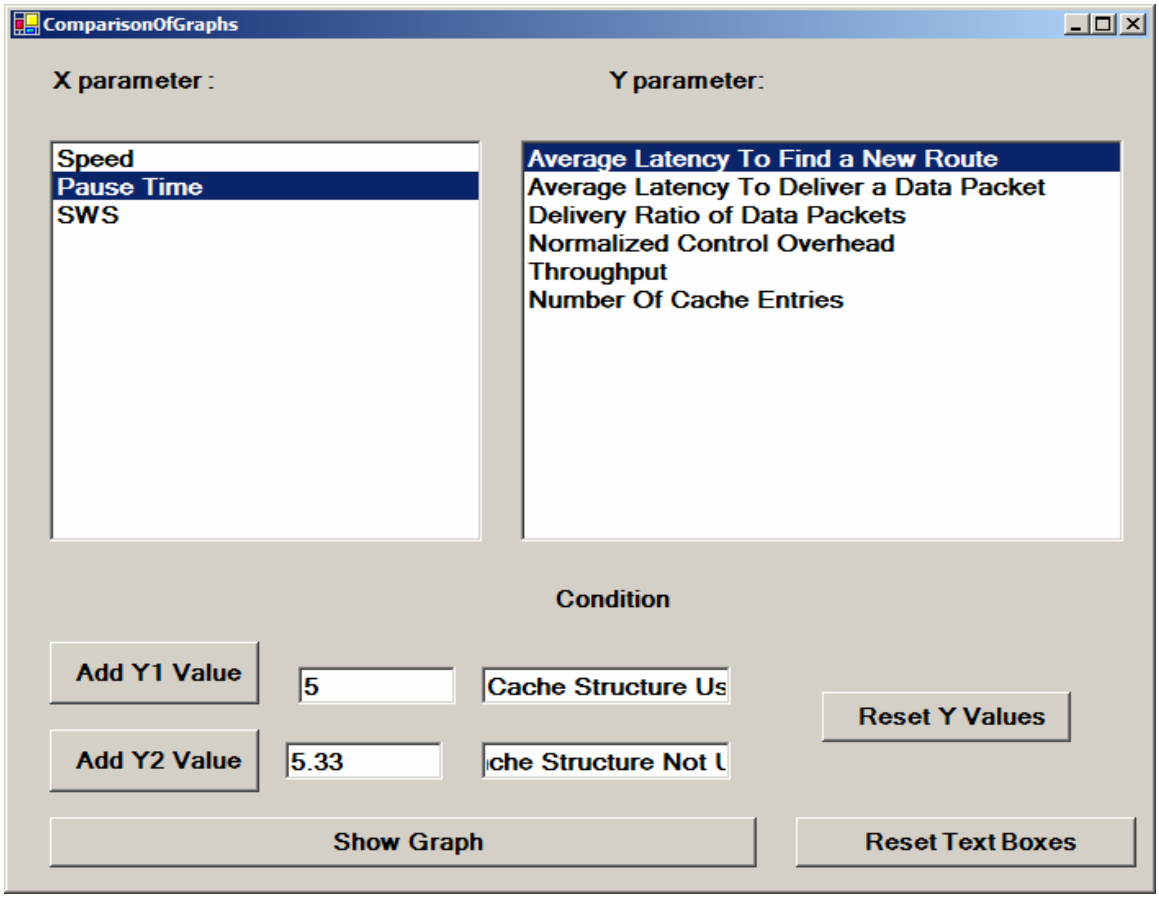

Figure B.6: User Interface of the Statistical Analyzer (6): Comparison of Graphs.

The "*Go to Compare Graphs*" button in Figure B.1 brings the above user interface to display two curves based on two different conditions. X and y- parameters can be entered by a user and y values can be manually added. The specific conditions for each curve are explained in the condition text box. The "*Show Graph*" button shows the specified graph, as shown in Figure B.7.

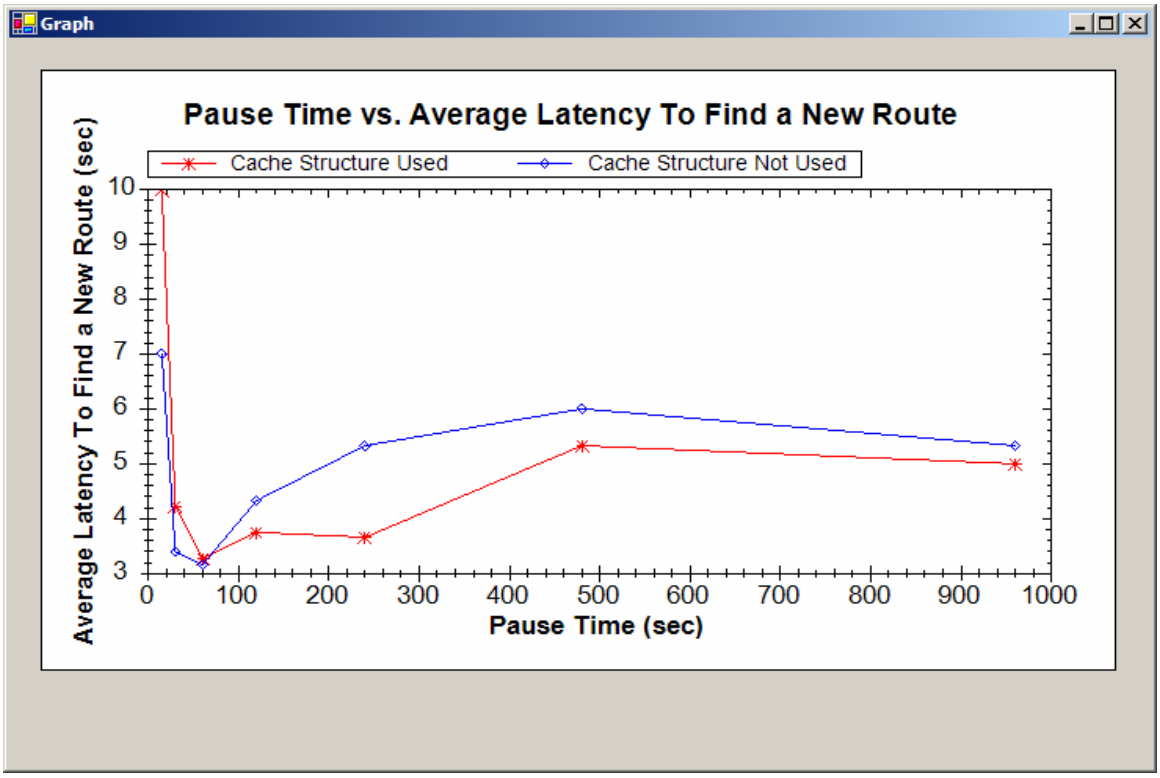

Figure B.7: User Interface of the Statistical Analyzer (7): Example for Comparison of Graphs.

B.2 User Interface of the Send and Receive Files

Figure B.8 shows the user interface for the Send and Receive Files application. This application is developed to transmit any file to the Statistical Analyzer. That is, this application transmits result files in each node wirelessly to the centralized data collector, the Statistical Analyzer.

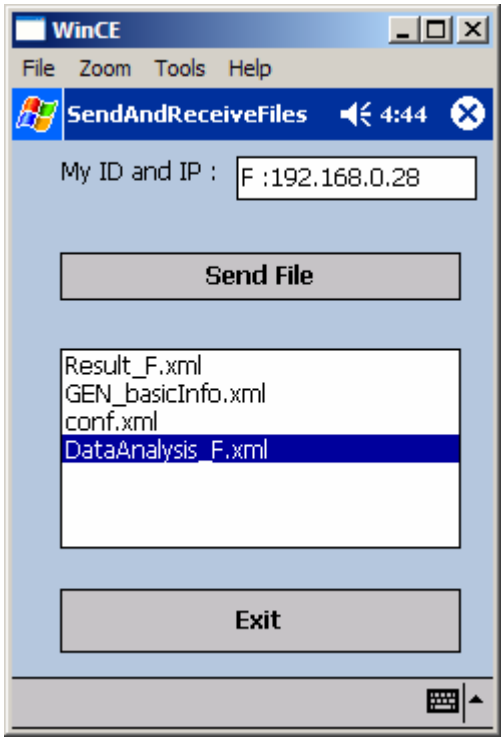

Figure B.8: User Interface of the Send and Receive Files.

Appendix C

C.1 conf. xml

The following XML file named conf.xml is generated based on input information described in another XML file named GEN_basicInfo.xml listed later in Appendix C.2. The input conditions are as follows:

Power Range: 100 m

Simulation Area: 200 m (Width)*200 (Height) m = 40,000 m²

Speed: 0 m/sec

Pause Time: 0 second

Participating Nodes: 7

```
<?xml version="1.0" encoding="us-ascii" ?>
- <Configuration>
  - <Node ID="B" IP="192.168.0.23" StartPt="(169,34)">
```

```
- <Sequence>
  - <Hop>
      <Destination>(34,43)</Destination>
      <Speed>0</Speed>
      <Pause>0</Pause>
    </Hop>
  - <Hop>
      <Destination>(84,143)</Destination>
      <Speed>0</Speed>
      <Pause>0</Pause>
    </Hop>
  - <Hop>
      <Destination>(24,89)</Destination>
      <Speed>0</Speed>
      <Pause>0</Pause>
    </Hop>
  - <Hop>
      <Destination>(28,193)</Destination>
      <Speed>0</Speed>
      <Pause>0</Pause>
    </Hop>
  - <Hop>
      <Destination>(169,165)</Destination>
      <Speed>0</Speed>
      <Pause>0</Pause>
    </Hop>
  - <Hop>
      <Destination>(42,56)</Destination>
      <Speed>0</Speed>
      <Pause>0</Pause>
    </Hop>
  - <Hop>
      <Destination>(176,37)</Destination>
      <Speed>0</Speed>
      <Pause>0</Pause>
    </Hop>
  - <Hop>
      <Destination>(129,107)</Destination>
      <Speed>0</Speed>
      <Pause>0</Pause>
    </Hop>
  - <Hop>
      <Destination>(117,25)</Destination>
      <Speed>0</Speed>
      <Pause>0</Pause>
    </Hop>
  - <Hop>
      <Destination>(104,94)</Destination>
      <Speed>0</Speed>
```

```
            <Pause>0</Pause>
          </Hop>
        </Sequence>
      </Node>
    - <Node ID="C" IP="192.168.0.24" StartPt="(139,73)">
      - <Sequence>
        - <Hop>
            <Destination>(193,87)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(129,67)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(198,197)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(115,42)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(30,44)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(26,47)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(33,19)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(101,199)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(122,144)</Destination>
            <Speed>0</Speed>
```

```
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(69,143)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    </Sequence>
  </Node>
- <Node ID="A" IP="192.168.0.25" StartPt="(178,43)">
  - <Sequence>
    - <Hop>
        <Destination>(91,0)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(41,58)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(9,179)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(105,187)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(67,145)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(46,119)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(51,147)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(3,137)</Destination>
        <Speed>0</Speed>
```

```
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(160,74)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(9,88)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        </Sequence>
      </Node>
    - <Node ID="F" IP="192.168.0.28" StartPt="(148,82)">
      - <Sequence>
        - <Hop>
            <Destination>(72,101)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(143,107)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(138,65)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(64,60)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(104,29)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(86,154)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(54,52)</Destination>
            <Speed>0</Speed>
```

```
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(29,13)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(150,67)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(92,19)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    </Sequence>
  </Node>
- <Node ID="G" IP="192.168.0.29" StartPt="(187,52)">
  - <Sequence>
    - <Hop>
        <Destination>(163,35)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(77,112)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(6,72)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(176,146)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(123,178)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(175,103)</Destination>
        <Speed>0</Speed>
```

```
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(135,94)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(11,10)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(199,142)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(171,20)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
      </Sequence>
    </Node>
  - <Node ID="D" IP="192.168.0.26" StartPt="(157,91)">
    - <Sequence>
      - <Hop>
            <Destination>(57,41)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(83,152)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(129,159)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(74,112)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(147,98)</Destination>
            <Speed>0</Speed>
```

```
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(90,13)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(31,23)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(38,106)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(137,24)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(96,147)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    </Sequence>
  </Node>
- <Node ID="E" IP="192.168.0.27" StartPt="(196,61)">
  - <Sequence>
    - <Hop>
        <Destination>(63,17)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(59,69)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(87,31)</Destination>
        <Speed>0</Speed>
        <Pause>0</Pause>
      </Hop>
    - <Hop>
        <Destination>(57,29)</Destination>
        <Speed>0</Speed>
```

```
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(144,49)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(79,119)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(59,193)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(59,143)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(142,64)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
        - <Hop>
            <Destination>(11,19)</Destination>
            <Speed>0</Speed>
            <Pause>0</Pause>
          </Hop>
      </Sequence>
    </Node>
  </Configuration>
```

C.2 GEN_basicInfo.xml

```
  <?xml version="1.0" encoding="us-ascii" ?>
- <GEN_BasicInfo>
    <NumOfNodes>7</NumOfNodes>
  - <SimulationArea>
      <Width>200</Width>
      <Height>200</Height>
    </SimulationArea>
    <PowerRange>100</PowerRange>
    <Speed>0</Speed>
```

```
    <PauseTime>0</PauseTime>
  </GEN_BasicInfo>
```

C.3 ADC_basicInfo.xml

```
  <?xml version="1.0" encoding="us-ascii" ?>
- <ADC_BasicInfo>
    <Source>A:192.168.0.25</Source>
    <Target>G:192.168.0.29</Target>
    <ACM>True</ACM>
    <Cache>True</Cache>
    <SWS>1</SWS>
    <NumOfDataPacket>100</NumOfDataPacket>
    <ReceiptPacket>False</ReceiptPacket>
  </ADC_BasicInfo>
```

C.4 Result_(NodeID).xml

This XML file is too big to be included and can be downloaded from: http://people.cs.vt.edu/~irchen/microsoft-grant/Website_HTML_Files/.

The experimental conditions used to generate this file are specified in GEN_basicInfo.xml and ADC_basicInfo.xml.

C.5 Data Analysis_(NodeID).xml

This XML file is used to describe performance data of our wireless ad hoc messenger.

```
  <?xml version="1.0" encoding="us-ascii" ?>
- <DataAnalysis ID="A" SourceNode="Yes" TimeNow="238">
  - <XParam>
      <SWS>1</SWS>
      <Speed>0</Speed>
      <PauseTime>0</PauseTime>
    </XParam>
    <Throughput>0.42</Throughput>
    <TotalDataPacketDelivered>100</TotalDataPacketDelivered>
    <AvgLatencyDeliveryingDP>1.06</AvgLatencyDeliveryingDP>
    <AvgLatencyFindingNewRoute>2</AvgLatencyFindingNewRoute>
    <DeliveryRatioOfDP>1</DeliveryRatioOfDP>
    <AvgNumOfCaCheEntries>5.75</AvgNumOfCaCheEntries>
    <TotalNumberOfCPSent>3</TotalNumberOfCPSent>
  </DataAnalysis>
```

Appendix D

D.1 Source Code

The source code for all programs developed in this thesis can be downloaded from http://people.cs.vt.edu/~irchen/microsoft-grant/Website_HTML_Files/.

Vita

Jin-Hee Cho is born in Pusan, South Korea, 1973. She graduated with Bachelor of Art in the Department of Social Welfare from Ewha Womans University, Seoul, South Korea in 1997 with full scholarship in Junior and Senior. She also pursued her master in the George Warren Brown School of Social Work at Washington University, St. Louis, Missouri in 1999. She worked as a social worker for a year. She is expected to finish her master study in Computer Science in Virginia Tech in the summer of 2004.