Mobile Ad-hoc Network Routing Protocols: Methodologies and Applications

Tao Lin

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the

Ph.D.
in
Computer Engineering

Dr. Scott F. Midkiff, Chair
Dr. Luiz A. DaSilva
Dr. Nathaniel J. Davis, IV
Dr. Ira Jacobs
Dr. Charles P. Koelling

March 19, 2004
Blacksburg, Virginia

Keywords: Mobile Ad-hoc Network, Routing,
Relay node set, Framework, Connected dominating set,
Simulation, Emulation

Mobile Ad-hoc Network Routing Protocols: Methodologies and Applications

Tao Lin

Dr. Scott F. Midkiff, Chair

Computer Engineering

Abstract

A mobile ad hoc network (MANET) is a wireless network that uses multi-hop peer-to-peer routing instead of static network infrastructure to provide network connectivity. MANETs have applications in rapidly deployed and dynamic military and civilian systems. The network topology in a MANET usually changes with time. Therefore, there are new challenges for routing protocols in MANETs since traditional routing protocols may not be suitable for MANETs. For example, some assumptions used by these protocols are not valid in MANETs or some protocols cannot efficiently handle topology changes.

Researchers are designing new MANET routing protocols and comparing and improving existing MANET routing protocols before any routing protocols are standardized using simulations. However, the simulation results from different research groups are not consistent with each other. This is because of a lack of consistency in MANET routing protocol models and application environments, including networking and user traffic profiles. Therefore, the simulation scenarios are not equitable for all protocols and conclusions cannot be generalized. Furthermore, it is difficult for one to choose a proper routing protocol for a given MANET application.

According to the aforementioned issues, my Ph.D. research focuses on MANET routing protocols. Specifically, my contributions include the characterization of differ-

ent routing protocols using a novel systematic relay node set (RNS) framework, design of a new routing protocol for MANETs, a study of node mobility, including a quantitative study of link lifetime in a MANET and an adaptive interval scheme based on a novel neighbor stability criterion, improvements of a widely-used network simulator and corresponding protocol implementations, design and development of a novel emulation test bed, evaluation of MANET routing protocols through simulations, verification of our routing protocol using emulation, and development of guidelines for one to choose proper MANET routing protocols for particular MANET applications.

Our study shows that reactive protocols do not always have low control overhead, as people tend to think. The control overhead for reactive protocols is more sensitive to the traffic load, in terms of the number of traffic flows, and mobility, in terms of link connectivity change rates, than other protocols. Therefore, reactive protocols may only be suitable for MANETs with small number of traffic loads and small link connectivity change rates. We also demonstrated that it is feasible to maintain full network topology in a MANET with low control overhead. This dissertation summarizes all the aforementioned methodologies and corresponding applications we developed concerning MANET routing protocols.

To my parents, Lin Hongqing and Zheng Ruilan

my brother and sister, Lin You and Gao Kaiyan

and my wife, Zheng Xing

Acknowledgements

I would like to take the opportunity to thank people who guided and supported me during this process. Without their contributions, this research would not have been possible.

First, I would like to thank my Ph.D. advisor, Dr. Scott F. Midkiff. There is an old fable in China. Once upon a time, a young man meets an old wizard, who can convert stones to gold. The wizard uses his magic to make gold and gives them to the young man. However, the young man does not want to accept the gold because he wants to learn the magic that can make gold out of stones. I think in the past four years, Dr. Midkiff taught me how to do the research. In other words, he taught me how to make gold out of stones. Besides this, he was always patient and helpful whenever his guidance and assistant were needed. As an international student, I felt a certain difficulty in technical writing. Dr. Midkiff also contributed much of his valuable time to help me improve my writing skills. Dr. Midkiff also gave advice on how to communicate and co-operate with other researchers as a professional. Therefore, I do want to show my deep appreciation to him.

I am also grateful to Dr. Luiz A. DaSilva, Dr. Nathaniel J. Davis, Dr. Ira Jacobs, and Dr. Charles P. Koelling for serving on my Ph.D. committee. I would like to thank Dr. DaSilva and Dr. Davis for their valuable advice on the project I was working on via emails, talks, and group meetings. I really appreciate Dr. Koelling for sharing his expertise in simulation and his sense of humor, which strengthened the fun in his simulation class. I also want to thank Dr. Jacobs, and other committee members as well, for spending time reviewing this work and giving valuable suggestions and comments on my work.

There were helps from people in research teams in and outside of Virginia Tech. I can still remember the time that Palan Annamalai, Karthik Channakeshava, Mike Christman, George C. Hadjichristofi, Dr. Jahng S. Park, and John Wells worked together with me. It was my great pleasure to work with them in our research team. I also appreciate valuable discussions with other faculty members and students, including Dr. Thomas

Hou, Peng Li, and Michelle Gong. Their suggestions and comments also encouraged me in these years. Besides this, I also want to thank authors of different routing protocols, especially Dr. Richard Ogier from SRI and Dr. Thomas Clausen from INRIA, France, for valuable discussions on their work. During the procedure that I developed the emulation test bed, I also received valuable information from online posts provided by authors around the world. Therefore, I want to thank all of them too.

I would like to show my great appreciation to my parents for their continuous support during all these years. They were the first ones that introduced the amazing world to me and encouraged me to explore the wonderful nature. My brother and sister always guide me during my explorations up to today, using their valuable experience and successful examples. Last, but not least, my wife, Xing Zheng, also contributed much of her time and efforts to support me during my study. Without any one of them, the work would not have been possible. Thus, I also want to thank them for their love, support, and encouragement.

This work was supported in part by funding from the U.S. Office of Naval Research (ONR) through the Navy Collaborative Integrated Information Technology Institutive (NAVCIITI). This support is gratefully acknowledged.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Today's Internet has been developed for more than forty years. Recently many network researchers are studying networks based on new communication techniques, especially wireless communications. Wireless networks allow hosts to roam without the constraints of wired connections. People can deploy a wireless network easily and quickly. End users can move around while staying connected to the network. Wireless networks play an important role in both military and civilian systems. Handheld personal computer connectivity, notebook computer connectivity, vehicle and ship networks, and rapidly-deployed emergency networks are all applications of this kind of network.

Hosts and routers in a wireless network can move around. Therefore, the network topology can be dynamic and unpredictable. Traditional routing protocols used for wired networks cannot be directly applied to most wireless networks because some common assumptions are not valid in this kind of dynamic network. For example, one assumption is that a node can receive any broadcast message sent by others in the same subnet. However, this may not be true for nodes in a wireless mobile network. The bandwidth in this kind of network is usually limited. Thus, this network model introduces great challenges for routing protocols.

Many mobile ad hoc network (MANET) routing protocols have been proposed. Past work focused on designing new protocols, comparing existing protocols, or improving protocols before standard MANET routing protocols are defined. Most research in this field is based on simulation studies of the routing protocols of interest in arbitrary networks with certain traffic profiles. However, the simulation results from different research groups are not consistent. This is because of the lack of consistency in MANET routing protocol models and application environments including networking and user traffic profiles. Therefore, simulation scenarios used in past studies are not fair for all protocols and their conclusions cannot be generalized. Furthermore, it is difficult for one to choose a proper routing protocol for a given MANET application.

The aforementioned problems can be aided by a framework that characterizes MANET routing protocols. However, there has been little research on this kind of framework. To our knowledge, we were the first to propose such a framework for MANET routing protocols based on the concept of a relay node set (RNS) [1]. The framework provides a new view of MANET routing protocols and highlights relations among properties of the protocols. It facilitates the design, comparison, and improvement of MANET routing protocols. It also identifies factors that affect the operations of routing protocols and can be used to guide the design of controllable MANET simulation models.

MANET applications usually have bandwidth and, often, power constraints. Therefore, overhead is an important issue for routing protocols in MANETs. Recent research has focused on simulation studies to investigate and improve routing protocols with respect to control overhead. To aid the systematic design and improvement of routing protocols, we used the RNS framework to develop an analytical model for control overhead of MANET routing protocols. Aided by this analytical model, we designed a new routing protocol using the concept of a connected dominating set (CDS). This new routing protocol has low control overhead and illustrates the utility of the framework. A prototype Linux implementation has been developed to verify this protocol. A simulation implementation of this protocol, using ns2, is also available.

According to the analytical model drawn from the framework, node mobility is one of the most important MANET characteristics. We defined a relative node mobility model to study the relation between node mobility and link duration in MANETs. We proposed preliminary work on an adaptive interval time control scheme based on a novel neighbor stability criterion. We also introduced a two-state connectivity mobility model for the simulation study of MANET routing protocols which allows easy control of network environment parameters that affect the performance of routing protocols. The research on link lifetime duration, the adaptive interval scheme, and the new mobility model can lead to future research topics.

We investigated a widely-used network simulator ns2, fixing bugs and incorporating assumptions for fair comparisons into the implementations of different routing protocols and the simulator itself. Comparison is done using the improved simulator and implementations. Simulation results show that our new protocol has low control overhead, maintains full topology information, and supports shortest path routing when compared to leading MANET routing protocols.

To emulate a MANET within a normal laboratory, we designed and implemented a dynamic switch that can emulate MANETs using a wired test bed. An integration study of different MANET protocols and applications, including our new routing protocol, verifies our routing protocol in a MANET test bed using this dynamic switch.

To summarize, my research focused on routing protocols for MANETs with emphasis on the methodologies and applications. The contributions include the characterization study of different routing protocols using a novel RNS framework, design of a new routing protocol for MANETs, a study of node mobility including a quantitative study of link lifetime in a MANET and an adaptive interval scheme based on a novel neighbor stability criterion, improvements of a widely-used network simulator and associated protocol models, design and development of a novel emulation test bed, evaluation of MANET routing protocols through simulations, verification of our routing protocol us-

ing emulation, and development of guidelines for one to choose proper MANET routing protocols for particular MANET applications.

The remaining chapters are organized as follows. Chapter 2 presents background, summarizes past work, including wireless networks and related routing protocols, and discusses the problem statement. Chapter 3 introduces our proposed framework and briefly discusses a few applications based on this framework. The new link state routing protocol is presented in Chapter 4. Study of node mobility, including link lifetime and neighbor stability, is summarized in Chapter 5. Simulation and emulation tools are investigated in Chapter 6. Extensive simulations and an integrated study of different MANET protocols and applications, which verifies our new routing protocol, are summarized in Chapter 7. Last, we summarize the results, conclude our contributions, and discuss future research topics in Chapter 8.

Chapter 2

Background and Problem Statement

This chapter discusses wireless mobile networks and presents routing protocols that can support such networks. Most of these routing protocols have draft specifications available on the Internet Engineering Task Force (IETF) web site [2]. Each protocol is illustrated by an example. Lastly, the current status of MANET routing protocols is addressed.

We define some terms as follows before we use them in this document.

A *router* or *node* is a basic hardware/software unit in the network that has the capability to forward packets based on its local routing table. Examples include Routers 1, 2, and 3 in Figure 2.1. A *host* is another basic unit in the network that may attach to or act as a router. It can be either the source or the sink of a data flow in the network. In the example shown in Figure 2.1, three computers, Host 1, Host 2, and Host 3, are three *hosts* in the example network. *Network components* refer to *hosts* and *routers* in a network. A host M is said to be a *neighbor* of another host, say N, if there is a bidirectional link between M and N. We can also say that hosts M and N are neighbors. For example, Router 3 and Host 3 in Figure 2.1 are neighbors. Router 1 and Router 2 are also neighbors. A *route* or *path* is a sequence of hosts connecting two end hosts or routers within which any two adjacent hosts in the route or path are neighbors. For example, a path between Host 1 and Host 2

contains Host 1, Router 1, Router 2, and Host 2.

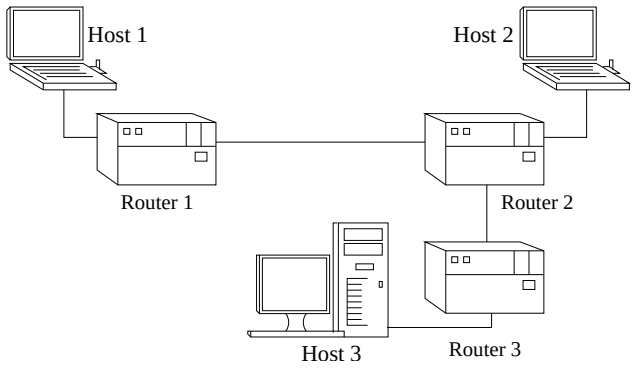


Figure 2.1: Illustration of some terms used in this document.

2.1 Wireless Networks

Like traditional wired networks, wireless networks are formed by routers and hosts. In a wireless network, the routers are responsible for forwarding packets in the network and hosts may be sources or sinks of data flows. The fundamental difference between wired and wireless networks is the way that the network components communicate. A wired network relies on physical cables to transfer data. In a wireless network, the communication between different network components can be either wired or wireless. Since wireless communication does not have the constraint of physical cables, it allows a certain freedom for the hosts and/or routers in the wireless network to move. This is one of the advantages of a wireless network.

Network components in a wireless network communicate with others using wireless channels. Different radio frequency (RF) spectrum ranges are used in wireless networks, for example, 27.5-29.5 GHz for the Local Multipoint Distribution System (LMDS) [3], 2.5-2.7 GHz for the Multipoint Multichannel Distribution System [4], and 5.15-5.35 GHz and 2.4-2.58 GHz for IEEE 802.11a [5] and 802.11b [6], respectively.

Signal strength in a wireless medium decreases when the signal travels further.

When the signal travels beyond a certain distance, the strength reduces to the point where reception is not possible [7]. The distance that the signal travels when it reaches this point is called the radio range for this signal. To simplify the transmission model regarding this property, people assume that the wireless signal is strong enough for the receivers to receive the signal if the receivers are inside of the radio range. Otherwise, the receivers cannot receive the signal at all. The details of wireless communications are not in the scope of this dissertation. Refer to [7] for details.

Several medium access control (MAC) protocols are used in wireless networks to manage the use of the wireless medium. Examples include the Bluetooth MAC layer protocol [8] and IEEE 802.11 MAC layer protocol [6]. This research focuses on the network layer. Thus, the details of these MAC protocols are beyond our scope. Refer to [6] and [8] for more details.

Because radio range is usually limited and the network components may have some mobility, the topology of a wireless network can vary with time. According to the relative mobility of hosts and routers, there are three different types of wireless networks.

1. **Fixed wireless network**. Fixed hosts and routers use wireless channels to communicate with each other and form a fixed wireless network. An example is a wireless network formed by fixed network devices using directed antennas, as shown in Figure 2.2.

2. **Wireless network with fixed access points**. Mobile hosts use wireless channels to communicate with fixed access points, which may act as routers for those mobile hosts, to form a mobile network with fixed access points. An example is a number of mobile laptop users in a building that access fixed access points, as illustrated in Figure 2.3.

3. **Mobile ad hoc network**. A mobile ad hoc network is formed by mobile hosts. Some of these mobile hosts are willing to forward packets for neighbors. Exam-

ples include vehicle-to-vehicle and ship-to-ship networks that communicate with each other by relying on peer-to-peer routings, as shown in Figure 2.4.

Generally speaking, traditional routing protocols that are used in wired networks can support routing in fixed wireless networks and mobile networks with fixed access points. Only one-hop routing is required over a link in a wireless network with fixed access points and many fixed wireless network. Routing in mobile ad hoc networks and some fixed wireless networks use multiple-hop routing. Routing protocols for this kind of wireless network should be able to maintain paths to other nodes and in most cases, must handle changes in paths due to mobility. Traditional routing cannot properly support routing in a MANET. This research focuses on mobile ad hoc routing.

To simplify the description of a MANET, the MANET model is usually illustrated as shown in Figure 2.5. Nodes *i*, *j*, and *k* are mobile nodes in the network. The dashed circles shown in the figure imply the radio coverage areas of nodes. In wireless networks, node *i* can hear node *j* if *i* is within the radio range of *j*. Node *i* is a neighbor of node *j* if node *j* can also hear node *i*. This is called a bi-directional connection. Two nodes are disconnected if one node is not in the radio range of the other. For example, nodes *j* and *k* are disconnected in the figure.

The nodes in a MANET can usually move. Generally, the connection between two nodes is lost when a node moves out of another node's radio range. Thus, the network topology may change from time to time. We call this kind of MANET a *truly mobile ad hoc network*.

There is another kind of MANET besides the truly mobile ad hoc network. All nodes in this type of ad hoc network are fixed. The connectivity between components is relatively static. For example, a sensor network is typically a fixed ad hoc network [9]. Network components in a sensor network are wireless sensors instead of general-purpose computers. They are often deployed in a large area. The sensors can collect information and route data back to a central processor or monitor. Those routes rely on intermediate

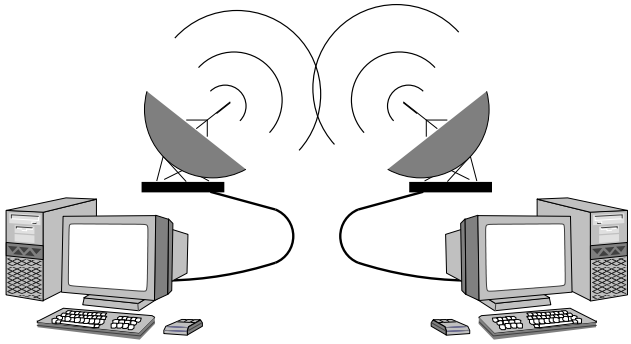


Figure 2.2: An example of a fixed wireless network.

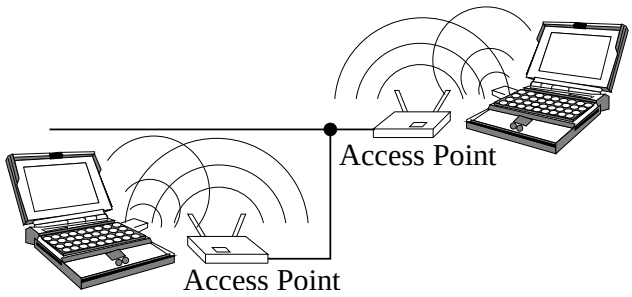


Figure 2.3: An example of a wireless network with access points.

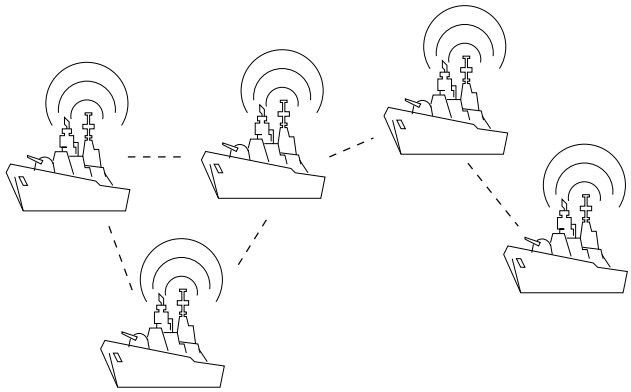


Figure 2.4: An example of a mobile ad hoc network.

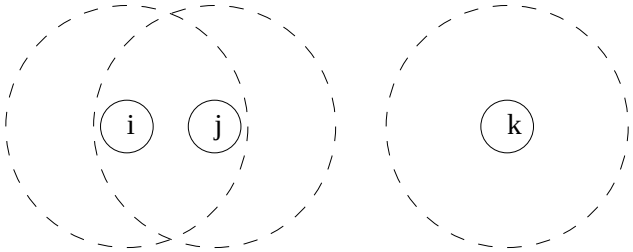


Figure 2.5: A simple network model for a mobile ad hoc network.

sensor nodes. The positions of sensors are often fixed once they are in place. Therefore, the topology is relatively static. Since people often deploy sensors in a random way, for example, by scattering them from an airplane, the topology of a sensor network is often unpredictable. Moreover, the topology may also change when sensors lose power. Therefore, we consider the sensor network as a fixed ad hoc network. Many MANET issues and solutions do apply to fixed ad hoc networks.

A network can be constructed using different wireless architectures. For example, the Office of Naval Research is interested in a wireless network model called a Virtual Operations Network (VON). A VON is a rapidly-deployed wireless network formed by naval vessels at sea. It may be deployed with little advanced planning. Ships can be considered as routers in the wireless network. There are two possible ways to set up a wireless network for a VON, as a mobile network with a fixed backbone network or as a MANET. A mobile network with fixed backbone network is shown in Figure 2.6. The backbone network is formed by some ships using satellite connections. Ships in the backbone act like wireless routers for other ships that attach to them via wireless channels. This model was examined by Wells [10]. The MANET approach considers all ships to be mobile nodes in a MANET. Ships communicate with one another with forwarding services provided by intermediate ships, as shown in Figure 2.4.

Usually, a node in a MANET is mobile with a certain static reference, say another node in the net. Besides this concept of mobility, there is another kind of mobility being studied. For example, in wired or wireless networks, hosts or a subnet as a whole may

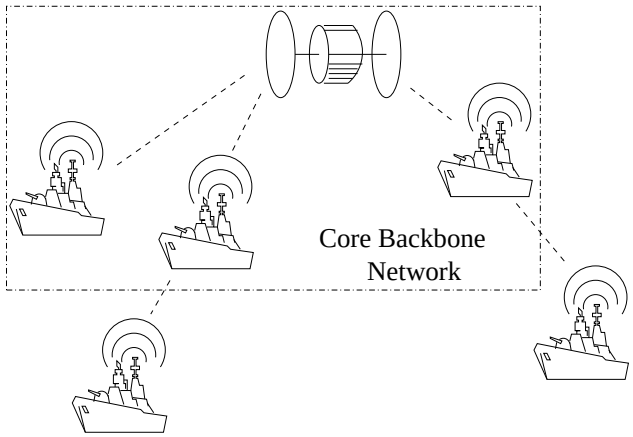


Figure 2.6: One possible network model for the VON.

move from one location to another. Traditional networks require re-configurations of IP addresses used by these mobile hosts or subnets at the new locations. A network enabled with Mobile IP [11] allows these mobile hosts or subnets to move around without any manual address re-configuration. Moreover, those roaming nodes can remain connected with others while they are moving. Since Mobile IP also supports a certain mobility in the network, we can consider it as a potential technique to support routing in some types of MANETs. Figure 2.7 shows a simple example of a mobile node moving in a network using Mobile IP. For more details, refer to Section 2.3.9.

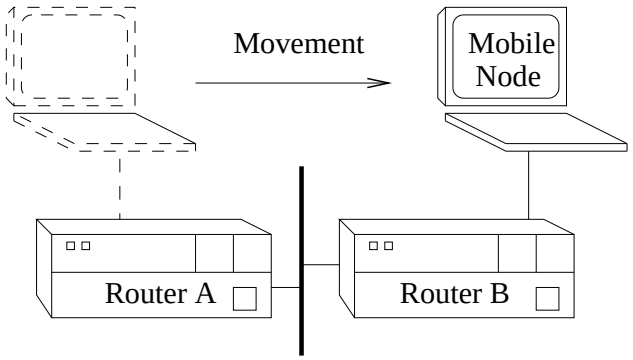


Figure 2.7: An example of a wireless network using Mobile IP.

Traditional routing protocols, such as Open Shortest Path First (OSPF) [12], can be used in some wireless networks with infrastructure. However, they cannot be directly

applied to most MANETs since they are based on assumptions that only apply in wired networks. Hence, some protocols have been proposed to solve the routing problem in MANETs. Details are presented in Section 2.3.

2.2 General Issues for Wireless Ad Hoc Network Routing

Guaranteeing delivery and the capability to handle dynamic connectivity are the most important issues for routing protocols in wireless mobile ad hoc networks. Once there is a path from the source to the destination for a certain period of time, the routing protocol should be able to deliver data via that path. If the connectivity of any two nodes changes and routes are affected by this change, the routing protocol should be able to recover if an alternate path exists. There are some other issues related to routing in wireless ad hoc networks. Whether to consider them depends on the specific environment or application. For example, overhead is particularly important in a wireless network with limited bandwidth [13, 14, 15, 16, 17, 18]. Power consumption may also be a problem in an ad hoc network with battery-powered nodes [19, 20, 21, 22]. Quality of service may be required in an ad hoc network supporting delay sensitive applications such as video conferencing [23, 24, 25, 26]. A routing protocol may need to balance traffic based on the traffic load on links [27, 28]. Scalability of routing protocols is an important issue for large networks [29, 30]. The routing protocol may need to implement security to protect against attacks, such as sniffer, man-in-the-middle, or denial of service [31, 32, 33]. Routing protocols may rely on information based on other layers. For example, the Global Positioning System (GPS) can be used in wireless ad hoc networks deployed in battlefields or connecting vehicles [34, 35]. Mobility prediction can improve routing in wireless networks with known movement patterns, such as the IRIDIUM system satellite network [36]. Information from the medium access control layer may be propagated to the network layer so that neighbors can be detected via MAC layer protocols [35]. The power of received signals from a neighboring node can be used to decide whether this

neighbor is moving closer or further away [34]. Cross-layer design in MANET is also a "hot" topic in the design of new MANET protocols [37, 38]. There is also an increasing number of papers published that systematically develop frameworks to study MANET systems [39, 40].

The design and evaluation of routing protocols are based on, but not limited to, these aspects. I consider some of these factors in my research. Details are discussed in the following chapters.

2.3 Routing Protocols and Techniques

Routing protocols for different types of wireless networks have been proposed by a number of researchers. Researchers traditionally classify these protocols as proactive protocols, reactive protocols, or hybrid of the two, based on the way they find new routes or update existing ones. Proactive routing protocols keep routes continuously updated, while reactive routing protocols react on demand [42]. Routing protocols can also be classified as link state protocols [43] or distance-vector protocols [44]. Routers using a link state routing protocol maintain a full or partial copy of the network topology and costs for all known links. Routers using a distance-vector protocol keep only information about next hops to adjacent neighbors and costs for paths to all known destinations. Generally speaking, "link state routing protocols are more reliable, easier to debug and less bandwidth-intensive than distance-vector" protocols [45]. Link state protocols are also more complex and more compute- and memory-intensive.

Table 2.1 lists some available routing protocols for mobile ad hoc networks. These protocols are briefly discussed in the following sections. There are some previous protocols, such as the Source Tree Adaptive Routing (STAR) protocol [46] and the Partial Tree-Sharing Protocol (PTSP) [47], which are not the focus of active investigation now and their ideas are similar to more recently proposed protocols, such as the Topology

Broadcast Based on Reverse-Path Forwarding (TBRPF) protocol [48]. We are not going to discuss the other protocols in this document.

Table 2.1: Some Routing Protocols for MANETs

	Reactive Protocols	*Proactive Protocols*
Link State Protocols	DSR, TORA	OLSR, TBRPF, TORA, LANMAR/FSR
Distance Vector Protocols	AODV	

In the following sections, we present routing protocols and techniques in more detail. Advantages and disadvantages of these protocols are briefly discussed. Note that OSPF [12] is a widely used routing protocol for traditional wired networks. Although it is not suitable for MANETs, we study the possibility of extending it to MANETs. Mobile IP may also support routing in some MANETs. Thus, it is also discussed here.

2.3.1 Open Shortest Path First (OSPF)

Open Shortest Path First (OSPF) [12] is a link state routing protocol. It is a mature proactive routing protocol widely used in today's wired networks. The basic idea in OSPF is to keep an identical topology database in all routers so that they can build routing tables locally. Routing tables are constructed based on shortest path trees [12]. Because of the properties of the shortest path tree, a route provided by OSPF is loop-free and always the shortest one. OSPF continuously maintains routes to all possible destinations. Hence, it is beneficial for networks with traffic patterns where a large number of hosts in one subnet always communicate with hosts in other subnets. (This is a common advantage of proactive protocols.) OSPF is a complex routing algorithm. Another disadvantage of OSPF is the large overhead of control packets needed to maintain the link state database.

An OSPF network is divided into several indexed areas. Area IDs are manually assigned to all subnets. Each area includes routers in one or more subnets, together with

associated network interfaces. Every area maintains one copy of the link state database in that area. Area 0 is always assigned to the backbone network. Two areas are connected to each other when they share edge routers. Non-backbone areas have to attach to the backbone network. A separate copy of OSPF runs in each area. Hence, gateway routers with multiple interfaces in multiple areas run multiple copies of OSPF. There are two major operations in OSPF, determining adjacency and synchronizing the link state database. Five types of route control packets are used to support these two operations: (i) hello packets, (ii) database description packets, (iii) link state request packets, (iv) link state update packets, and (v) link state acknowledgement packets.

Figure 2.8 illustrates a network using the OSPF routing protocol. Three routers, *A*, *B* and *C*, belong to Subnet I. Routers *C* and *D* are in Subnet II. Assume that Subnet I is configured to be the backbone network, so it is Area 0. Assume that Subnet II is defined as Area 1. The bring-up adjacency operation uses hello packets to detect and maintain adjacency between routers. In our example, Router *B* sends a hello packet to its neighboring router, Router *A*. Router *A* adds the router ID of *B* (there is a unique ID for each router) into its hello message and broadcasts it. When Router *B* receives the hello message originated from *A*, it detects that its ID is in the recently-heard ID list of *A*. This implies a two-way connection between Routers *A* and *B*. Similar procedures happen between other pairs of neighboring routers in this example network.

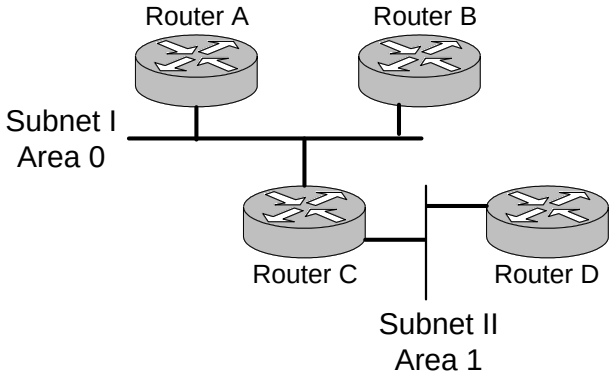


Figure 2.8: Example of the OSPF routing protocol.

After two hosts set up two-way communications with each other, entries in the link state database are exchanged and synchronized by database description packets, link state request packets, and link state update packets. A link state acknowledgement packet is used to acknowledge received packets. In this stage, a master router is selected from these two routers. The master router is responsible for setting the sequence numbers during the following procedures. Each router first briefly declares all records in its link state database. If a record received from a neighbor router is more recent than the corresponding record in its own database, the full copy of that record is requested. Similar operations occur between other pairs of neighboring routers. Thus, the databases of neighbor routers can be synchronized.

In a broadcast network, i.e. with a physical network that supports broadcast, a designated router is selected in each subnet. The designated router broadcasts the ID list of all routers in the same subnet. Other routers only broadcast the ID of their designated router. Therefore, a router can determine all routers in the same subnet by the ID of its designated router and the ID list generated by that designated router. For example, assume Router *B* is the designated router in Subnet I. Router *B* is responsible for broadcasting the list of router IDs in Subnet I, say Routers *A*, *B* and *C*. Routers *A* and *C* only broadcast their designated router ID, Router *B*.

Note that Router *C* has two interfaces in two areas. It is the edge router between Areas 0 and 1. Router *C* is responsible for exchanging link state information between the two areas. For example, in Area 0, Router *C* is the advertisement router for the information originated in Area 1, even if some of the links are not connected to Router *C*. If, later, one of these links is used to form a path, Router *C* is the gateway to reach this selected link. The role of Router *C* in Area 1 is similar. After a router synchronizes its link state database with neighbors, it starts to calculate the shortest path tree. A shortest path algorithm, named Dijkstra's algorithm [41], is used at each router. Routing entries are built based on the computed shortest path tree. Routers broadcast link state changes, such as new links found or broken links detected, to all their known neighbors. Shortest path

trees are rebuilt if necessary.

2.3.2 Dynamic Source Routing (DSR)

The Dynamic Source Routing (DSR) [49] protocol is a distance-vector routing protocol for MANETs. When a node generates a packet to a certain destination and it does not have a known route to that destination, this node starts a route discovery procedure. Therefore, DSR is a reactive protocol. One advantage of DSR is that no periodic routing packets are required. DSR also has the capability to handle unidirectional links. Since DSR discovers routes on-demand, it may have poor performance in terms of control overhead in networks with high mobility and heavy traffic loads. Scalability is said to be another disadvantage of DSR [29], because DSR relies on blind broadcasts to discover routes.

There are two main operations in DSR, route discovery and route maintenance. Figure 2.9 shows a simple example for DSR. Routers A, B, and C form a MANET. Routers A and C are disconnected, while both of them connect to router B. Assume that at the beginning, the route caches that memorize previous routes in the routers are empty. When Router A wants to send a packet to Router C, it broadcasts a route request to start the corresponding route discovery procedure. Router B receives the request since it is within the radio range of A. Router C is the destination in the request and B does not have a route entry to C in its cache at this time. Hence, Router B appends its own ID to the list of intermediate router IDs in the request and rebroadcasts it. When C receives the broadcast route request message originated by B, it determines that the destination ID matches its own ID. Thus, the route from A to C is found. To help the initiator and all intermediate routers construct proper routing entries, Router C sends a reply back to A using source routing if links are bi-directional. This procedure is feasible because all intermediate routers are in the ID list of the corresponding route request. Intermediate routers construct proper routing tables when they receive the reply originated from C. Thus, a route from A to C is built.

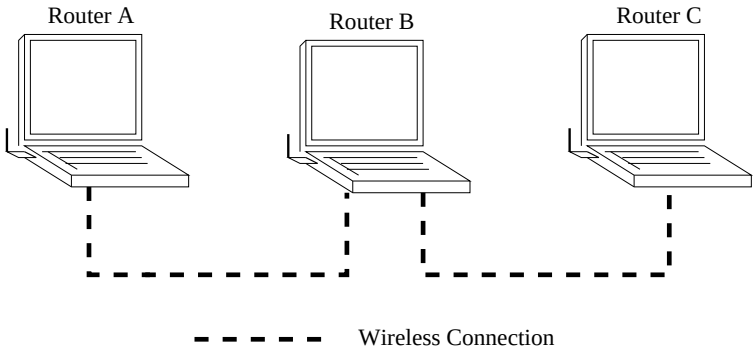


Figure 2.9: Example of DSR and AODV routing protocols.

During the route discovery procedure, routers maintain ID lists of the recently seen requests to avoid repeatedly processing the same route request. Requests are discarded if they were processed recently since they are assumed to be duplicates. If a router receives a request and detects that the request contains its own ID in the list of intermediate routers, this router discards the request to avoid loops.

The route maintenance procedure is used when routes become invalid due to the unpredictable movement of routers. Each router monitors the links that it uses to forward packets. Once a link is down, a route error packet is immediately sent to the initiator of the associated route. Therefore, the invalid route is quickly discarded [42].

To handle unreliable transmissions of control messages, DSR either relies on the underlying MAC protocol to provide guaranteed delivery or it retransmits control messages for a certain number of times. Since DSR is a reactive protocol, it cannot tell whether a destination is unreachable or the route request is lost. Therefore, it suffers more overhead if the underlying MAC layer does not support guaranteed delivery. This is a common problem for reactive routing protocols because when no reply message is heard, routers with a reactive routing protocol cannot tell the difference between the case of a transmission error and the case of unreachable nodes. Reactive routing protocols try to use extra acknowledgements or a small number of retransmissions to solve this problem and, thus, introduce more overhead. Proactive routing protocols periodically broadcast

control messages and remove local routing entries if they time out. Hence, they do not have this problem. But, of course, the the periodically broadcast control messages contribute to overhead.

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

The Ad hoc On-demand Distance Vector (AODV) routing protocol [50] is a reactive MANET routing protocol. Similar to DSR, AODV broadcasts a route request to discover a route in a reactive mode. The difference is that in AODV, a field of the number of hops is used in the route record, instead of a list of intermediate router addresses. Each intermediate router sets up a temporary reverse link in the process of a route discovery. This link points to the router that forwarded the request. Hence, the reply message can find its way back to the initiator when a route is discovered. When intermediate routers receive the reply, they can also set up corresponding forward routing entries. To prevent old routing information being used as a reply to the latest request, a destination sequence number is used in the route discovery packet and the route reply packet. A higher sequence number implies a more recent route request. Route maintenance in AODV is similar to that in DSR [42]. One advantage of AODV is that AODV is loop-free due to the destination sequence numbers associated with routes. The algorithm avoids the Bellman-Ford "count to infinity" problem [50]. Therefore, it offers quick convergence when the ad hoc network topology changes which, typically, occurs when a node moves in the network [50]. Similar to DSR, poor scalability is a disadvantage of AODV [29].

We use the example topology shown in Figure 2.9 to illustrate the discovery procedure of AODV. Note that Routers A and C are disconnected from each other while both of them connect to B. When Router A starts a route discovery to C, a route request is broadcast. The request packet contains the requested destination sequence number, which is 1 greater than the one currently kept at A. For example, assume that the destination sequence number for C at A is 0x00000000, then the destination sequence number in the

route discovery packet is 0x00000001. The intermediate routers reply to the source if they know the route to that destination with the same or higher destination sequence number. We assume that *B* does not have a record for a route to *C*. Therefore, *B* first sets up a temporary link pointing back to *A*. In the second step, it increases the number of hops by 1 and rebroadcasts the request. When *C* receives that request, it creates a new destination sequence number. A route reply with that new sequence number is sent by *C*. The initiator and all intermediate routers build routing entries associated with this new sequence number when they receive the reply. The number of hop values can be used to find a shorter path if a router receives two replies with the same destination sequence number. AODV uses a similar scheme as DSR to handle unreliable transmission of control messages.

2.3.4 Temporally-Ordered Routing Algorithm (TORA)

The Temporally-Ordered Routing Algorithm (TORA) is "an adaptive routing protocol for multihop networks" [51]. TORA is a distributed algorithm so that routers only need to maintain knowledge about their neighbors. TORA also maintains states on a per-destination basis like other distance-vector algorithms. It uses a mix of reactive and proactive routing. Sources initiate route requests in a reactive mode. At the same time, selected destinations may start proactive operations to build traditional routing tables. Usually, routes to these destinations may be consistently or frequently required, such as routes to gateways or servers. TORA supports multiple path routing. It is said that TORA minimizes the communication overhead associated with adapting to network topology changes [51]. The reason is that TORA keeps multiple paths and it does not need to discover a new route when the network topology changes unless all routes in the local route cache fail. Hence, the trade off is that since multiple paths are used, routes may not always be the shortest ones.

TORA uses the concept of *height* associated with a certain destination to describe

the routing metric used by routers. Like water flows in pipes, routers with higher heights may forward packet flows to neighbors with lower heights. Note that since heights for routers are associated with particular destinations, the paths to forward packets are also associated with the corresponding destinations. In networks using TORA, an independent copy of TORA runs for each possible destination. So for different destinations, routers may have different heights and links can have different directions. In the following paragraphs when we discuss the operation of TORA, we assume that it is a copy of TORA with respect to a certain destination.

TORA assigns directions to all links according to the heights of their neighboring routers in terms of *upstream* or *downstream*. A link is an upstream link for the "lower" neighboring router. At the same time, it is also a downstream link for the "higher" neighboring router. An upstream link for a router implies that data flows to the corresponding destination can only come into this router via that link. A downstream link for a router means data flows can only leave this router to the neighboring router via this link. Figure 2.10 gives an example of the concept of heights of routers and the direction of links. Routers *A* and *C* have higher heights than the other routers. The destination router has the lowest height among all routers. The link between Routers *A* and *B* is a downstream link for *A* and is an upstream link for *B*. Note that Router *C* has a higher height than *B* although *C* is one hop away from the destination. This is because the height assignment algorithm used in TORA does not always favor the shortest path.

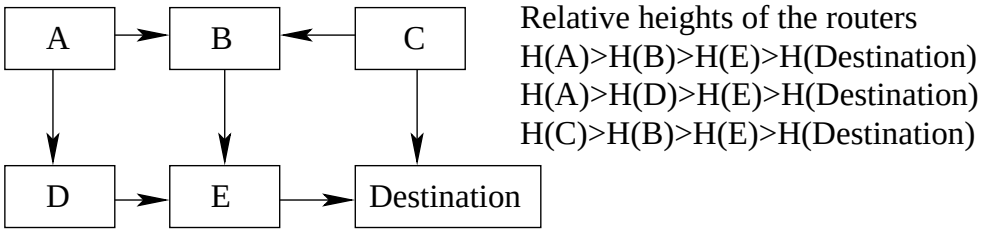


Figure 2.10: Example of TORA routing protocol.

TORA is a complex algorithm compared with DSR. It has four operations: (i) creating routes, (ii) maintaining routes, (iii) erasing routes, and (iv) optimizing routes. The

creating routes operation is responsible for selecting proper heights for routers and forming a directed sequence of links leading to the destination in a previously undirected network. The maintaining routes procedure is the operation that responds to network topology changes. The operation of erasing routes is used to set routers' heights to NULL and set links to undirected. TORA uses the optimizing routes function to adjust the heights of routers to improve routing. Four packets are used to perform these operations: query (QRY), update (UPD), clear (CLR), and optimization (OPT) [51].

We use the example shown in Figure 2.9 to describe the procedure for creating routes in the reactive mode. Note that Routers *A* and *C* are disconnected from each other but both are connected to *B*. Assume that Router *A* wants to send a packet to *C*. TORA initiates the procedure by sending a QRY packet. Router *B* rebroadcasts the request since *B* is not the destination and it does not know a route to *C*. Router *B* sets a route-requested flag so that it can discard any further QRY packets received for the same destination before it knows how to get to *C*. When *C* gets the QRY packet, it replies with an UPD packet. This UPD packet identifies the relevant destination and the height of the source of the UPD packet. To reduce redundant replies to a given route request, Router *C* maintains the time at which the UPD packet was last sent and the time at which links to neighboring routers became active. Because of the preset route-requested flag, when Router *B* receives the UPD packet, it updates its local routing table by adding an entry with *C* as the destination. It also sends a new UPD to its known neighbors. The new UPD packet contains the relevant destination and the height of *B*. In this way, all routers in the network choose proper heights according to the height of *C*. All associated links are assigned directions based on those heights.

The proactive mode of route discovery uses OPT packets. In Figure 2.9, we assume that Router *C* is the destination of a QRY packet. It initiates this procedure by sending out an OPT packet, identifying itself as the destination of the routes, the proactive mode, and the height of *C*. The OPT packet contains a sequence number that is used to uniquely identify this packet and ensure that each router processes and forwards the given OPT

packet at most once. When Router *B* receives the OPT packet, it sets the mode of the operation to proactive mode, then resets the height. Like the procedure for forwarding UPD packets in the reactive mode, Router *B* sends a new OPT packet to its neighbors with the relevant destination and the new height of Router *B*.

A router *i* is said to have no downstream links if all its neighbor routers have higher heights than its height. If a router has no downstream links, packets to the corresponding destination are not able to leave this router. The route maintenance procedure is used to solve this problem once a router determines that it has no downstream links. The procedure modifies the height of the router with no downstream links. Once a router changes its height with respect to a certain destination, it broadcasts the new height via an UPD packet. Neighboring routers modify their heights as appropriate when they receive this UPD packet.

When a partition of the network is detected, which usually occurs in the procedure to maintain routes, the router with no downstream links starts the procedures to erase routes. The node broadcasts a CLR packet with the relevant destination and related parameters. Routers reset heights and set link directions after the procedure to erase routes.

2.3.5 Optimized Link State Routing Protocol (OLSR)

The Optimized Link State Routing (OLSR) protocol [53] is a proactive link state routing protocol for MANETs. One key idea is to reduce control overhead by reducing the number of broadcasts as compared with pure flooding mechanisms. The basic concept to support this idea in OLSR is the use of multipoint relays (MPRs) [53, 54]. MPRs refer to selected routers that can forward broadcast messages during the flooding process. To reduce the size of broadcast messages, every router declares only a small subset of all of its neighbors. "The protocol is particularly suitable for large and dense networks" [53].

MPRs act as intermediate routers in route discovery procedures. Hence, the path discovered by OLSR may not be the shortest path. This is a potential disadvantage of OLSR.

OLSR has three functions: packet forwarding, neighbor sensing, and topology discovery. Packet forwarding and neighbor sensing mechanisms provide routers with information about neighbors and offer an optimized way to flood messages in the OLSR network using MPRs. The neighbor sensing operation allows routers to diffuse local information to the whole network. Topology discovery is used to determine the topology of the entire network and calculate routing tables. OLSR uses four message types: Hello message, Topology Control (TC) message, Multiple Interface Declaration (MID) message, and Host and Network Association (HNA) message. Hello messages are used for neighbor sensing. Topology declarations are based on TC messages. MID messages contain multiple interface addresses and perform the task of multiple interface declarations. Since hosts that have multiple interfaces connected with different subnets, HNA messages are used to declare host and associated network information. Extensions of message types may include power saving mode, multicast mode, etc. [53] The example shown in Figure 2.11 presents the basic idea of the OLSR protocol.

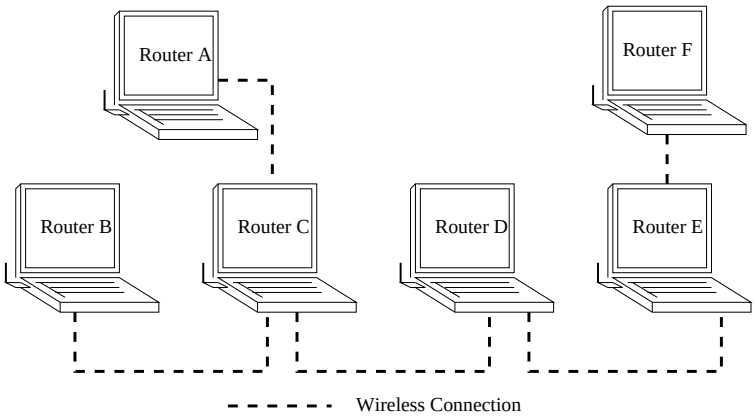


Figure 2.11: Example of OLSR routing protocol.

First, we discuss some concepts used in OLSR, namely one-hop neighbor set, two-hop neighbor set, MPR set, and MPR selector (MPRS). The one-hop neighbor set is formed by all adjacent routers. For example, Router *C* forms the one-hop neighbor set of Router

A. A two-hop neighbor set is the set of routers that are two hops away. Routers *B* and *D* form the two-hop neighbor set for Router *A*. The MPR set of a router is a subset of neighboring routers that are responsible for forwarding control messages sent by that router. The MPR set should be able to cover all the two-hop neighbors of that router. For example, Router *D* is a neighboring node to Router *C*. It covers Router *C*'s two-hop neighbor, Router *E*. Therefore, {Router *D*} is the MPR set of Router *C*. Since the MPR set of a router is responsible for rebroadcasting messages sent by that router, the routing protocol is closer to optimal with a smaller MPR set. Qayyum, et al. [54] give a simple algorithm to select MPRs, together with an example. The MPR selector (MPRS) set of one router is formed by routers that select this router as one of their MPR routers.

Table 2.2 shows the MPR set and MPR selector set for each router in Figure 2.11. Consider Router *D* as an example. Its neighbor set is $\{C, E\}$ and the two-hop router set is $\{A, B, F\}$. The MPR set of Router *D* can cover the entire set of two-hop routers only if it includes Router *C* and Router *E*. Similarly, Router *C* and Router *E* select *D* as one of their MPR routers, so the MPR selector set of *D* is $\{C, E\}$.

Table 2.2: MPR Sets and Selectors in the OLSR Example

Router ID	MPR Set	MPR Selector(s)
A	C	NULL
B	C	NULL
C	D	A, B, D
D	C, E	C, E
E	D	D F
F	E	NULL

OLSR routers periodically broadcast hello packets to one-hop neighbors. Each router builds a list of neighbors and a list of two-hop neighbors based on received hello messages. Each router also creates one MPR set and one MPR selector set. Routers that have non-empty MPRS lists broadcast their MPRS sets to neighbors via TC packets.

Therefore, the size of control messages is reduced compared with broadcasting a list of all neighbor routers. A router rebroadcasts received packets if and only if the sender of that packet is in its MPR selector set. This helps to reduce the frequency of flooding. Routers build routing tables based on received TC packets. For example, assume Router *A* wants to set up a routing entry with destination Router *F*. It searches for MPR routers of *F* in the received TC packets, which is Router *E* in our example. Since *E* is not in any known route, MPR routers of *E* are also searched. Note that a route found by OLSR is always formed by hops from MPR selectors to corresponding MPR routers. Thus, Router *A* can eventually figure out the next router to *F* is Router *C*. Routing entries are reset once paths become invalid due to a link failure. This is similar to other link state routing protocols.

2.3.6 Topology Broadcast Based on Reverse-Path Forwarding (TBRPF)

The Topology Broadcast Based on Reverse-Path Forwarding (TBRPF) protocol is another link state, proactive routing protocol for MANETs [48]. Each router running TBRPF computes a source tree to all reachable destinations based on partial topology information stored locally. The source tree is also known as the shortest path tree. To reduce overhead, routers in TBRPF only broadcast part of their source tree to neighbors. The partial source tree is called the reportable tree. The main idea of sharing reportable trees with neighbors comes from the Partial Tree-Sharing Protocol (PTSP) described in [47]. Basically, in the local copy of network topology, a link cost is equal to the actual value if this link is in the shortest path tree. Otherwise, the cost is equal to or greater than the real value. The procedure to generate a reportable tree at a router is as follows. Links that are in this router's shortest path tree are checked. If such a link is estimated to be in the neighbors' shortest path trees, it is added to the reportable tree. Note that the estimated results may not be correct, but they do include the correct link costs. TBRPF is said to work better in dense networks [48].

TBRPF has two modules, the neighbor discovery (TND) module and the topology

discovery and route computation module. "The key feature of TND is that it uses 'differential' HELLO messages which report only changes in the status of neighbors" [48]. This reduces the size of HELLO packets used in this module. The HELLO packet in TBRPF may contain three lists of router IDs. They are in three different formats: Neighbor Request, Neighbor Reply, and Neighbor Lost. The neighbor request list includes the IDs of new neighbors whose HELLO messages are heard for the first time. This implies that the links to those neighbors are currently one-way links. Note that the neighbor request list is always included in HELLO packets, even if it is empty. The other two lists may not be included if they are empty. HELLO packets are sequenced by senders.

The TND module is responsible for discovering any new neighbors and detecting the loss of any neighbors. After continuously hearing HELLO packets for a certain number of times, usually twice, a router responds by sending HELLO packets with a neighbor request set in each of its next NBR_HOLD_COUNT (typically three) HELLO messages, or until a neighbor reply message is received from the new neighbor. This avoids short lived links. When a router receives a neighbor's reply message, it declares a bidirectional link by sending NBR_HOLD_COUNT HELLO messages, including the received neighbor reply message. In the case of missing HELLO packets for NBR_HOLD_COUNT times, a router sends NBR_HOLD_COUNT neighbor lost messages to that neighbor. A neighbor is declared lost if no reply comes from that neighbor. The topology discovery and route computation module allows routers to build source trees and report parts of those trees to their neighbors. We use the example shown in Figure 2.12 to illustrate TBRPF.

The network has six routers as shown in the figure. We concentrate on Router *F* and Router *E*. The source tree of Router *F* does not include the link between *B* and *C* since link *BC* is not in the shortest paths to any destinations starting from Router *F*. Because *F* estimates that its neighboring Router *E* does not use any of its link(s) to form the shortest path tree with root at *E*, no link is added to the reportable tree except the link between *E* and *F*. Therefore, the size of broadcast messages generated at *F* is much smaller than

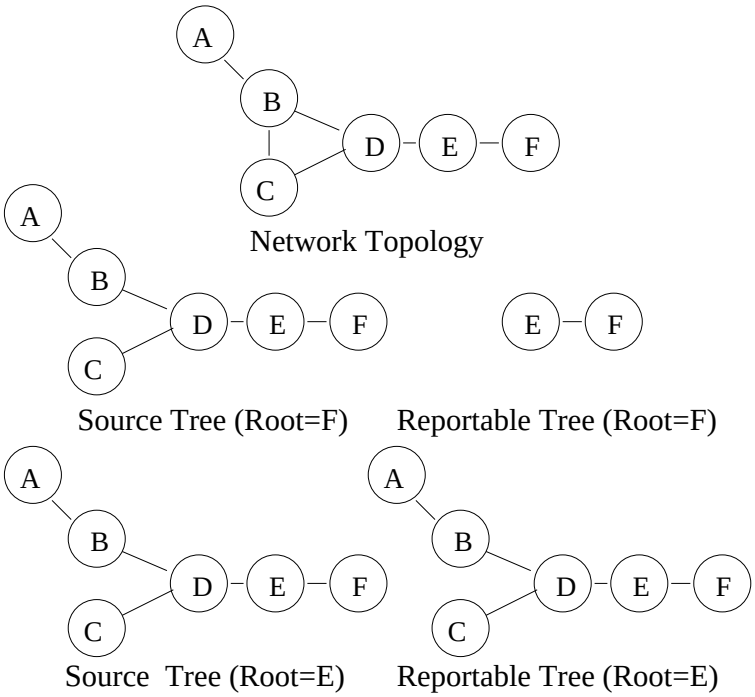


Figure 2.12: Example of TBRPF routing protocol.

a message that contains all links. For Router *E*, its source tree is the same as that of *F*. However, all links in the source tree may be used by its neighbors to form corresponding source trees. Thus, *E*'s reportable tree is equal to its source tree. Note that routers do not broadcast all known links. Therefore, overhead may be reduced by TBRPF. It is said that in a more dense network, fewer links are reportable [48].

Reportable trees are built in all routers in the network guided by the TBRPF algorithm. When reportable trees are broadcast, necessary link information is propagated to all routers in the network. If a router detects a link failure or a new link, it notifies its neighbors whose source tree may be affected by the changed link. As mentioned before, although the estimated result does not always match the actual result, the estimated results from neighbors contain the necessary link information. Thus, TBRPF can guarantee that correct shortest path trees at all routers are built by redundant estimations.

2.3.7 Landmark Routing Protocol (LANMAR) and Fisheye State Routing Protocol (FSR)

The Landmark Routing (LANMAR) protocol is a proactive protocol for large-scale ad hoc networks. The protocol uses "the notion of logical subnets in which the members have a commonality of interest and are likely to move as a 'group"' [55]. This protocol requires that the router that is elected as the landmark in a subnet has knowledge of all the members in its group. This idea is similar to the concept of designated router in a broadcast network with the OSPF protocol. Each router maintains a distance vector to landmarks in every subnet. LANMAR elects landmarks by the weight of routers, such as the degree in one subnet. In the case of a tie, the router with the lowest ID is selected. This protocol relies on a proactive routing protocol within each subnet. This underlying protocol provides "scoped" (maximum number of hops) routing in each subnet. It works together with the LANMAR protocol and provides information about other routers in the "scoped" area before and during the selection of a landmark. The Fisheye State Routing (FSR) protocol [56] can be used as the underlying protocol for LANMAR. FSR uses different update frequencies for different destinations based on the distance to that destination. For destinations within the same subnet, a router uses the underlying protocol to update the routing information periodically. For routers outside the "scoped" area, only landmarks update the routing information with a non-zero frequency. Other routers use the route toward the landmark of their subnet as the default route to a remote host or group if they do not have a direct connection to the remote destination. Note that the underlying protocol can also be another proactive routing protocol, such as the OLSR protocol described in Section 2.3.5. The use of landmarks is said to reduce overhead [55]. Routes between subnets are not optimal because the landmark in one subnet is not guaranteed to be in the shortest path from one router in this subnet to a destination outside of this subnet. Figure 2.13 presents an example of LANMAR and FSR.

In Subnet I in this example, Routers 1, 2, 3, 4, and 5 have degrees of 2, 2, 1, 4, and 1,

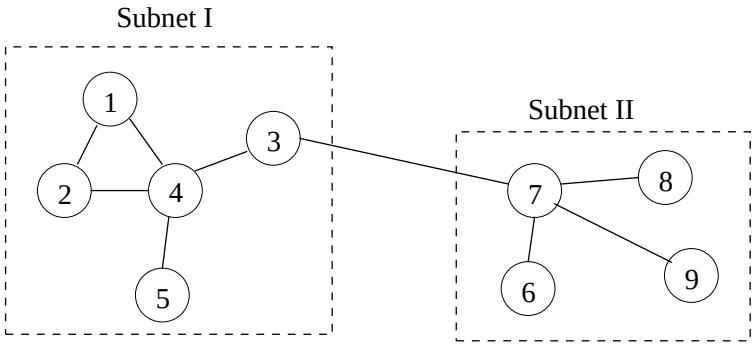


Figure 2.13: Example of LANMAR and FSR routing protocols.

respectively. Note that although Node 3 has degree of 2, only one of its links is in Subnet *I*. So Router 4 is elected as the landmark of Subnet I. Similarly, Router 7 is selected as the landmark of Subnet II. Routers in Subnet I know routes to any other node in the same subnet because of the underlying routing protocol, such as FSR. For the route to a remote router, say a router in Subnet II, Routers 1, 2, and 5 use the landmark, Router 4, as the next hop. Router 3 has a direct connection to Subnet II, so it is a little different. Thus, Router 4 is responsible for directing all routes to routers in Subnet II via Routers 3 and 7.

2.3.8 Zone Routing Protocol (ZRP)

The Zone Routing Protocol (ZRP) is a prototype routing protocol. ZRP is formed by two sub-protocols, the Intrazone Routing Protocol (IARP) and the Interzone Routing Protocol (IERP). IARP is "a limited scope proactive routing protocol used to improve the performance of existing globally reactive routing protocols" [57]. It relies on the service of a certain neighbor discovery protocol (NDP) to provide neighbor information. IARP may use a scheme based on the time-to-live (TTL) field in IP packets to control the zone range. (When a broadcast packet passes a router, the value of TTL is decremented by one before it is rebroadcast, and when TTL equals to zero, the packet is not rebroadcast.) IERP is the reactive routing component of ZRP [58]. This scheme is responsible for finding a global path. It avoids global queries for destinations that would be sent to surrounding

r-hop neighbors. When global queries are required, "the routing zone based broadcast service can be used to efficiently guide route queries outward, rather than blindly relaying queries from neighbor to neighbor" [58]. ZRP tries to combine the advantages of reactive and proactive routing protocols. The potential disadvantage is the lack of route optimization.

We use the example network in Figure 2.13 to briefly show the concept of ZRP. The range of the zone is set to one. So routers in Subnets *I* and *II* use proactive IARP to find routes to other routers in the same subnet. For routes to the other subnet, reactive IERP is used.

2.3.9 Mobile IP

In a traditional wired network, hosts or routers should attach to the networks to which their network interface(s) belong. If they change their attachment, they should also change their IP address or host-specific routes must be propagated throughout the network. Mobile IP [11] allows IP nodes (hosts and routers) to seamlessly roam among IP subnetworks without addressing constraints. This brings a certain freedom for IP nodes, which is the very motivation to use Mobile IP [11].

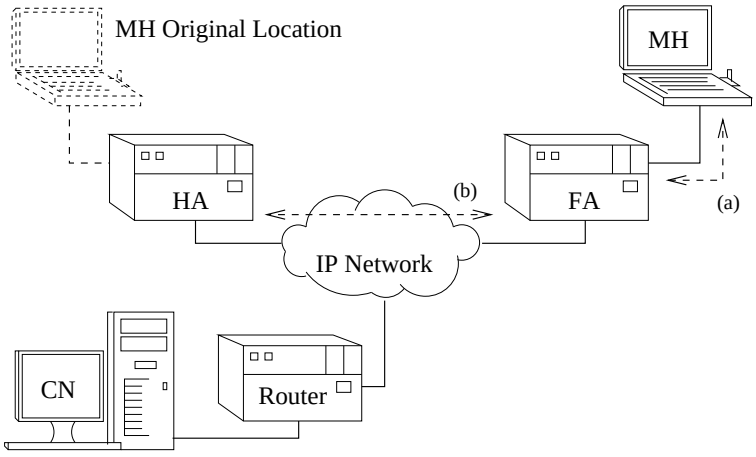
Mobile IP introduces the concepts of a home agent (HA) and a foreign agent (FA). A mobile node (MN) is the roaming host (or router). A HA is a router on a MN's home network that can deliver packets to the MN when it is away from its home network by tunnelling packets. The HA also maintains information about the current location of the MN. A FA is a router on the foreign network that the MN is currently visiting. After a MN registers at the FA, the FA provides routing services to the MN. The FA de-tunnels packets that were tunnelled by the MN's HA and forwards them to the MN. The FA also serves as a default router for a registered MN [11].

Figure 2.14 gives an example of Mobile IP operation. The MN attaches to its HA.

Now the MN moves to a new location, attaching to a FA. The MN registers at the FA and gets a care-of address from the FA. This address is used by the FA to identify this MN. This procedure is shown as procedure (a) in Figure 2.14. After that, the MN registers the care-of address at the FA and the HA, which is procedure (b) in Figure 2.14. We can assume that these procedures use authentication to prevent potential attacks from "bad" nodes. We do not discuss this in detail here.

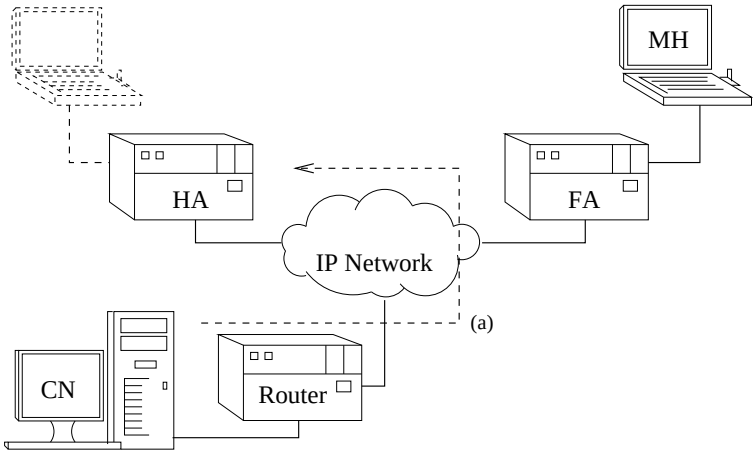
Figures 2.15 and 2.16 show the case when another host, a corresponding node (CN), wants to communicate with the MN. In this figure, we assume that CN is in a third network and sends a packet to the MN. Because of the prefix of the destination address, this packet travels through the IP network to the MN's home network. Once the HA receives this packet, as shown in Figure 2.16, it sends a tunnelled packet to the FA. The tunnelled packet contains the original packet for the MN. The FA de-tunnels this packet and directs the original packet to the MN. These two steps are procedures (a) and (b) in Figure 2.16. If route optimization is used, the HA also sends an updated packet specifying the new location of that MN to the sender, CN. The CN can then send future packets directly to the MN's new location.

The example we presented is for a MN. Similarly, if hosts in a small network are relatively static and this network has a Mobile IP enabled router, this stub network can also roam by using Mobile IP. This implies that if a MANET is formed by some of these stub networks and MNs, Mobile IP is a possible approach to support routing, with or without other routing protocols. Note that Mobile IP does solve the addressing problem to permit changing attached points for mobile hosts or subnets. However, the backbone network, which is usually static, requires a routing protocol to support traditional routing. Therefore, applying Mobile IP to MANET requires additional investigation of the interworking of Mobile IP and the backbone routing protocol in a MANET.



(a) Register at FA, get care−of address
(b) Register FA and care−of address at HA

Figure 2.14: Example of Mobile IP (Part 1).



(a) CN sends packet to MH based on MH's home IP address

Figure 2.15: Example of Mobile IP (Part 2).

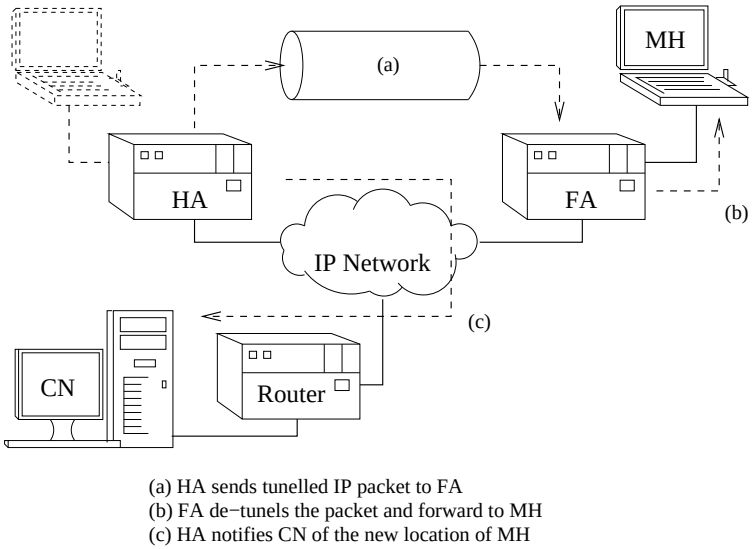


Figure 2.16: Example of Mobile IP (Part 3).

2.4 Summary and Problem Statement

We briefly introduced basic concepts in wireless networks in previous sections, including network architectures, wireless channels, MAC layer protocols, and network layer routing protocols. We presented proposed routing protocols for MANETs. We also discussed protocols and techniques for wired networks, such as OSPF and Mobile IP, which could be potentially used for MANET routing.

The routing protocol is one of the fundamental protocols in MANETs. Standard routing protocols for MANETs have not yet been defined. Currently, there are four leading routing protocols, AODV, DSR, OLSR, and TBRPF, as determined by the IETF MANET group [2]. Before our research, most prior work focused on simulation-based comparisons among different MANET routing protocols. However, due to a lack of proper characterization of different MANET protocols, these simulation experiments are not well designed. For example, the simulation results from different research groups cannot be directly compared [59]. There are no clear conclusions that can be drawn from this prior work. In other words, the relationship between the simulation conditions and MANET

routing protocols remains unclear. Therefore, the conclusions based on the simulation experiments cannot be generalized and new methodologies to study MANET routing protocols are clearly needed.

Based on this deficiency within the MANET research community, my primary research goal is to provide a new methodology to aid the analysis and evaluation of MANET routing protocols. The following chapters summarize my research and contributions toward this goal, including the characterization of different routing protocols using a novel framework, design of a new routing protocol for MANETs, a study of node mobility including a quantitative study of link lifetime in a MANET, an adaptive interval scheme based on neighbor stability, improvement of a simulator and corresponding protocol models, design and development of a novel emulation test bed, evaluation of MANET routing protocols through simulation experiments, and emulations of different MANET protocols and applications based on an integration testbed.

Chapter 3

Relay Node Set Framework for MANET Routing Protocols

In this chapter, we propose a framework for MANET routing protocols that can characterize relationships among different MANET routing protocols. Aided with this framework, we identify the parameters that can affect the operations and, thus, performance of MANET routing protocols. The parameters are used to guide the simulation study presented in Chapter 7. Work addressed in this chapter is reported in [1].

3.1 Requirements of a Framework

As we mentioned in Section 2.3, a MANET routing protocol is usually classified as a pure proactive protocol, a pure reactive protocol, or a hybrid of the two. A proactive routing protocol periodically maintains routes to all possible destinations, while a reactive protocol builds a route on demand when there is no known route. Past study of MANET routing protocols focused on designing new protocols, comparing existing protocols, or improving protocols before standard MANET routing protocols are established. Many

researchers have studied these protocols using simulations of arbitrary networks with certain traffic profiles [42, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69]. However, due to the lack of consistent characterization of different MANET protocols, prior simulation experiments are not well designed. For example, convincing conclusions cannot be drawn from some studies because of some flaws in the simulator and simulation models that are used. Some protocols that perform well, in terms of control overhead or throughput, in some scenarios may have poor performance under other conditions. Moreover, the relationship between the simulation conditions and the MANET routing protocols remains unclear. Therefore, the conclusions based on these simulations cannot be generalized.

These efforts can be aided by a framework that can characterize MANET routing protocols. However, previously there was lack of research on this topic. In the next section, we present such a framework using the concept of a *relay node set* (RNS). The framework describes routing protocols using four modules. The framework provides a different view of MANET protocols and highlights relations among different protocols. The framework provides the following capabilities.

- We can describe MANET routing protocols with the RNS framework so that researchers can understand the protocols more easily. This framework characterizes different MANET routing protocols and highlights the internal relationships among different protocols.
- Quantitative models based on the RNS framework can be used to identify factors that affect control overhead for different MANET routing protocols. The framework allows comparison of routing protocols by analytical models coupled with network parameters and traffic profiles. These parameters and profiles could come from simulations or measurements. Therefore, this quantitative model can help researchers estimate how control overhead changes when the network environment changes and, thus, aid in selection of proper protocols for specific application environments.
- Possible ideas for improving proposed MANET routing protocols can be found us-

ing the RNS framework. For example, reduction of factors that affect the overhead of a protocol can be used to improve the performance of that protocol in terms of control overhead. The modularized framework allows a subprotocol to be replaced by another subprotocol to form a better MANET routing protocol as long as these two subprotocols have the same functionality.

- The RNS framework and the corresponding quantitative model can aid the design, evaluation, and validation of new MANET routing protocols with emphasis on control overhead. For example, we can expect some efficient algorithms to achieve low overhead for particular application environments. Based on the factors that affect control overhead, we can also evaluate and improve MANET simulation models (Refer to Sections 5.1 and 5.2 for more details).

- A better understanding of control overhead based on the RNS framework can be useful for routing protocols that incorporate adaptive controls, such as SHARP protocol [70]. These adaptive routing protocols try to achieve different application specific goals, e.g., minimal packet overhead, targeted packet loss rate, or limited delay jitter.

In this chapter, we concentrate on the explanation of the framework and provide examples of its capabilities. Section 3.2 presents the RNS framework in detail, together with a prototype of an analytical model with an emphasis on control overhead. Using the framework, we give descriptions of some MANET routing protocols as examples in Section 3.3. Analysis and comparison are also illustrated in this section. Section 3.5 addresses the factors that affect the operations and performance of different MANET protocols, which provides guidance for the simulation study in the following chapters. Section 3.6 gives a summary and describes potential future work based on this framework.

3.2 The Relay Node Set Framework

We define two terms before we present the framework, *cover* and *relay node set*. When a node broadcasts, all of its neighbors should be able to receive that message. (Here, we assume that the MAC layer protocol can guarantee delivery.) Referring to this property, we say that the sender covers all of its neighbors. A set of nodes, say set M, is covered by another set of nodes, say set N, when any node in M is covered by at least one node in N. Figure 3.1 shows an example. Node *2* covers nodes *1*, *3*, and *5*. The set of black nodes covers the set of white nodes.

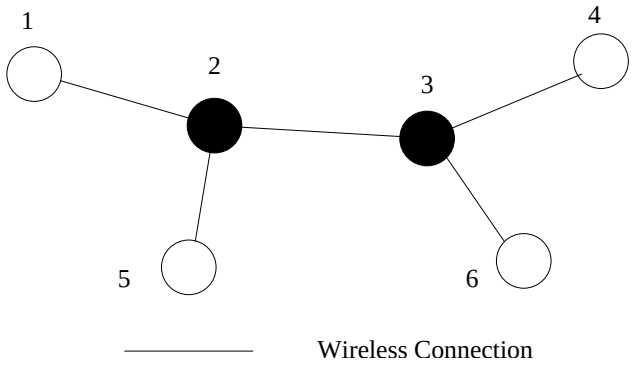


Figure 3.1: Example of relay node set and the definition of cover.

We make a general assumption that all nodes in a MANET network act as routers and are willing to forward data packets from their neighbors. The relay node set is a set of nodes that have the capability to retransmit control messages in the MANET network. Nodes that are not in a RNS, called non-RNS nodes, always stay silent after they have processed control messages received from their neighbors. Figure 3.1 also illustrates the role of an RNS. For example, assume that the control message is a neighbor list for each node. Every node broadcasts to nodes within its radio range. Only the two black nodes, nodes *2* and *3*, are needed to retransmit the first received control messages from their neighbors so that all nodes can receives every other node's neighbor list. Thus, all nodes can build routing tables according to the topology indicated by neighbor lists. The black node set in this example is a RNS.

The framework presented in this section is built using the concept of the RNS. It can describe a broad class of MANET routing protocols. As shown in Figure 3.2, this framework contains four modules, RNS building, RNS control message (unicast or broadcast) propagation, RNS maintenance used when the topology changes, and unreliable (control message) transmission handling. Note that in this research, we only consider the basic routing function for these protocols and not other issues, such as quality of service and security. The arrows in Figure 3.2 indicate dependency relationships between pairs of modules. For example, the arrow from the module that handles unreliable transmissions to the module that handles the building of the RNS implies that the latter relies on the service of or information from the former.

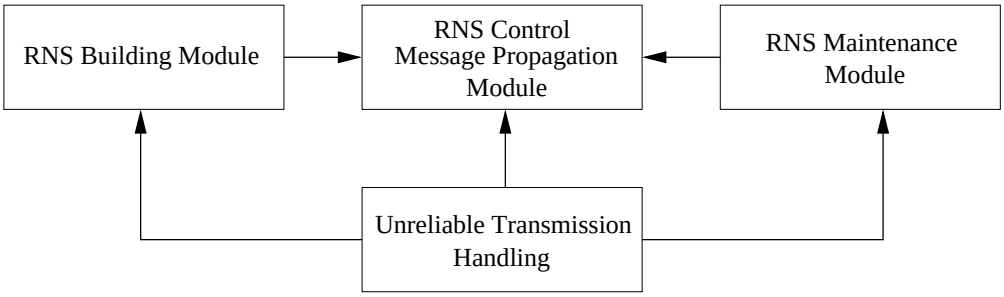


Figure 3.2: Four modules in the RNS framework.

Different schemes used in these modules result in different MANET routing protocols, including both reactive and proactive routing protocols. In the RNS building module, different routing protocols use different methods to build the RNSs and the result could be one RNS for the entire connected network or multiple sets of RNSs for the network at a given time. RNSs can be associated with a certain node, a certain link, or a certain pair of nodes when multiple sets of RNSs are allowed to exist in a network at a given time. A RNS may or may not cover all nodes in the network, depending on the algorithm that builds it. For example, Dynamic Source Routing (DSR) [49] builds an RNS when an initiator starts the route request procedure by broadcasting a route request message. This RNS is associated with a pair of source and destination nodes and may not cover all nodes in the network. This is a typical feature of reactive routing protocols.

RNSs built by proactive routing protocols usually cover the entire connected network. Routes can be built based on the control messages propagated by nodes in the RNS. For example, in DSR, a route request reply message is sent back to the initiator by unicast. Thus, all intermediate nodes and the initiator can build the routing entry for that particular destination.

Control messages sent in an RNS are different for different routing protocols. RNS nodes may unicast or broadcast control messages. A control message can be a list of visited nodes for the control message, a route request reply from a destination to the corresponding initiator, default gateway information for each node, or a link state record. These control messages are used to build routing entries. Generally speaking, the quality of the routing information is proportional to the size of the control message. And, the more information we want to exchange, the more overhead we have.

In general, reactive routing protocols do not have the capability to continuously maintain RNSs. Once an old path is broken due to the movement of one or more intermediate nodes, the RNS building procedure starts again and new routing paths generated by the new RNSs replace the old ones. Proactive routing protocols usually have some scheme to maintain RNSs when an old one becomes invalid. Examples of this are given in the following section. Therefore, the frequency of building RNSs in reactive routing protocols is usually higher than in proactive protocols although this depends on the frequency of starting new traffic flows for a reactive protocol. We can assume that the total rate of RNS building and maintenance functions in a certain network environment for any protocol is a constant when the traffic load in the network is heavy enough. This is because when the topology changes, protocols either maintain old RNSs or re-build new RNSs.

In a wireless network, a transmission may fail due to the poor quality of the wireless channel. Control messages can be damaged or lost due to transmission errors or buffer overflows. Therefore, a routing protocol usually should have a separate module to

handle unreliable transmissions. Reactive protocols, such as DSR and AODV, may re-try for a few times if a transmission failure is detected; while proactive protocols, such as OLSR and TBRPF, periodically rebroadcast control messages and, thus, handle unreliable transmissions.

Our RNS framework can describe a broad class of routing protocols for MANETs and can illustrate some internal similarities among protocols. We are also able to analyze protocols with this framework. For example, according to the four-module RNS framework, the total control overhead for a MANET routing protocol is formed by four overhead components: (i) the overhead to build or rebuild the RNS, (ii) the overhead to maintain the RNS, (iii) the overhead to propagate control messages in the RNS, and (iv) the overhead to handle unreliable transmissions. Therefore, we can develop an analytical model for overhead. Equation 3.1 represents the total control overhead for a MANET routing protocol based on this framework, expressed as the number of bytes transmitted during time period t.

$$
\begin{aligned}
&Overhead = O_{building} + O_{maintenance} + O_{propagation} + O_{retransmission} \\
&\begin{cases}
O_{building} = \sum N_{building} \sum S_{RNS} \sum P_{control\_building} \\
O_{maintenance} = \sum N_{maintenance} \sum S_{maintenance} \sum P_{control\_maintenance} \\
O_{propagation} = \sum N_{propagation} \sum S_{propagation} \sum P_{control\_propagation} \\
O_{retransmission} = \sum N_{retransmission} \sum S_{retransmission} \sum P_{control\_retransmission}
\end{cases}
\end{aligned}
\tag{3.1}
$$

Here O stands for the overhead and the subscripts indicate the corresponding modules to which these variables belong, P represents the size of corresponding control packets, N refers to the number of corresponding procedures executed in time period t, S is the size of the corresponding RNS, and the subscripts indicate the corresponding modules associated with the variables. In Equation 3.1, the overhead for the RNS building module and the RNS control message propagation module can be considered as the

static overhead of a routing protocol. The overhead for the maintenance module is due to the dynamic topology of a network. The overhead for the retransmission module provides robustness for a routing protocol. Generally speaking, if one of the terms becomes smaller, some of the others may need to become larger. Proactive routing protocols tend to have large overheads in the RNS building module and the RNS control message propagation module when compared to reactive protocols. Proactive protocols tend to have small overhead to maintain the RNS so that the number of re-building operations is small. Protocol designers must balance the overhead associated with these modules for a given environment so that routing protocols have optimal overall overhead for that environment. Descriptions of routing protocols with this analytical model can be used to identify overhead factors. Comparison of results among protocols based on this framework can provide directions to improve existing MANET routing protocols. Designers and decision makers can solve optimization problems among these four modules, according to their particular application environment, when they design or choose MANET routing protocols. In the next section, we show simple examples of analysis and comparison of typical routing protocols using the framework.

3.3 Description and Analysis of Routing Protocols using the RNS Framework

In this section, we use the RNS framework to describe MANET routing protocols and give an analytical model for each protocol with the emphasis on control overhead. Note that in this section we assume that the underlying MAC layer protocol supports guaranteed transmission. Therefore, the overhead in the unreliable transmissions handling module is relatively small for these routing protocols.

3.3.1 Analysis of DSR with the RNS Framework

DSR is a typical on-demand reactive routing protocol. (Please refer to Section 2.3.2.) A route request is broadcast when there is no known route to the expected destination. A node that is not the destination rebroadcasts the route request if it does not know the route. All nodes that forward a certain route request form an RNS. A RNS in DSR is always associated with the pair of nodes consisting of the request initiator and the destination node. This procedure is the RNS building module in DSR, which stops when a route is found. A RNS in DSR may not include all nodes in the network, unless the destination node is not in the connected network (and this fact remains unknown) or the route is not discovered until the last node in the network joins the RNS. The number of RNSs built in a network using DSR is less than or equal to the number of the pairs of source and destination nodes defined by the corresponding traffic profile. In DSR, only route request reply or route failure notification is sent back to the initiator in the corresponding RNS using unicast. This is the RNS control message propagation module. DSR does not have any scheme to maintain RNSs. Therefore, a route request is regenerated when a known route fails. This implies that DSR rebuilds corresponding RNSs for some pair of source and destination nodes if old ones fail. In DSR, to avoid the case that a control packet is lost (e.g., due to a buffer overflow at the receiver or a packet collision) or corrupted (e.g., due to a transmission channel error), acknowledgments or a certain number of request retries may be used when MAC layer protocols cannot provide guaranteed deliveries. This is the module to handle unreliable transmissions. We ignore this module since we assume here that control messages are guaranteed to be delivered by the MAC layer protocol. DSR does not have a RNS maintenance module, therefore it suffers large overhead to build or rebuild RNS. According to our analysis, the total overhead for DSR is given by Equation 3.2.

$$Overhead = \sum_{i=1}^{N_{pair}} \left[\sum_{j=1}^{N_{i,build}} (\overline{P_{i,j,req}} \times S_{i,j,RNS} + \overline{P_{i,j,rep}} \times S_{i,j,ret\_p} + \overline{P_{i,j,err}} \times S_{i,j,RNS}) \right] \quad (3.2)$$

The definitions of variables are listed as below.

- N_{pair} is the number of pairs of source and destination nodes.
- $N_{i,build}$ is the total number of building or rebuilding operations for the i-th RNS.
- $\overline{P_{i,j,req}}$ is the average size of the route request packet when the i-th RNS is rebuilt for the j-th time.
- $S_{i,j,RNS}$ is the size of the i-th RNS when this RNS is rebuilt for the j-th time. Note that $S_{i,j,RNS}$ can be zero when the route between the corresponding source and destination nodes is known, for example, due to the route cache recording other routes that contain the desired route.
- $\overline{P_{i,j,rep}}$ is the average size of the route request reply message and corresponding route failure notification, if there is any, for the i-th RNS when this RNS is built for the j-th time.
- S_{i,j,ret_p} is the size of the subset of the i-th RNS that includes all nodes on the path from the node that discovered that path back to the initiator and part of nodes adjacent to that path when the i-th RNS is built for the j-th time. Note that $S_{i,j,ret\_p} \leq S_{i,j,RNS}$.
- $\overline{P_{i,j,err}}$ is the average size of the error message when the i-th RNS is rebuilt for the j-th time.

Note that some of these variables are dependent on others. When the network size increases, $\overline{P_{i,j,req}}$, $\overline{P_{i,j,rep}}$, $\overline{P_{i,j,err}}$, $S_{i,j,RNS}$ and S_{i,j,ret_p} may all increase. When nodes have higher mobility, $N_{i,build}$ could become larger. When the traffic load becomes heavier, N_{pair} becomes larger. Therefore, DSR suffers high overhead in a network with high mobility nodes and heavy traffic loads. DSR is also not suitable for large networks. These analysis results match the simulation results presented in [68].

3.3.2 Analysis of AODV with the RNS Framework

AODV is similar to DSR. (Refer to Section 2.3.3.) The difference is the route request packet for RNS building in AODV contains the number of hops, while the one in DSR contains the list of intermediate node IDs. Therefore, the RNS model for AODV is similar to that of DSR. The overhead for AODV is similar to that in DSR. Analytical results for DSR can also apply to AODV. Aron and Gupta also present similar simulation results for AODV [68].

3.3.3 Analysis of TORA with the RNS Framework

TORA uses blind broadcast similar to DSR to build RNSs in both reactive and proactive procedures. Refer to Section 2.3.4. A RNS built in the proactive mode in TORA covers all nodes and is associated with a certain destination node. Route request replies are broadcast to all nodes in the network. A RNS built in the reactive mode usually covers part of the entire network and is associated with a certain pair of source and destination nodes. Route request replies are sent back to the initiator using unicast. The difference between TORA and reactive routing protocols, such as DSR and AODV, is that TORA has its own maintenance scheme to assign and reassign node heights (and, thus, link directions) to maintain RNSs when a route fails. Unlike some other routing protocol specifications, the TORA draft does not address the function handling unreliable transmissions. The draft assumes that TORA works on top of lower layer mechanisms or protocols that provide reliable and in-order control packet delivery between neighbors. The overall control overhead for TORA expressed as the number of bytes for duration t, is shown in Equation 3.3. Note that there are two modes used in TORA, reactive and proactive. Therefore, the format of control overhead expression in TORA is different from DSR and AODV discussed in previous sections.

$$
\begin{aligned}
&Overhead = O_r + O_p + O_m \\
&\begin{cases}
O_r = \sum_{i=1}^{N_{pair}} \sum_{j=1}^{N_{i,build\_r}} (P_{QRY} \times S_{i,j,RNS} + P_{UPD} \times S_{i,j,return\_path}) \\
O_p = \sum_{i=1}^{N_{p\_dest.}} \sum_{j=1}^{N_{i,build\_p}} (P_{OPT} \times N) \\
O_m = \sum_{i=1}^{N_{pair}+N_{p\_dest.}} \sum_{j=1}^{N_{i,maintain}} (P_{UPD} \times S_{i,j,update} + P_{CLR} \times S_{i,j,erase})
\end{cases}
\end{aligned}
\tag{3.3}
$$

In Equation 3.3, O_r is the overhead to generate and rebuild RNSs in the reactive mode, O_p is the overhead to generate and rebuild RNSs in the proactive mode, and O_m is the overhead to maintain RNSs. Some variables are the same as those in Equation 3.2. Others are defined as follows.

- N_{pair} is the number of pairs of source and destination nodes for which paths are found in reactive mode.
- $N_{i,build\_r}$ is the total number of building or rebuilding operations for the i-th RNS in reactive mode.
- P_{QRY} is the size of the reactive route query (QRY). It is a constant in TORA.
- P_{UPD} is the size of the request reply (UPD) message. It is also a constant in TORA.
- $N_{p\_dest.}$ is the average number of destination nodes that start the proactive procedure.
- $N_{i,build\_p}$ is total the number of building or rebuilding operations for the i-th RNS in proactive mode.
- P_{OPT} is the size of the route building (OPT) packet in the proactive procedure. It is a constant in TORA.
- N is the total number of nodes in the network.

- $N_{i,maintain}$ is the number of times that the maintenance procedure is used for the i-th request.
- $N_{i,j,update}$ is the number of nodes that need to adjust their heights when the topology changes for the i-th request, including in both proactive and reactive modes, in the j-th maintenance operation for the i-th RNS.
- P_{CLR} is the size of route erasing (CLR) packets. It is a constant in TORA.
- $S_{i,j,erase}$ is the number of nodes that need to propagate the CLR packet during the route erasing procedure for the i-th request, including both proactive and reactive modes, in the j-th maintenance operation for the i-th RNS.

The overhead in TORA can be reduced if we can find a good algorithm to assign heights to nodes so that we can avoid height reassignments for a large number of nodes or frequent re-construction of the entire RNS when the topology changes. The proactive procedure requires more overhead than the reactive protocol in TORA. Similar to other reactive routing protocols, the performance of TORA in terms of overhead depends on the properties of the network and traffic. TORA also has high overhead for large networks with high mobility and heavy traffic.

3.3.4 Analysis of OLSR with the RNS Framework

OLSR (see Section 2.3.5) is a typical proactive routing protocol. OLSR uses periodic "hello" messages to exchange neighbor lists between neighboring nodes. By using the neighbor list information, multiple relay (MPR) node sets are built for all nodes with a simple algorithm. An MPR node set is a small subset of neighboring nodes that covers all of the center node's two-hop neighbors and may rebroadcast any control message generated or forwarded by that center node. Information about MPR sets is also sent to neighbors via the "hello" messages. All nodes generate their own MPR selector (MPRS)

sets. The MPRS set for a node is the set of neighboring nodes that select this node as a member of their MPR sets. Only nodes with non-empty MPRS sets broadcast control messages containing their MPRS sets. Generally, a node re-broadcasts a first-received control message sent by its neighbor if and only if this neighbor selects it as one of the neighbor's MPR nodes. Note that in the propagation procedure, if a node i is in the MPR node set for another node, say node j, and node i already received the broadcast control message originated by a certain initiator from a third node, say node k, before it receives this message from node j, node i keeps silent. In other words, node i is not included in the RNS, which reflects the first-seen rule for OLSR. The procedure described above is the RNS building procedure in OLSR. Therefore, the RNSs in OLSR are associated with certain source nodes and multiple copies of RNSs associated with different initiators can co-exist at any given time.

The sizes of different RNS sets may not be the same. The reason that these RNS sets may have different sizes is due to the distributed selection algorithm of MPR nodes, which is performed by neighbor nodes and may not be consistent among neighboring nodes. Each node rebroadcasts control messages sent by nodes that are in its MPRS set which have not been seen before. Therefore, information about all MPRS sets can be propagated to all nodes in the network with a small number of retransmissions. Routes can be built based on the information about MPRS sets of other nodes. MPRS nodes act as gateways and pairs of MPR and MPRS nodes form routes. This is the RNS propagation procedure for OLSR. When the topology changes, the OLSR protocol uses local modifications of MPR and MPRS sets to maintain RNSs at nodes within a two-hop range of the changed link. This is the RNS maintenance procedure for OLSR. OLSR periodically broadcasts control messages. Therefore, it has the capability to handle unreliable transmissions.

The total overhead in OLSR is shown in Equation 3.4. Note that since OLSR periodically broadcasts control messages, we can assume that the overhead to handle unreliable

transmissions is included in the overhead for propagation.

$$
\begin{aligned}
&Overhead = O_{construction} + O_{maintenance} + O_{propagation} \\
&\begin{cases}
O_{construction} = \sum_{i=1}^{N_{hello}} \sum_{j=1}^{N} P_{i,j,hello} \\
O_{maintenance} = \sum_{i=1}^{N_m} \sum_{j=1}^{N_{i,adjust}} (P_{i,j,MPRS} \times S_{i,j,RNS}) \\
O_{propagation} = \sum_{i=1}^{N_{update}} \sum_{j=1}^{N_{i,MPRS}} (P_{i,j,MPRS} \times S_{i,j,RNS})
\end{cases}
\end{aligned}
\tag{3.4}
$$

Some parameters are defined previously. The other variables are defined below.

- N_{hello} is the total number of periods in which hello messages are sent.
- $P_{i,j,hello}$ is the size of the ith hello packet sent by node j. The actual overhead in the RNS building phase can vary since the size of the hello messages varies.
- N_m is the number of RNSs maintenance operations due to link changes. The RNS maintenance operation is invoked each time the topology changes. Thus, N_m depends on random changes in the topology.
- $N_{i,adjust}$ is the number of nodes that need to adjust MPRs in the ith maintenance operation. When the topology changes, the MPRSs of some nodes within a two-hop range may change. This parameter depends on the exact MPR selection algorithm that is used and on network properties.
- $P_{i,j,MPRS}$ is the size of the MPRS declaration packet (topology control message) for node j in the nodes that adjust their MPRS sets when the ith link changes. The size of the MPRS declaration packet may vary, so the actual overhead due to MPRS declaration packets varies.
- $S_{i,j,RNS}$ is the size of the RNS for node j in the ith maintenance operation.
- N_{update} is the total number of periods in which broadcast topology control messages are sent.

- $N_{i,MPRS}$ is the number of nodes with a non-empty MPRS at the i-th periodic broadcast.

Parameters in OLSR's overhead are independent of traffic profiles. This is a common feature for proactive routing protocols. They can have a fixed upper bound on overhead in a network with any type of traffic. This can be an advantage for proactive protocols for networks with many traffic flows. For MANETs with large size and low link density, OLSR may suffer scalability problems because nearly all nodes need to re-broadcast control messages from other nodes when the link density is low for a large connected MANET.

3.3.5 Analysis of Simple Gateway Protocol with the RNS Framework

Wu, et al. presented a simple gateway protocol in [75]. Their basic approach is to use an algorithm (refer to Section 4.1.1 for details) to select, in a distributed manner, a minimal set of nodes that are connected and cover all the other nodes in the network. This set of nodes is called a connected dominating set (CDS). CDS nodes act as default gateways for nearby nodes. Default gateways broadcast lists of attached nodes. Only gateway nodes retransmit the information sent by other gateways. Because of the connectivity and coverage properties of those nodes in the connected dominating set, these control messages are propagated to all gateways. Thus, routes to all nodes can be built for every node. The idea of this simple gateway protocol is similar to that of OLSR if we map both protocols to the four-module RNS framework. From the point of view of the framework, both these protocols select subset of nodes as the RNSs and the broadcast control messages sent by the RNS nodes are all subsets of neighbor lists of the RNS nodes. One difference is that the simple gateway protocol builds only one copy of a RNS while OLSR builds multiple copies of RNSs. Another difference is that the information sent in the RNS of the simple gateway protocol is the default gateway information generated by RNS nodes, not MPRS information. (This concept of gateways is close to the concept of MPRS sets in

OLSR.) The simple gateway protocol builds RNSs in two phases. In the first phase, nodes declare themselves as RNS nodes once they have two non-adjacent neighbors. In the second phase, each RNS node selected in the first phase examines its RNS neighbors with larger node IDs. If there is such a neighbor that covers all of its other neighbors or there are two such neighbors that are adjacent and cover all its other neighbors, this node does not remain a RNS node. Here nodes use a special message type to declare themselves as RNS nodes. Like TORA, Wu, et al. do not fully address the module for unreliable transmissions. We assume that this protocol is similar to OLSR so the module to handle unreliable transmissions can be ignored. Thus, Equation 3.5 gives the overall overhead for the simple gateway protocol.

$$
\begin{aligned}
&Overhead = O_{construction} + O_{maintenance} + O_{propagation} \\
&\begin{cases}
O_{construction} = P_{annouse\_CDS}(S_{0,1st} + S_{0,CDS}) + \sum_{i=1}^{N_{hello}} \sum_{j=1}^{N} P_{i,j,hello} \\
O_{maintenance} = \sum_{i=1}^{N_m} [P_{annouse\_CDS}(S_{i,1st\_phase} + S_{i,2nd\_phase}) + \\
\qquad\qquad S_{i,CDS} \sum_{j=1}^{N_{i,adjust}} P_{i,j,gateway}] \\
O_{propagation} = \sum_{i=1}^{N_{update}} (S_{i,CDS} \sum_{j=1}^{S_{i,CDS}} P_{i,j,gateway})
\end{cases}
\end{aligned}
\tag{3.5}
$$

Some variables are the same as those used in previous equations. Others are defined as follows.

- $P_{annouse\_CDS}$ is the size of the packet for a node to announce itself as a CDS node to neighbors.
- $S_{0,1st}$ is the size of the CDS generated in the first phase at the initialization of the entire procedure.
- $S_{i,CDS}$ is the size of the i-th CDS when the second phase of the selection algorithm ends.
- $S_{i,1st\_phase}$ is the number of CDSs generated at the first phase in the i-th maintenance operation due to link changes.

- $S_{i,2nd\_phase}$ is the number of CDSs generated in the second phase in the i-th maintenance operation due to link changes.
- $P_{i,j,gateway}$ is the size of the gateway message generated by node j in the i-th periodic broadcast.

This protocol is very similar to OLSR from the RNS framework's point of view. Therefore, it has similar properties as we discussed in Section 3.3.4. The only difference between Wu's protocol and OLSR is that nodes in RNS are selected by themselves in Wu's protocol while they are selected by neighbors in OLSR. Which approach is better is still an open question and is currently being discussed in the IETF MANET group.

3.3.6 Analysis of TBRPF with the RNS Framework

TBRPF (see Section 2.3.6) is also a proactive routing protocol that provides shortest path routing. Each node uses periodic "hello" messages to detect links to its neighbors. Based on the local link state database, each node first builds a shortest path tree to all possible destinations. A node decides whether or not to report links in its shortest path tree to its neighbors by an estimation algorithm based on its local link state database. Information that is shared with a neighbor is considered to be reportable. Basically, a neighbor node is added to a reportable node set if this node has at least one neighbor which is not connected to this neighbor. Links in the shortest path tree are added to a reportable link set if one end point is in the reportable node set or one adjacent link that does not connect to the center node and is included in the reportable link set. Therefore, the reportable link set in the shortest path tree form a reportable tree. Each node broadcasts its reportable tree. This is the RNS building module in TBRPF. For any link, there is a set of nodes that broadcasts that link to neighboring nodes. Therefore, an RNS is built for each link in TBRPF. The control messages sent in TBRPF are reportable trees. Nodes have enough information to build proper shortest path trees based on reportable trees from neighbors.

When the topology changes, the maintenance module uses online computation to update corresponding RNSs and the link state update is propagated to all related nodes in the associated RNSs. Similar to OLSR, TBRPF uses periodic broadcast messages to handle unreliable transmissions. The overhead for TBRPF is presented in Equation 3.6.

$$
\begin{aligned}
&Overhead = O_{construction} + O_{maintenance} + O_{propagation} \\
&\begin{cases}
O_{construction} = \sum_{i=1}^{N_{hello}} \sum_{j=1}^{N} P_{i,j,hello} \\
O_{maintenance} = \sum_{i=1}^{N_m} \sum_{j=1}^{S_{i,RNS}} P_{i,j,LS} \\
O_{propagation} = \sum_{i=1}^{N_{update}} \sum_{j=1}^{E_i} \sum_{k=1}^{S_{i,j,RNS}} P_{i,j,k,LS}
\end{cases}
\end{aligned}
\tag{3.6}
$$

Some of these variables are the same as those in previous equations. Others are defined as follows.

- $S_{i,RNS}$ is the size of the RNS for link i.
- $P_{i,j,LS}$ is the size of a link state description for link i at node j.
- E_i is the number of edges in the i-th periodic broadcast in TBRPF.
- $S_{i,j,RNS}$ is the size of RNS for the link j.
- $P_{i,j,k,LS}$ is the size of a link state description for link j at node k in the i-th periodic broadcast.

Similar to OLSR and other proactive protocols, the control overhead in TBRPF depends on parameters for network profiles. In TBRPF, the RNS selection algorithm is based on shortest path trees maintained at distributed nodes. Therefore, nodes running TBRPF may require more processing times compared to OLSR.

3.3.7 Analysis of Hybrid Protocols with the RNS Framework

Other protocols, e.g., the landmark routing protocol with fisheye state routing protocol (see Section 2.3.7), the interzone routing protocol with intrazone routing protocol (see Section 2.3.8), are combinations of reactive and proactive protocols. Similar to TORA (see Section 2.3.4), we can also categorize these protocols by this RNS framework, although details are not presented here.

3.3.8 Analysis of Hierarchical and Cluster Protocols with the RNS Framework

From the perspective of the framework, the RNS in hierarchical and/or cluster routing protocols is also in the form of clustered node sets. Different control messages are propagated in different subsets of the RNSs according to the content of these messages and the roles of RNS nodes in the routing protocol. Generally, the RNS maintenance module in this type of protocol can limit the change of RNSs within a limited local range. That is why this kind of protocol is scalable.

Note that we can describe all these protocols in a similar way using the four-module framework. This implies that it is possible to use similar finite state machines to describe and formally verify MANET protocols and to implement them in simulators in a finite state machine based framework. This is a potential future research topic. In the next section, we illustrate the use of the framework through a comparative study of two link state routing protocls, OLSR and TBRPF.

3.4 Example Comparison of Two Proactive Protocols: OLSR and TBRPF

We discuss using the RNS framework to compare two protocols in this section. There are some schemes proposed in the literature to improve MANET routing protocols in terms of control overhead. Those schemes can be derived from the analytical model based on the RNS framework. For example, Perkins, et al. describe an effort to reduce the range of the RNS built when a route request is sent [50]. Perkins, et al. [50] and Johnson, et al. [49] incorporate routing caches to reduce the propagation range of control messages. It can be seen from the analytical model that these approaches are trying to limit the size of RNS in the RNS construction operation. In other words, reducing the blind broadcast range reduces the control overhead. Moreover, this analytical model can also guide us to improve MANET routing protocols in other ways. Since little research has been done to improve proactive routing protocols in terms of control overhead, the following paragraphs give such an example with OLSR.

OLSR uses "hello" messages not only to detect link connections, but also to exchange MPR information. So the overhead of "hello" messages in OLSR is larger than that of TBRPF. The packet size of $P_{i,j,MPRS}$ is formed by a header (MAC layer header and an IP header) and a data payload. Equation 3.7 illustrates the calculation of the packet size. Here, we assume that P_{unit} is the basic unit to describe a four-byte node ID. We can assume $P_{i,j,MPRS}$ equals the size of packet header (MAC and IP headers) plus several basic units to describe the MPRS. The link description packet used in TBRPF, defined as $P_{i,LS}$, shares one header since a node broadcasts a reportable tree. Each link needs at most two IDs, each of size P_{unit}. Some links with common nodes can lead to smaller packet sizes. This yields the upper bound shown in Equation 3.7.

$$
\begin{aligned}
&P_{i,j,MPRS} = P_{header} + (|MPRS| + 1) \times P_{unit} \\
&P_{i,LS} \leq P_{header}/S_{reportable\_tree,i} + 2P_{unit} \leq P_{header} + 2P_{unit}
\end{aligned}
\tag{3.7}
$$

In the RNS maintenance module, the overhead for OLSR, shown in Equation 3.4, equals the sum of products of $P_{i,j,MPRS}$ and $S_{i,j,RNS}$. Generally, if $P_{i,j,MPRS}$ increases, $S_{i,j,RNS}$ will also increase. Therefore, we can assume that the covariance between these two variables is greater than zero. Now, we have the lower bound for the OLSR overhead for the RNS maintenance module, shown in Equation 3.8. Based on Equations 3.6 and 3.7, the upper bound of the overhead for the RNS maintenance module for TBRPF can be formulated as shown in Equation 3.8.

$$
\begin{aligned}
O_{maint,OLSR} &\geq N_m \bullet \overline{N_{adjust}} \bullet \overline{S_{i,j,RNS}} \\
&= [P_{header} + (\overline{|MPRS|} + 1)P_{unit}]N_m\overline{N_{adjust}} \bullet \overline{S_{RNS,OLSR}} \\
O_{maint,TBRPF} &\leq N_m \textstyle\sum_{i=1}^{S_{i,RNS}}(P_{header} + 2P_{unit}) \\
&= N_m(P_{header} + 2P_{unit})\overline{R_{RNS,TBRPF}}
\end{aligned}
\tag{3.8}
$$

We used simulation to estimate the parameters for the size of the RNSs for these two protocols. Since dynamic topologies can be considered as a sequence of static topologies, we implemented these protocols in C++ with sequences of static topologies. Simulations were done using a 100×100 unit square map. Two nodes can communicate with each other if the distance between them is less than the given maximum radio range. The number of nodes ranged from 2 to 100. OLSR and TBRPF were simulated. The latest OLSR draft states that a node "should select an MPR set such that any two-hop neighbor is covered by at least MPR_COVERAGE MPR nodes" [53]. We assume that the minimum MPR set is used in OLSR, i.e., MPR_COVERAGE equals 1. We generated 1000 random connected topologies for each set of parameters and obtained the average size of RNSs. Results are shown in Figures 3.3, 3.4, and 3.5 with radio ranges of 25, 50, and 75 units,

respectively. Note that we only show the average values in these figures. Since for a given number of nodes in a network, say N, the range for all possible RNS sizes is [0, N]. Therefore, the variance of results can be large. In other words, our comparison results only give an idea on the average performance and which protocol generates smaller RNSs really depends on the actual MANET application it applies to.

According to the simulation results, in most cases when the maximum radio ranges were 25 and 50, the average size of RNSs in OLSR is larger than in TBRPF. When the maximum radio range was 75, the size of RNSs in TBRPF and OLSR leveled off to constant values of a little over two for TBRPF and a little less than two for OLSR. Generally, the size of the MPRS set was greater than or equal to one. We also assumed that $N_{i,adjust}$ is greater than or equal to one. Therefore, based on these results and assumptions, OLSR usually has larger overhead in the maintenance module than TBRPF.

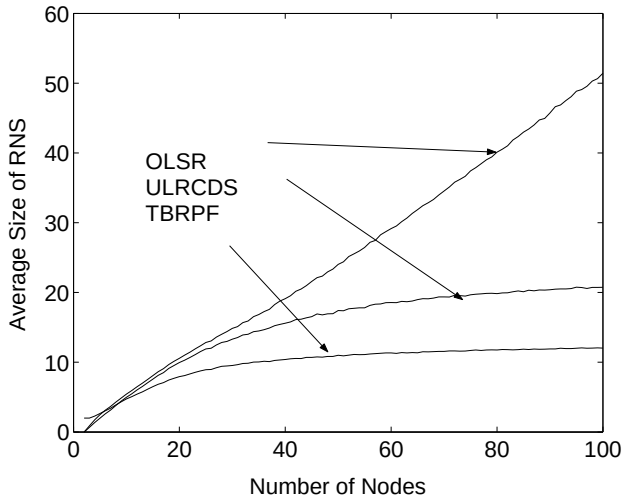


Figure 3.3: Average size of RNS versus number of nodes (Radio range=25).

According to the RNS framework, we can improve the first two modules in OLSR without increasing the overhead in the propagation module. For example, the Updated Local Ripple CDS (ULRCDS) algorithm described in Section 4.1.4 with smaller average RNS sizes should have less overhead in the building and maintenance modules for OLSR. Therefore, we can reduce the control overhead in OLSR using ULRCDS. Using other al-

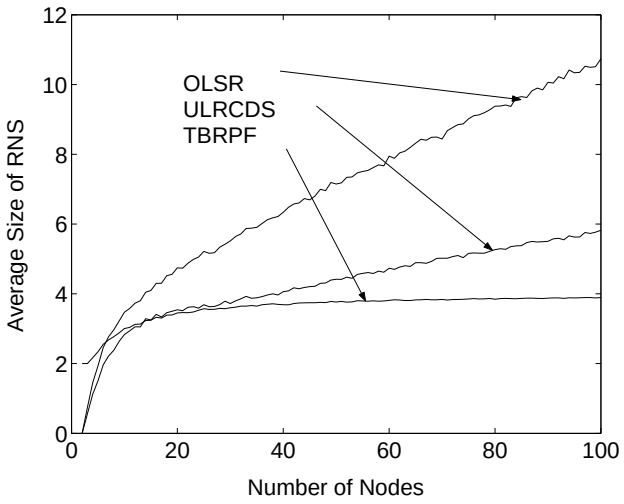


Figure 3.4: Average size of RNS versus number of nodes (Radio range=50).

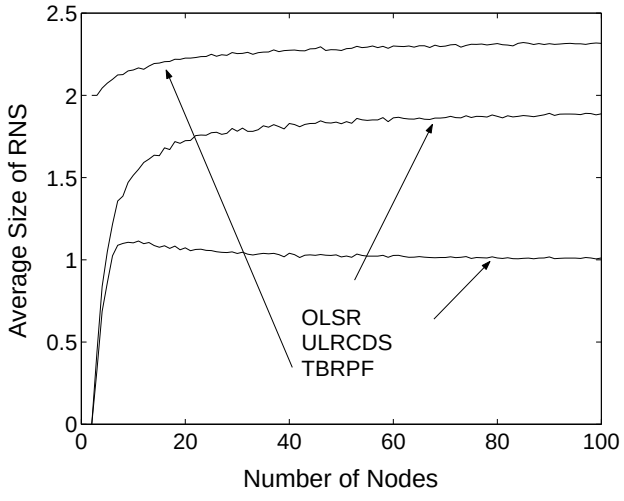


Figure 3.5: Average size of RNS versus number of nodes (Radio range=75).

gorithms that generate small size of MPRS is also mentioned in the OLSR draft [53] as possible direction to reduce OLSR control overhead. ULRCDS can build one copy of the connected dominating set as the RNS for the network. The "hello" messages are similar to those of OLSR. Two-hop neighbor information is used. (Refer to Section 4.1.4 for details of this algorithm.) Simulation results shown in Figures 3.3, 3.4, and 3.5 show that the average RNS size in the ULRCDS is smaller than that in OLSR. Therefore, it may result in less overhead than the algorithm used by OLSR.

3.5 Environmental Factors for MANET Routing Protocols

We described the framework and different MANET routing protocols in the previous sections. This section discusses environmental factors that affect the operations and, thus, the performance of different MANET routing protocols.

Based on the analytical model of control overhead shown in Equation 3.1, the environmental parameters that can affect the control overhead in a MANET are listed in Table 3.1 with brief definitions. Basically, the control overhead of proactive protocols, such as OLSR and TBRPF, depends on the values of Δ and D. The control overhead of reactive protocols, such as AODV, depends on the values of Δ, D, λ, L, and TF. Currently, most widely used mobility models are defined based on node movement patterns. The relationships between parameters used in common mobility models and variables listed in Table 3.1 are not clear. Therefore, we investigate trends of control overhead with respect to variables defined in Table 3.1 in Chapter 7. In a real application environment, these parameters are often measurable or can be estimated with reasonable accuracy. Therefore, it is possible to adaptively adjust routing protocol parameters or even change protocols according to the parameters measured in real-time in a system. This is beyond the scope of our research and may be a topic for future research.

Table 3.1: Environmental Parameters that Affect Control Overhead

Variables	Definitions
N	Number of Nodes
E	Average Number of Edges
T(UP)	Average Link Duration
T(DOWN)	Average Link Down Time
Δ	Average Degree $(2E/N)$
D	Link Density $(\Delta = 2E/N^2)$
λ	Average Number of Link Change Rate
L	Average Length of Path
TF	Number of Traffic Flows

3.6 Summary and Conclusions

We presented a framework based on the concept of a relay node set that can characterize MANET routing protocols. We developed an analytical model with the RNS framework for control overhead for MANET routing protocols. Simple examples were used to show how we can compare and, possibly improve routing protocols using the RNS framework. There are some parameters defined in the analytical model that may not be measured directly for a MANET application. This is a limitation of using the RNS framework. One suggested approach is to use simulations or real-time measurements to estimate those values. This suggests a potential research topic for MANET routing protocols in which estimates of environmental parameters, including network and user application profiles, are used to adaptively choose different routing protocols or different sub-functions for one protocol.

We can describe and analyze a MANET routing protocol using the RNS framework. We can also use the RNS framework and the analytical model to provide guidance to design new protocols and improve existing protocols. For example, to achieve low control overhead, we can design a proactive algorithm that constructs a small RNS, which is

discussed in Chapter 4.

Guided by the RNS framework, we can estimate of trends of control overhead for different protocols when the network and/or user traffic profiles change (See Chapter 7). Simulations summarized in Chapter 7 can also provide guidance for users to select proper MANET routing protocols according to their application environments.

Chapter 4

A New Link State Proactive Routing Protocol

Guided by the RNS framework described in Chapter 3, we designed and developed a link state proactive routing protocol, which we call Open Shortest Path First using Minimal Connected Dominating Set algorithm (OSPF-MCDS). It maintains full topology in distributed nodes and provides shortest path routings. In this chapter, we present our work on this routing protocol. The content of this chapter was summarized in [72].

4.1 Connected Dominating Set

A common source of overhead in a wireless ad hoc network comes from blind broadcast. In a wireless ad hoc network, a node rebroadcasts all received broadcast messages. Nodes may receive multiple copies of the same message from more than one neighbor. Therefore, according to the RNS framework, reducing redundant broadcast messages can reduce channel bandwidth consumption and increase bandwidth efficiency. A minimal *connected dominating set* (CDS) can be used to reduce redundancy due to blind broadcasts.

In a simple graph $G(N, E)$, N is the set of nodes and E is the set of edges. Assume a node set $T \subset N$ such that for all u in $N - T$, there exists $v \in T$, such that edge $(u, v) \in E$. This is the cover property for the CDS. Set T is called a dominating set. Set T is also called a connected dominating set (CDS) when T forms a connected graph. This is the connectivity property for the CDS [41].

Figure 4.1 gives an example of a CDS. Black nodes 2 and 3 are connected and cover all nodes in the network. They form a CDS for this graph. It is obvious that broadcasting by using nodes in a CDS as relay nodes can reduce redundant rebroadcasts compared with blind broadcast. Broadcast messages can be propagated to all nodes in the CDS because of the connectivity property of the CDS. Non-CDS nodes are able to receive messages because of the cover property of the CDS.

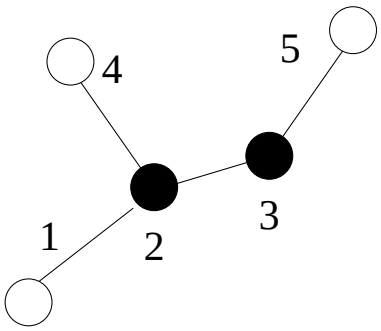


Figure 4.1: An example of a CDS in a simple network.

Assume all nodes rebroadcast broadcast messages that are not seen before. The redundant traffic reduced by a CDS can be indicated by the size of the non-CDS node set divided by the total number of nodes in the graph. A simple simulation is presented here to show the improvement as compared to blind broadcast. In the simulation, n nodes are randomly generated in an area of 100×100 square units. Radio range is used to decide whether or not two nodes are connected. Three radio ranges, 25, 50, and 75, are used and only connected networks are used. To find an optimum CDS for all topologies, all possible node sets are examined. The CDS with the minimum size, the minimum CDS (MCDS), is kept. For each set of parameters, we tried 1000 replications with different random node

placements. Figure 4.2 shows the percentage of reduced overhead due to using a CDS compared with blind broadcast. Similar to the simulations presented in Section 3.4, we only show the average values in Section. Since for a given number of nodes in a network, say N, the range for all possible reductions is $[0, 100\%]$, the variance of results can be large. In other words, the simulation results presented in this section only give a rough idea on the average reduction and the actual improvement really depends on network topologies.

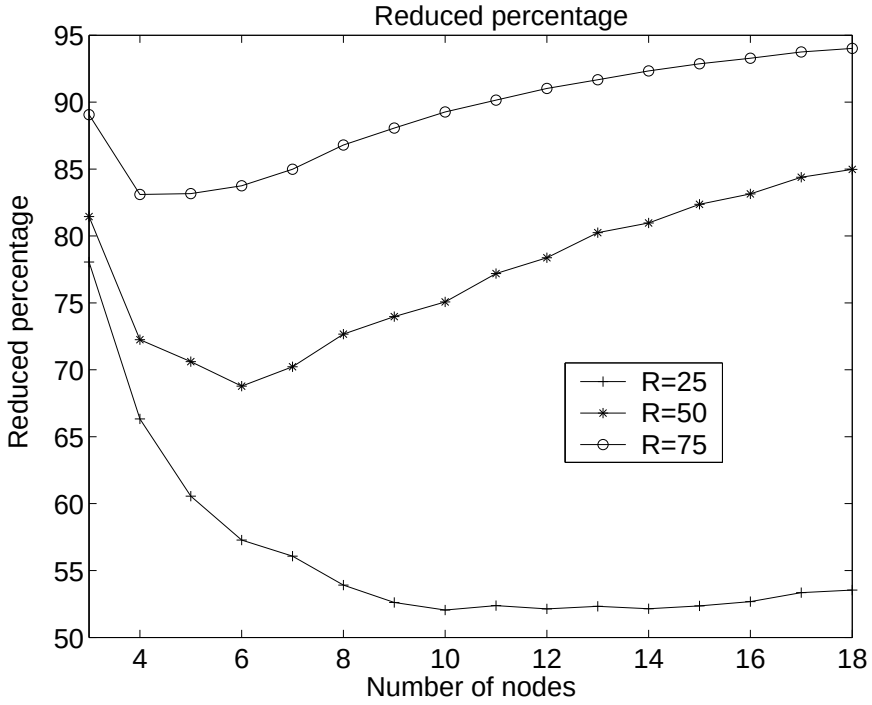


Figure 4.2: Reduction of broadcast overhead using optimum CDS.

According to Figure 4.2, overhead is reduced by over 50% for all radio ranges and values of n. Savings increase when radio range increases or number of nodes increases, implying better performance in a highly dense network. Therefore, broadcasting based on a small CDS can be a promising approach to reduce routing overhead. In this chapter, we examine and propose a new link state proactive routing protocol using CDSs to efficiently broadcast control messages.

Finding a minimal CDS for a connected graph is an NP-complete problem. For a

small graph, we can enumerate all possible cases to find a minimal solution. However, this approach is not feasible for larger graphs. Different approximation algorithms to find a minimal CDS are proposed in the literature [73, 74, 75]. Algorithms that use all information in the network are called global algorithms. Local algorithms utilize only local neighbor information. Das and Bhargavan [74] propose implementations of two global algorithms described by Guha and Khuller [73]. Wu and Li present a local distributed algorithm [75].

In Section 4.1.1, we introduce others' work on approximation algorithms for finding a minimal CDS. We present four algorithms we developed in the following sfections. Section 4.1.6 shows the analysis of the approximation algorithms in terms of time and message complexity. Simulation studies of different algorithms are presented in Section 4.2. Section 4.3 proposes a proactive link state routing protocol integrated with a CDS approximation algorithm. We present our conclusions and an outline of future work in Section 4.4.

4.1.1 Survey of Existing Algorithms

In this section, we present approximation algorithms for finding a minimal CDS proposed in [73], [74] and [75]. The two global algorithms proposed by Das, et al. assume that nodes keep identical copies of the entire topology. (This is always true in a proactive link state routing protocol.) The first algorithm, which we name Das CDS I algorithm, defines an effective degree for every node. The effective degree of a node is the number of its non-CDS neighbors. The Das CDS I algorithm has two stages. The first stage is responsible for finding a small dominating set C' using a loop. At the beginning of an iteration in this loop, nodes are assigned weights – their effective degree. One iteration ends by adding the node with the largest weight to C'. The loop in the first stage stops when C' covers all nodes. Note that after the first stage, C' may contain several disconnected components. In the second stage, the Das CDS I algorithm tries to use the minimum number of extra

nodes to connect the components of C′ so that a CDS can be formed. In this stage, links are assigned weights. If a link connects two non-CDS nodes that are in the same component, this link is assigned a weight of infinity. The weight for other links is the number of end points that are not in C′. The lightest links are chosen to connect disjointed components. This algorithm ends when a CDS is found.

The second algorithm, which we call Das CDS II, starts at the node with the highest degree. It extends the set of selected CDS nodes by including one of the nodes adjacent to the CDS set with the largest number of uncovered neighbors or two-hop uncovered neighbors until no uncovered node is left. Both of Das and Bhargavan's algorithms are based on Guha and Khuller's approximation algorithm [73]. Therefore, they have similar performance in terms of the average size of the final CDS sets.

The time complexity for the Das CDS I algorithm is $O((\nu + \gamma)\Delta)$, where ν is the number of nodes in the network, γ is the size of the final CDS, and Δ is the maximum node degree. The Das CDS II algorithm has time complexity $O((\gamma + \Delta)\gamma)$. A metric, the performance ratio, is defined to characterize the worst-case performance of approximation algorithms in [73]. Assume that an algorithm can find a CDS with size C and that the minimum CDS size is C^* . The performance ratio is defined as the maximum value of C/C^*. Analysis in [73] shows that these two algorithms achieve the performance ratios of $2H(\Delta) + 1$ and $2H(\Delta)$, respectively, where $H(\Delta) = \sum_{i=1}^{\Delta}(1/i) \leq ln\Delta + 1$.

The local algorithm, which we call Wu CDS, is implemented in a distributed manner. This algorithm uses two phases and assumes that all nodes know all the other nodes that are within their two-hop range. In the first stage, a node is selected as a potential member of the CDS if and only if it has two non-adjacent neighbors. Nodes broadcast if they elect themselves as members of the potential CDS in the first phase. Two extensions are used in the second phase to reduce the size of the CDS. A node stays in the CDS unless a neighbor CDS node with a larger ID covers its entire neighbor set. As an extension, if the neighbor set of a node is covered by two adjacent CDS neighbors with larger IDs, this

node may change itself to a non-CDS node. The time complexity turns out to be $O(\Delta^2)$, where Δ is the maximum node degree. This is better than the two global algorithms described above. Comparison has been done between Wu CDS and Das CDS I in [75]. Simulation results show that Wu CDS produces a smaller CDS than Das CDS I, except in the case of a high-density network [75]. Because the Das CDS II algorithm is close to Das CDS I, we can expect similar results if we compare Wu CDS and Das CDS II. Another similar local algorithm is presented in [76]. This is a modest extension of Wu CDS. Similarly, there is another paper discussing a CDS algorithm for broadcast in MANETs [77]. Their approach is similar to one algorithm discussed in [74]. Therefore, we do not compare these two algorithms in this work.

In the following sections, we present four new approximation algorithms to find a minimal CDS for a given graph. Two are global algorithms and two are distributed local algorithms. We assume that nodes have unique IDs. Control messages travel like water ripples in all directions from the center. We simulate this phenomenon in a network by emulating a broadcast procedure (described in Section 4.1.2) or by broadcast of a real message (described in Sections 4.1.3 and 4.1.4). Based on observations, we also developed another global algorithm (described in Section 4.1.5). Before we present our algorithms, we first define two concepts.

Definition 4.1.1 *The n-hop neighbor set of node i consists of nodes whose minimum distance from i is exactly n and is denoted as $N_i[n]$. In other words, $\forall j \in N_i[n]$, the number of hops in the shortest path from i to j is n.*

As an example, consider the topology of Figure 4.1. For node 1, $N_1[0] = \{1\}$, $N_1[1] = \{2\}$, $N_1[2] = \{3, 4\}$, and $N_1[3] = \{5\}$.

Definition 4.1.2 *Node i is covered by node j when node i is the neighbor of node j, i.e. node i and j are in each other's radio range. A set of nodes N is covered by set of nodes M iff every node in N is covered by at least one node in M.*

We assume that nodes periodically flood HELLO messages with neighbor lists to one-hop neighbors. This means that nodes have knowledge of other nodes within a two-hop range. In a wireless ad hoc network, broadcast messages may not travel exactly in the same pattern in all directions because of channel delays or processing delays in the nodes. Therefore, our second assumption is that a node receives duplicate broadcast messages from all of its neighbors in a certain time T_0. This is a reasonable assumption because if time T_0 is larger than the idle interval time for the hello protocol, neighbors that do not send back expected messages are already considered to be lost. In Sections 4.1.2, 4.1.3, 4.1.4, and 4.1.5, we first assume that the underlying MAC layer protocol guarantees delivery (Section 4.1.7 presents further discussions that does have have assumption). Note that duplicate messages received by a node in $N_i[n]$ may only come from nodes in $N_i[n-1]$, $N_i[n]$, and $N_i[n+1]$. The next subsections present four approximation algorithms for a minimal CDS: global ripple algorithm, local ripple algorithm, updated local ripple algorithm, and global CDS algorithm.

4.1.2 Global Ripple Algorithm for Minimal CDS

The global ripple algorithm (GRCDS) assumes all nodes try to maintain identical copies of the entire network topology. This assumption is reasonable when the network mobility is low. When a link changes, the network is in a transient state in which some nodes have different copies of the topology database. We concentrate on low mobility cases for this algorithm at this stage. Here the terms of low mobility is defined as small link connectivity change rates with respect to the frequency to exchange control messages in the CDS algorithms. (This assumption is required for all proactive routing protocols. Otherwise, none of those proactive routing protocol can work properly.)

The algorithm runs locally with global topology information. There are two stages. In the first stage, the algorithm broadcasts a zero-payload or "virtual" message starting at the node with the minimum node ID. The initiator of this message is marked as a

potential CDS node. When a node broadcasts, all of its one-hop neighbors can receive the broadcast message.

Rule 4.1.2.1 *A node j in $N_{Initiator}[n]$ rebroadcasts the first-received virtual message and is marked as a potential CDS node unless the sender with the smallest node ID among all senders in $N_{Initiator}[n-1]$ covers all neighbors of node j.*

A node may receive duplicate messages from several neighbors. To ensure that the results of the distributed algorithm are deterministic, we assume that nodes in $N_i[n]$ always examine the first node in $N_i[n-1]$ that broadcasts the virtual message to it. The node with the smaller ID wins a tie. Since the global algorithm starts with the node having the smallest ID and ends with the node having the largest ID, the processing order and, hence, results, are deterministic. We assume nodes in $N_i[n]$ receive all duplicate messages from their neighbors in time T_0. A node eventually gets to know whether or not its neighbors are potential CDS nodes by examining the sources of duplicated messages.

In the second stage, a potential CDS node decides whether or not to stay in the CDS based on the list of potential CDS neighbors, together with neighbor list information. Potential CDS nodes are examined in ascending order of the node ID. A potential CDS node performs a coverage check and a connectivity check in this stage. The coverage check is used to determine if the potential CDS node's non-CDS neighbors are covered by its potential CDS neighbors. Whether or not these CDS neighbors are connected is determined by checking connectivity.

Rule 4.1.2.2 *If results of both coverage and connectivity checks are positive, the potential CDS node is marked as a non-CDS node.*

Figure 4.3 presents an example of GRCDS in an eleven-node network. The potential CDS nodes, selected using Rule 4.1.2.1, are in black in Figure 4.3 (a). Figure 4.3 (b) shows the final CDS in black after the second stage.

In this example, GRCDS initiates a "ripple" from node 1, the node with the smallest node ID. Node 1 is marked as a potential CDS node in the first stage. According to Rule 4.1.2.1, nodes 2, 4, 5, 7, and 8 are also marked as potential CDS nodes. Note that nodes 4 and 5 are both marked because their ripple source, node 2, does not cover all of their neighbors and node 3 is not marked because its ripple source, node 2, covers all of its neighbors. In the second stage, Rule 4.1.2.2 removes node 4 because its CDS neighbors (nodes 2, 5, 7 and 8) cover all of its neighbors and form a connected node set. Node 1 is removed for the similar reason. Therefore, the final size of the CDS is 4, which happens to be the optimum solution. Complexity analysis and simulation results are shown in Section 4.1.6 and Chapter 6.

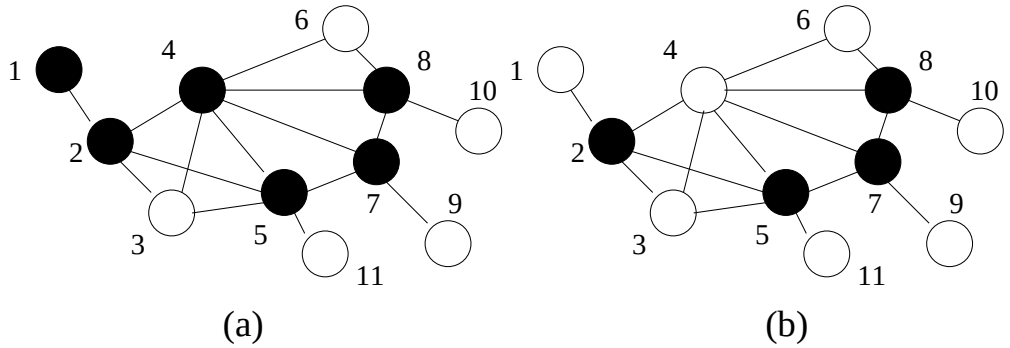


Figure 4.3: Example of GRCDS.

Now we prove that the node set formed by GRCDS is always a CDS for a given graph G.

Assertion: Prove the final node set generated by the GRCDS is a CDS.

Proof: First, we describe the algorithm as follows. Let S stand for the initial node that sends out the virtual message, N and CDS represent the node sets that received the virtual message and rebroadcast the message, respectively. At the beginning, $N = CDS = S$. Nodes in the neighbor set of CDS which are not in N, say $N[CDS]$, receive this virtual message for the first time. Therefore, $N = N \bigcup N[CDS]$. We use $N_{rebroadcast}[CDS]$ to represent the nodes in $N[S]$ that rebroadcast the virtual message. Therefore, $CDS =$

$CDS \bigcup N_{rebroadcast}[CDS]$. If N contains all nodes in the network, this procedure stops.

Now we prove the connectivity feature. According to the procedure we described above, the set CDS keeps increasing by adding only neighboring nodes that rebroadcast the virtual message until the first stage algorithm stops. Therefore, the final CDS is a connected node set.

Now, we prove the coverage property in the first stage by using contradiction. Assume there exists a node i that is not covered by the final CDS after the algorithm stops. In other words, it is not in the final CDS and is not adjacent to any nodes in CDS. Therefore it cannot receive the virtual message since, according to the procedure described above, node i has no neighbor that broadcast the virtual message. In other words, node set N does not contain all nodes because there is at least one node, i, that is not in N. Thus, we have a conflict with our assumption. Therefore, the coverage property is proved.

Based on the proof of connectivity and coverage properties, the final node set is a CDS. □

4.1.3 Local Ripple Algorithm for Minimal CDS

GRCDS is based on the assumption that all nodes maintain identical copies of the topology. Although GRCDS is a global algorithm, it checks each node only based on the local information. This suggests the idea for a local ripple algorithm for a minimal CDS (LRCDS). The LRCDS algorithm uses a real control message, named SETUP_MCDS. To guarantee that all nodes have the same result with this distributed algorithm, the control message processed later by all nodes should be initiated by one node.

We assume that this unique initiator is the node with the minimum node ID in the network. If nodes have full topology information, they are able to determine the node with the minimum ID. However, if nodes in a network only have part of the topology

information, they cannot find the minimum node ID in the network. In this case, we define an initiator ID field in the SETUP_MCDS message. Nodes first assume themselves to be the node with the minimum node ID if their IDs are the minimum ones according to their best knowledge of the entire node set in this network. These nodes initiate their SETUP_MCDS messages using their own IDs. When other nodes receive multiple SETUP_MCDS messages initiated by different source nodes, they can check the originator ID field so that the messages with larger originator IDs are discarded, together with the corresponding results. Thus, with the extra control overhead in the SETUP_MCDS messages, we can guarantee the final results of the LRCDS algorithm in all nodes are based on the same initiator of the SETUP_MCDS message. In the following discussion, we assume that every node in a network knows the minimum node ID in the network.

Rule 4.1.3.1 *A node rebroadcasts the first-received message and marks itself as a potential CDS node unless the first sender covers all of its neighbors.*

Rule 4.1.3.2 *If results of both coverage and connectivity checks are positive, the potential CDS node is marked as a non-CDS node.*

The LRCDS algorithm uses rules similar to GRCDS to select and maintain the CDS based on the local neighbor information. According to Rule 4.1.3.1, the sender of the SETUP_MESSAGE message is considered as a potential CDS node. Therefore, nodes can sniff the SETUP_MCDS messages and keep a list of potential CDS neighbors. By our earlier assumption, after time T_0, nodes receive the lists of all potential CDS neighbors. To avoid the case where two potential CDS neighbors unmark themselves because they each believe that the other node remains marked, nodes only check CDS neighbors with larger IDs. It is easy to prove that the result at node i with Rule 4.1.3.2 does not affect the results at neighbors with larger IDs. In the second stage, coverage checking determines whether or not all neighbors are covered by the CDS nodes with larger IDs. Connectivity checking also considers only the list of CDS neighbors with larger IDs.

In a manner similar to that for GRCDS, we can prove the set of nodes formed by LRCDS is a CDS. Complexity analysis and simulation results are discussed in Section 4.1.6 and Chapter 6, respectively.

4.1.4 Updated Local Ripple Algorithm for Minimal CDS

As reflected in the simulation results reported in Chapter 6, the two ripple algorithms presented in the previous subsections do not have good performance in highly dense networks when compared with Das and Bhargavan's [74] approximation algorithms. One reason is that Das and Bhargavan's algorithms favor nodes with larger degrees. To address this problem, we propose the updated local ripple algorithm for a minimal CDS (ULRCDS). This algorithm assumes that nodes periodically exchange neighbor lists. Therefore, nodes are aware of their neighbors' neighbor lists and degrees.

This revised algorithm favors nodes with larger degrees. The simulation results in Chapter 6 show significant improvements due to this modification. To make this algorithm more robust for wireless ad hoc networks, we present some ideas for cases when the network topology changes. Assume a node notifies its neighbors if it is deleted from the CDS by Rule 4.1.4.2. For example, a CDS_DEL message can be used. Then, nodes can keep lists of CDS neighbors by sniffing SETUP_MCDS messages and checking CDS_DEL messages. When a link-up event occurs in the network, only the CDS neighbors of the two end points of this link need to check Rule 4.1.4.2. A link-up event does not increase CDS size unless a new node joins the network. The end node of this link in the CDS, or the node with the smaller ID if neither is a CDS node, is responsible for broadcasting information about this new link.

Rule 4.1.4.1 *When a node i has a neighbor that is not covered by the first sender, say node j, node i checks whether there is a common neighbor between i and j, say node x, that covers that neighbor. Node i rebroadcasts the message if such a node x does not exist or if x has smaller degree. In the*

case of a tie, the node with the smallest ID is chosen to rebroadcast the message.

Rule 4.1.4.2 *If results of both coverage and connectivity checks are positive, the potential CDS node is marked as a non-CDS node.*

Link-down events can occur in three different types of links: a link between two non-CDS nodes, a link between a CDS node and a non-CDS node, and a link between two CDS nodes. In the case of a link-down event between two non-CDS nodes, the CDS does not change since this link does not change the coverage and connectivity of the CDS. The end node with the smaller ID broadcasts information about the loss of this link. In the case of a link-down event between a CDS node and a non-CDS node, the CDS node broadcasts this event and checks Rule 4.1.4.2. The non-CDS node checks its local list of CDS neighbors. If all the other neighbors are non-CDS nodes, this node broadcasts the link-down event with a LOOKUP_CDS message to its neighbors. The neighbors announce themselves as potential CDS nodes by replying with LOOKUP_CDS_REPLY. They check Rule 4.1.4.2 after time T_0. Note that both end points of this link broadcast the link-down information. In the case of a link down between two CDS nodes, the CDS may become partitioned. Therefore, both CDS end points broadcast the link-down event to their remaining CDS neighbors with a special RESET_CDS bit set. A new CDS selection procedure starts when the node with the minimum known ID receives the message with the RESET_CDS bit set.

Since the nodes keep a copy of all known nodes, a node can detect a new neighbor that is new to this network. A full copy of link state databases is exchanged. Two networks are merging if all node IDs received from the new neighbors are unknown. In this case, the end points of the new link broadcast the database from one network to another. The CDS is re-selected after the merge.

The algorithm and ideas presented in this section are used for a CDS discovery protocol in a proactive link state routing scheme for wireless ad hoc networks, OSPF-

MCDS, as described in Section 4.3.

4.1.5 Global CDS Algorithm

Global algorithms have the advantage of being more likely to produce a smaller CDS. Therefore, a global algorithm with minimal CDS size (GCDS) could be a better choice in some environments, such as small- or medium-sized low mobility networks. In this section, we present another global algorithm for these types of networks. According to the simulation results presented later, this algorithm gives the best CDS among all four algorithms.

Here are two necessary definitions. A *closed neighbor set of node i* is a set of neighbor nodes of i in which any node's neighbor is either in this set or is node i. Note that not all nodes have closed neighbor sets. If a node does have at least one such closed neighbor set, it is possible to generate a closed neighbor set by starting from any node in that closed neighbor set. An *open neighbor set of node i* is a set of neighbor nodes of i that does not include any closed neighbor set. Some nodes in this set have neighbors that are not adjacent to node i. These nodes are referred to as *exit nodes*.

We categorize the candidate CDS nodes into two types. A *potential* CDS node is likely to be a CDS node, while a *primary* CDS node must be in the CDS. Primary CDS nodes include all the nodes that have at least one closed neighbor set and at least one neighbor that is not in that closed neighbor set. For example, in Figure 4.1, both nodes 2 and 3 are primary CDS nodes because they have closed neighbor sets 1 and 5, respectively, and they both have at least one neighbor, 4 and 2, respectively, which is not in one of their closed neighbor sets. A node i becomes a potential CDS node if all of its neighbors form a closed neighbor set and node i has the largest degree compared with other neighbors. In case of a tie, the node with the larger ID is selected. This rule also breaks ties in the following cases. Note that if the network is fully connected, no node is chosen as a

candidate CDS node. If an open neighbor set of node i is detected and one exit node that covers all neighbors has larger degree than node i, node i maintains its type. Otherwise, it becomes a potential CDS node. For all other cases, the nodes become non-CDS nodes. Note that a node with degree of 1 is always a non-CDS node.

This algorithm assigns CDS or non-CDS nodes with proper types. In the second stage, the algorithm checks potential CDS nodes in ascending order of node IDs. (We tried checking in the order of node degrees, but this did not improve the performance in terms of the average size of the CDS.) For each potential CDS node, if its CDS neighbors cover all of its neighbors and they are connected in the entire potential CDS, this node becomes a non-CDS node. The proofs for connectivity and cover properties of the final CDS are similar to those of previous algorithms.

When we use GCDS in a proactive link state routing protocol for wireless ad hoc networks, nodes do not need to transmit extra messages. Nodes only need to do an online computation. The time complexity is proportional to $O(\nu\Delta^2)$, where ν is the number of nodes in the network and Δ is the maximum node degree. When a node sees a link-down event, it first decides whether or not to rebroadcast the event according to its node type. Then the node updates its topology database and recomputes the entire CDS node set. When a node sees a link-up event, it re-computes the CDS set and decides to re-broadcast or not according to the computation results. This algorithm trades computation time and processing energy for control overhead traffic and communications energy.

4.1.6 Analysis and Comparison of Algorithms

We analyze the message, time, and memory complexity of the four CDS approximation algorithms in this section. The GRCDS algorithm does not require extra messages except HELLO messages and other link state information exchanges. Assume m is the number of edges in the network, γ is the size of final CDS, and ν is the number of nodes in the net-

work. The message complexity is $O(m\gamma+\nu)$ if we count the HELLO messages introduced by the underlying hello protocol since each node in the final CDS broadcasts the topology and each node declares its associated interfaces and network prefixes. The memory complexity is $O(\nu^2)$ since for each node in the network, we may examine all of its neighbors, which is at most ν. The time complexity of Rule 4.1.2.1 at each node is $O(\Delta)$, where Δ is the maximum node degree. Nodes broadcast a sorted neighbor list of length $O(\Delta)$. So for a total ν nodes, the first stage takes $O(\nu\Delta)$. For each CDS node obtained from Rule 4.1.2.2, the coverage check takes $O(\Delta^2)$ and the connectivity check time is proportional to $O(\Delta^2)$ if we use a depth-first search (DFS) [78]. So the second stage takes $O(\nu\Delta^2)$. Here ν is the size of the network and it is the upper bound of the size of potential CDS obtained in the first stage. The LRCDS algorithm does require extra SETUP_MCDS messages, so the extra message complexity is $O(\nu)$ since, in the worst case, each node rebroadcasts the SETUP_CDS message. The memory complexity is $O(\Delta^2)$ because each node only maintains a two-hop neighbor matrix and the time complexity is $O(\Delta^2)$ because the LRCDS operates on the two-hop neighbor matrix. ULRCDS and GCDS have the same message, time, and memory complexities as LRCDS and GRCDS, respectively.

Wu CDS [75] and Das CDS I and Das CDS II [74] are compared with our algorithms in Table 4.1. Note that we only count extra messages for these algorithms. (HELLO messages and exchanges of link state information are considered as overhead for general routing, not for CDS discovery.) From Table 4.1, we see that Wu CDS, LRCDS and ULRCDS have the smallest time complexity. This is an advantage of a local algorithm. The number of extra control messages used by LRCDS and ULRCDS are the same as in Wu CDS.

Simulation results to compare these algorithms are shown and discussed in Chapter 6. In the following section, we present an extended study of these algorithms.

Table 4.1: Comparison of Different Approximation Algorithms

Algorithm	Time Complexity	Extra Message
Das CDS I	$O((\nu+\gamma)\Delta)$	0
Das CDS II	$O((\gamma+\Delta)\gamma)$	0
GRCDS	$O(\nu\Delta^2)$	0
GCDS	$O(\nu\Delta^2)$	0
LRCDS	$O(\Delta^2)$	$O(\nu)$
ULRCDS	$O(\Delta^2)$	$O(\nu)$
Wu CDS	$O(\Delta^2)$	$O(\nu)$

4.1.7 Extended Study of CDS algorithms

In this section, we prove two hypotheses. The first one is that LRCDS, ULRCDS, GRCDS, and GCDS can always have better performance than Wu CDS [75] in terms of the size of the resulting CDS, which is also shown in our simulation results in Chapter 6. The second hypothesis is that the GCDS algorithm can work properly without support for guaranteed delivery by MAC layer protocols. This implies that we can apply the GCDS algorithm directly to the link state routing protocol. We prove the first hypothesis next.

Assertion: LRCDS, ULRCDS, GRCDS, and GCDS always have equal or better performance than Wu CDS.

Proof: In the first step of Wu CDS, nodes mark themselves if they have any two nonadjacent neighbors. Marked nodes declare themselves by broadcasting a message to their neighbors. Two rules are used in the second stage. Define *N[u]=*{*neighbors of node u*} $\bigcup$ {*u*}.

Rule 4.1.7.1 *A marked node u is unmarked if N[u] belongs to N[v], where v is a marked neighbor of u, and ID(u) < ID(v).*

Rule 4.1.7.2 *A marked node u is unmarked if* {*neighbors of node u*} *belongs to the union of*

{neighbor set of v} and {neighbor set of w} where v and w are neighbors and are both marked neighbors of node u, ID(u)=min{ID(u), ID(v), ID(w)}.

In the LRCDS algorithm, a node marks itself as a CDS node if and only if the sender of the received SETUP_MCDS message does not cover all of its neighbors. This implies that this node is guaranteed to be marked in the first stage of Wu CDS unless it is the initiator. The converse does not apply. The initiator is a CDS node in the first phase of LRCDS. Note that the initiator has the smallest ID in the network so it becomes a non-CDS node in the next phase in LRCDS if it is a non-CDS node in the first phase in Wu CDS. Therefore, in the first stage, the size of the candidate CDS set in LRCDS is at most one more than that in Wu CDS. In the second stage of LRCDS, Rule 4.1.3.2 is used to examine possible CDS nodes. Recall that Rule 4.1.3.2 checks the cover and connectivity properties. Coverage checking passes when all neighbors are covered by the set of CDS neighbors with larger IDs. Connectivity checking passes when all CDS neighbors with larger IDs are connected. It is easy to prove that both extensions in Wu CDS imply that Rule 4.1.3.2 passes. But, Rule 4.1.3.2 also applies when the size of the CDS neighbors with larger IDs is greater than two, such as in the example of Figure 4.3.

GRCDS and GCDS use more information to remove CDS nodes than LRCDS in the second stage. ULRCDS has a smaller CDS than that of LRCDS in the first stage. Therefore, the solutions provided by ULRCDS, GRCDS, and GCDS are at least better than that provided by LRCDS. Therefore, LRCDS, ULRCDS, GRCDS and GCDS can perform as well as or better than Wu CDS in all cases. □

As mentioned above, we can use the GCDS algorithm in a proactive link state routing protocol if it works properly without support for guaranteed delivery from the MAC layer protocol. We prove that the link state protocol using the GCDS algorithm introduced in Section 4.3 can function properly without the hypothesis of the guaranteed delivery.

Assertion: The GCDS algorithm can work properly without the hypothesis of guar-

anteed delivery by the MAC layer.

Proof: Assume every node in a CDS periodically broadcasts the entire local topology database. Links that are not refreshed within a certain time expire, i.e. they time out. Define a CDS link as a link with at least one CDS end point.

First, when a link-down event occurs, assume that the transmission of the corresponding link state record from a CDS node to its one or more neighbors is corrupted due to a transmission collision, other interference, or is lost due to a buffer overflow. Since link state information expires due to the time out property and the CDS nodes also periodically rebroadcast, eventually this link down event is propagated to all of its neighbors.

Now we concentrate on the case of link-up events in the network. Generally speaking, if a node remains as a CDS node all of the time, it periodically broadcasts its link state records. So even if some transmissions of control messages are not successful, its neighbors may still receive valid control messages by the next periodic broadcast from this CDS node as long as this CDS node is still in the CDS set. Now we need to show that the link state protocol using the GCDS algorithm can still work properly if this CDS node changes to a non-CDS node after an invalid transmission. We need to prove that in this case, the link-up control message can still be propagated to its neighbors properly.

Figure 4.4 illustrates a network formed by two subnets at time T_0 . At this time, all nodes have identical global topology information and there is one set of CDS nodes in the network. In the example in Figure 4.4, there is a CDS link formed by a CDS node, A, and one of its neighbor nodes, B. Node B may be a CDS or non-CDS node. Above all, if the link from A to B is the only CDS link that connects to node B or Subnet II, A should not change to a non-CDS node to maintain the properties of a CDS. Therefore, we assume that in this network there are at least two CDS links from Subnet I to Subnet II, including the one associated with node A.

Assume that a transmission error occurs when node A broadcasts a link-up event generated in Subnet I. We also assume that before A starts the next broadcast, another

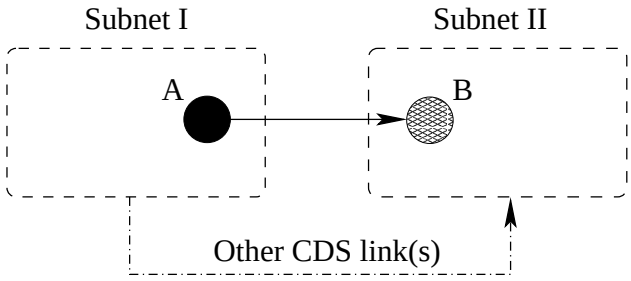


Figure 4.4: Example of a network with the GCDS algorithm.

control message arrives at node A. Node A changes to a non-CDS node based on the new topology. This implies that there is at least one other CDS link from Subnet I to Subnet II. Therefore, although node A changes to a non-CDS node, we still have another path to propagate the corresponding link state information to nodes in Subnet II. So, eventually, the transmission error is recovered by the other CDS link from Subnet I to Subnet II.

Thus, we proved that the link state routing protocol using the GCDS algorithm works without assuming that the underlying MAC layer protocol provides guaranteed delivery. In other words, we can use the GCDS algorithm in the link state routing protocol independent of the underlying MAC layer protocols. □

Note that although we proved that the GCDS algorithm can function properly without guaranteed delivery, the GCDS algorithm may suffer longer convergence time than the local algorithms when there are control message transmission errors.

4.2 Comparison of Approximation Algorithms for Minimal CDS

This section presents and briefly discusses simulation results for approximation algorithms to find a CDS. Section 4.2.1 describes the simulation model, Section 4.2.2 presents simulation results, and Section 4.2.3 summarizes the comparisons.

4.2.1 Simulation Environment

Simulations are done with C++ programs running under the Linux operating system. The simulation conditions are similar to those used by Wu and Li [75]. Nodes were randomly generated in a 100×100 square unit area. Radio range is used to decide whether two nodes are connected. Ranges of 25, 50, and 75 units were used. A depth-first search algorithm was used to ensure that all topologies used were connected networks. One-thousand connected networks were generated for each set of parameters. Since Das CDS I and Das CDS II algorithms perform similarly [73, 74, 75], we implemented Das CDS I, together with Wu CDS, LRCDS, GRCDS, ULRCDS, and GRCDS. Note that in simulations of LRCDS and ULRCDS, a node in $N_i[n]$ randomly picks a sender from its neighboring nodes in $N_i[n-1]$.

4.2.2 Simulation Results

The average size of the CDS versus the number of nodes in the network are presented in Figures 4.5, 4.6, and 4.7 (According to the reason discussed in Section 4.1, we do not show the variance of simulation results in these figures). GCDS always gave the best (smallest) size in these three scenarios. It is seen in our results that GCDS, GRCDS, LRCDS and ULRCDS all gave better performance than Das CDS I and Wu CDS in terms of the average size of the CDS when the radio range is 25 and 50. Figure 4.7 also shows that in the case of a dense network with radio range of 75, Das CDS I is the best when compared with Wu CDS, LRCDS and GRCDS. A possible reason is that nodes have large degrees in a dense network and Das CDS I favors nodes with the largest effective degree. (The effective degree is the number of non-CDS neighbors.) Nodes with higher degree have a higher probability to be chosen to form a solution that is close to optimal. This is the reason that we modify LRCDS to form ULRCDS and GCDS, which also consider degree in selecting CDS nodes. GCDS turns out to be the best algorithm among all the algorithms

for building a minimal CDS. It is also shown in the graphs that LRCDS, GRCDS, ULRCDS, and GCDS always outperform Wu CDS. This is consistent with the proof in Section 4.1.7.

Note that the data in [75] for the Das CDS I algorithm do not match ours when the radio range is 50. In fact, the data in [75] for the Das CDS I algorithm are the same when radio range equals to 25 and 50, which is not reasonable. Our simulation has validation procedures to validate every output CDS by checking the coverage property and connectivity for each solution. Our data show that when the radio range increases from 25 to 50, the average size of the CDS decreases. Therefore, this provides some verifications for our results.

4.2.3 Summary of Comparisons

We presented four connected dominating set approximation algorithms to find a minimal CDS for a wireless ad hoc network: the global ripple CDS algorithm, the local ripple CDS algorithm, the updated local ripple CDS algorithm, and the global CDS algorithm. Analysis and simulation experiments were used to compare these four algorithms with three other algorithms reported in the literature. The time complexity of ULRCDS was the same as Wu's algorithm [75] and LRCDS. GCDS and GRCDS had larger time complexity compared to other local algorithms. GCDS had the best performance among the algorithms in terms of the average size of the CDS. OSPF-MCDS protocol based on the MCDS algorithms discussed in this section is introduced in the following section.

4.3 Extending OSPF using Minimal CDS Algorithms (OSPF-MCDS)

As stated in the previous sections, a minimal CDS may reduce overhead compared with blind broadcast. Therefore, using a minimal CDS helps to reduce broadcast overhead.

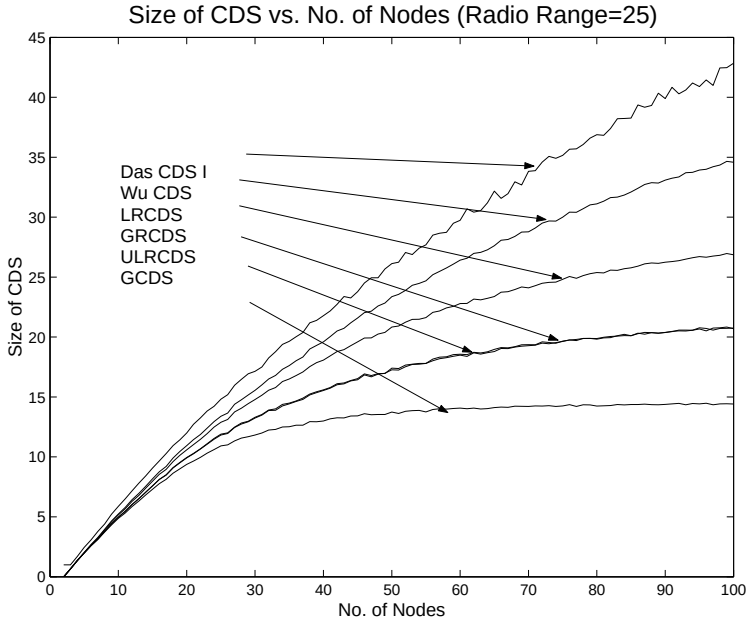


Figure 4.5: Size of CDS for radio range of 25.

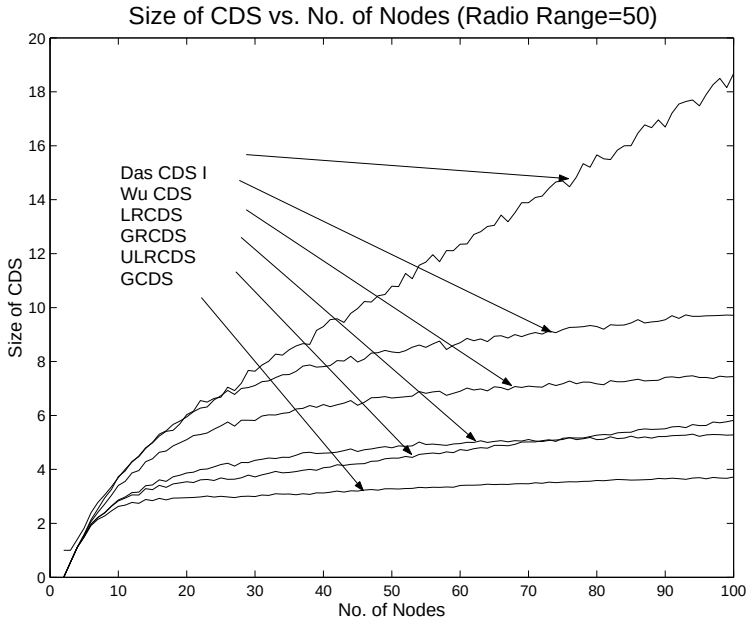


Figure 4.6: Size of CDS for radio range of 50.

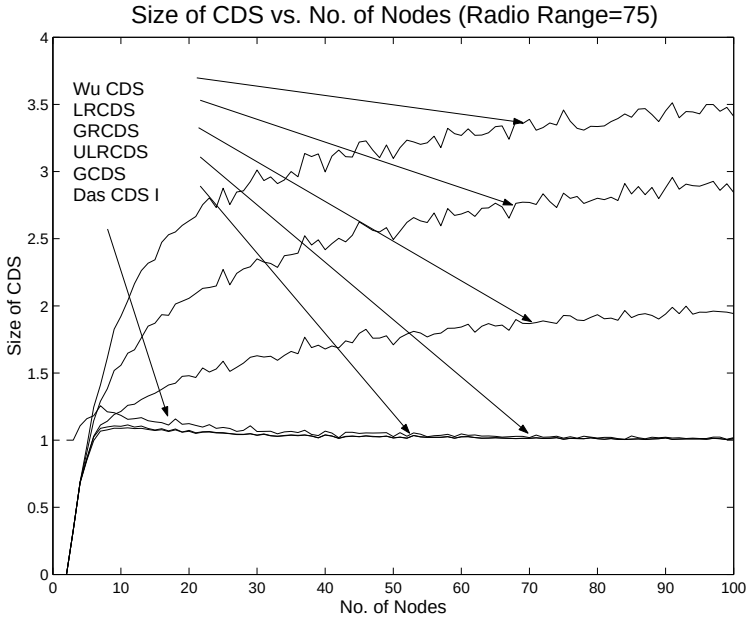


Figure 4.7: Size of CDS for radio range of 75.

This allows us to extend the traditional OSPF protocol for MANET by replacing the designated routers with a minimal CDS. We call this new link state protocol Open Shortest Path First using Minimal Connected Dominating Set (OSPF-MCDS). This new proactive link state protocol maintains full topology information in distributed nodes and also supports optimum path routing like OSPF. This protocol has low control overhead since it uses CDSs to broadcast link state control messages. The details of this protocol are described as follows.

The protocol is formed by three sub-protocols: the hello protocol, the CDS discovery protocol, and the link state synchronization protocol. Section 4.3.1 briefly discusses the trade-offs of using different MCDS algorithms for OSPF-MCDS, Section 4.3.2 describes control messages used by OSPF-MCDS, and Section 4.3.3 provides the specifications of the protocol. A full draft specification of OSPF-MCDS is in the Appendix.

4.3.1 Trade-offs using Different MCDS Algorithms

Using GCDS can have smaller size of CDS. Thus, the number of re-transmissions and the number of originators of broadcast messages are minimized. However, the trade off is more computation time. Another disadvantage is that GCDS is not scalable with respect to the network size. When the quality of wireless communications is poor and some control messages are missing, GCDS algorithm can suffer from network topology inconsistency problem. In other words, nodes selected in distributed nodes cannot form a MCDS for a certain period of time, which reduces the throughput and increases end-to-end delay from users' perspective.

The advantages of using ULRCDS for CDS discovery include reduced overhead and low computation time since the time complexity is $O(\Delta^2)$. ULRCDS is a local distributed algorithm. Therefore, when a link changes state, we do not need to do a recalculation at all nodes, as is required in global algorithms. ULRCDS can work even when the topology changes rapidly, as long as the link change rate is relatively small compared to the frequency to broadcast control messages, and transmission delay is high. In other words, it does not suffer from network topology inconsistency problems. It consumes a small amount of energy compared to CDS approximation algorithms proposed by peers due to lower time complexity and fewer control messages. The disadvantage of ULRCDS is obvious; it is not optimum. In other words, the size of CDS selected by ULRCDS may not be the smallest compared to other algorithms. The performance of ULRCDS, in terms of the size of selected CDS nodes, can be improved. For example, the size of the CDS can be reduced by checking for overlap areas with not only the sender but, also, with other neighbors.

We use a local algorithm presented in Section 4.3.3.1 for the simulation and emulation study summarized in Chapter 7. We also provide options for users to choose global or local CDS algorithms according to their MANETs. If a better MCDS algorithm is developed later, one may also replace the algorithms we used with the better algorithm.

4.3.2 Format of Control Messages

Periodic broadcast HELLO messages are used to detect neighbors. Figure 4.8 shows the format for HELLO messages defined by us. To simplify the descriptions in this document, we assume that every node has one interface used in a MANET. (The Appendix explains the extension for nodes with multiple interfaces.) The *Node ID* in the HELLO message is the IP address of that node. If an expected HELLO message does not arrive from a neighbor node within an idle interval time, this neighbor is considered to be lost. If a new HELLO message is heard by node i, the next HELLO packet from node i contains the new neighbor's ID. A two-way connection is up if a node receives a new HELLO message that contains its own ID in the recently heard node list.

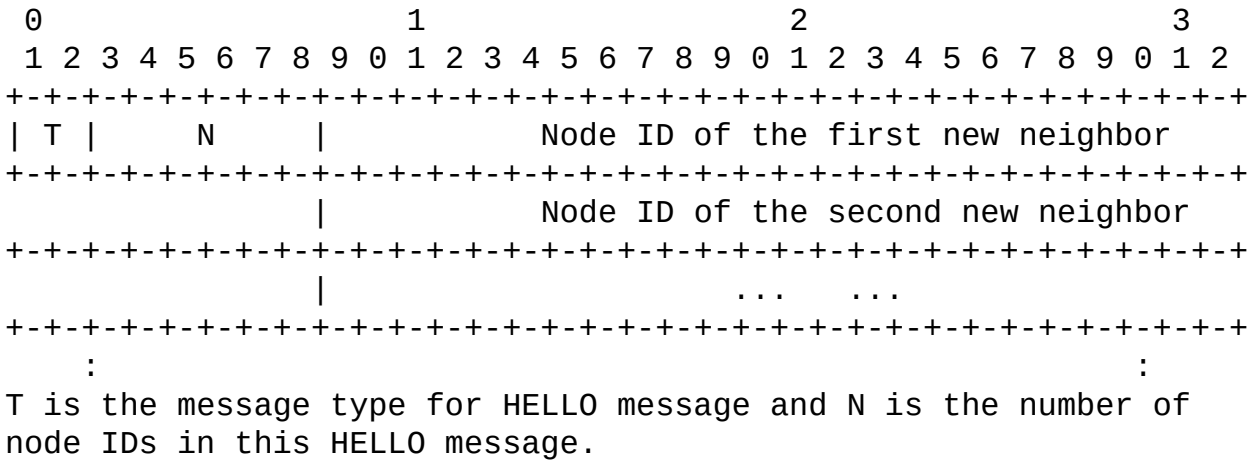


Figure 4.8: Format of a HELLO message.

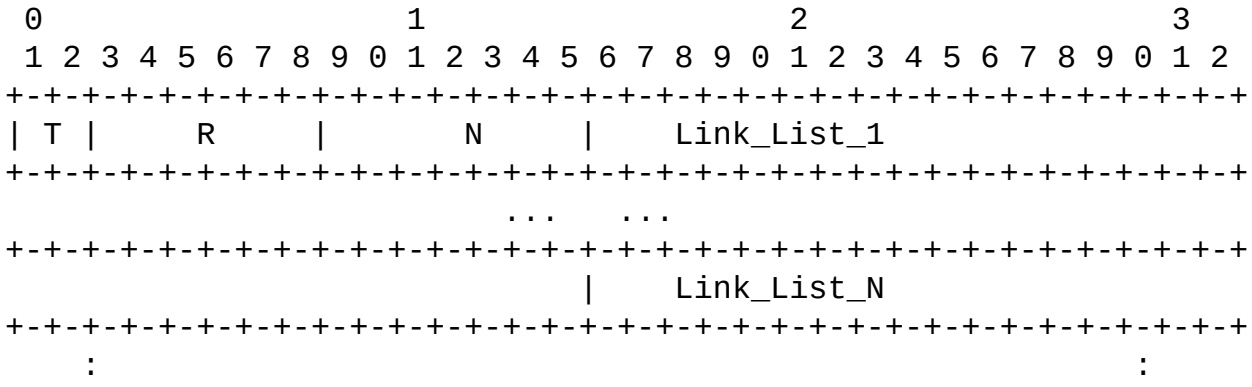
Unlike some other protocols that use CDS as default gateway routers for data packets [74, 75] and broadcast the lists of attached nodes, the selected CDS nodes in OSPF-MCDS are only responsible for rebroadcast of link state messages. Figure 4.9 shows one control message format that describes link state information. Messages of this type are broadcast by nodes. The format for a link list in this control message is shown in Figure 4.10. All nodes periodically broadcast part of their neighbor lists. Note that only the

node with the larger ID will include the neighbor node in its broadcast message. Therefore, there is only one node that broadcasts information about a neighboring link. A CDS node also inserts link state entries for non-neighboring links in its local database into the broadcast control message. When a CDS node receives a broadcast message sent by its neighbor for the first time, it processes and then re-broadcasts this packet. A node deciedes its role as a MCDS or non-MCSD right before it generates its own periodic link state messages. Before a node adds a link in a periodic link state message, it increases the sequence numbers for that links in its local database. The sequence number for links can be used to decide whether a received link state information is new or old.

When a MANET node has multiple interfaces which are not in the same subnets, i.e. this node connects to multiple subnets such as other MANETs or wired subnets, another control message is used to declare these connections. This message is illustrated in Figure 4.11 and the format of a description entry contained in this message is specified in Figure 4.12.

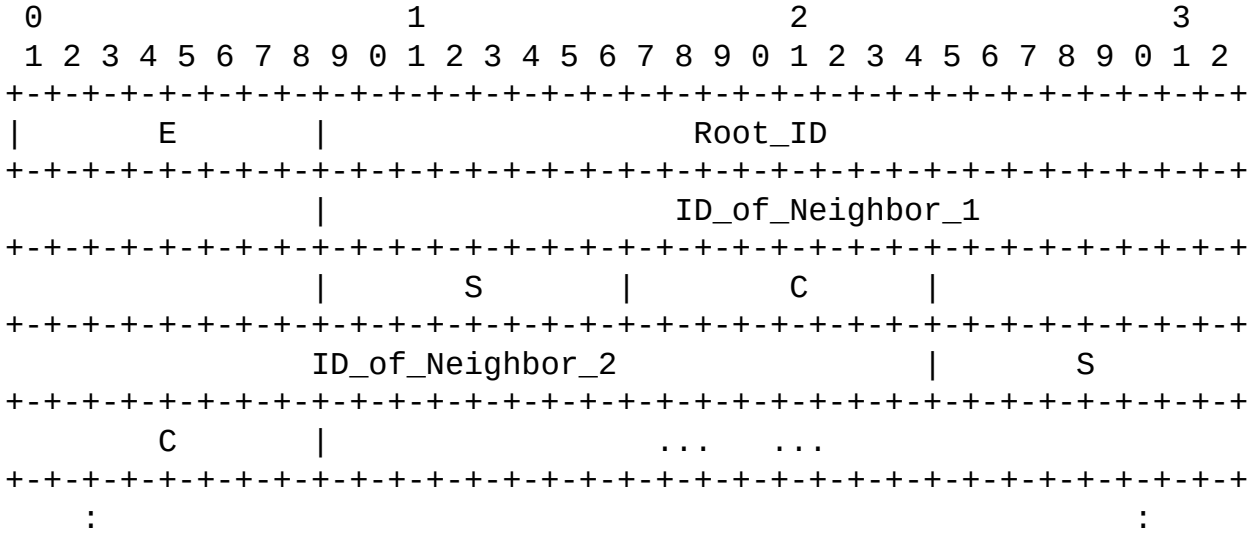
There is another control message that describes link up or down event, which is described in Figure 4.13. When a link goes down, both end nodes send out a link down message using this format. If a new link is up, the node with larger node ID sets the sequence number for this edge and sends out a link-up message using this format. Note that this message has a cost field. Therefore, it can be used to update link cost for an existing link. The definition of link cost can be hop count, traffic load, or energy consumption. Our model for OSPF-MCDS simulation and emulation discussed in Chapter 7 use hop count as the link cost.

All nodes can have the full copy of the topology since they can receive all link state information propagated via CDS nodes. Thus, we can use a shortest path algorithm, such as Dijkstra's algorithm used in the Open Shortest Path First (OSPF) protocol [12], to compute optimum routes for each node pair. Like OLSR and TBRPF, periodic broadcasts of the neighbor lists from CDS nodes can be used to handle unreliable transmissions.



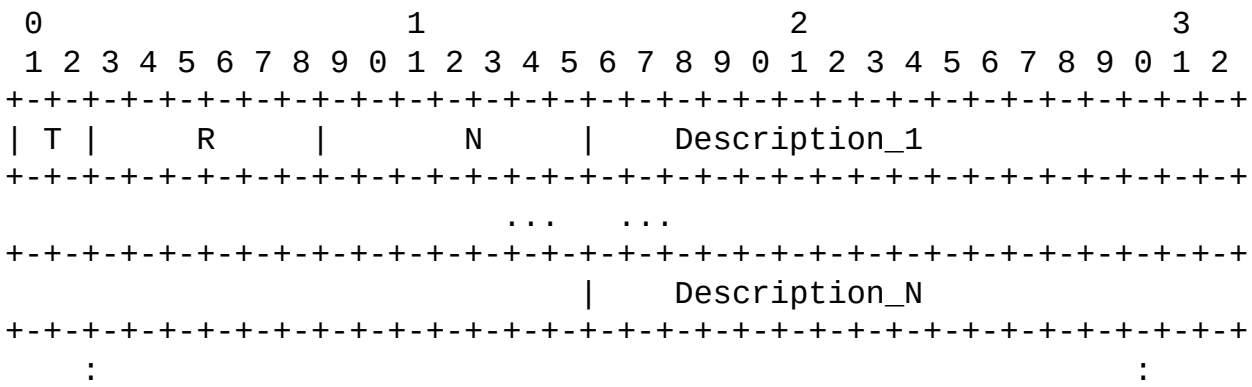
T is the message type for this control message, R is reserved field for future use, and N is the number of link lists in this message.

Figure 4.9: Format of a Link Database Description (LDD) message.



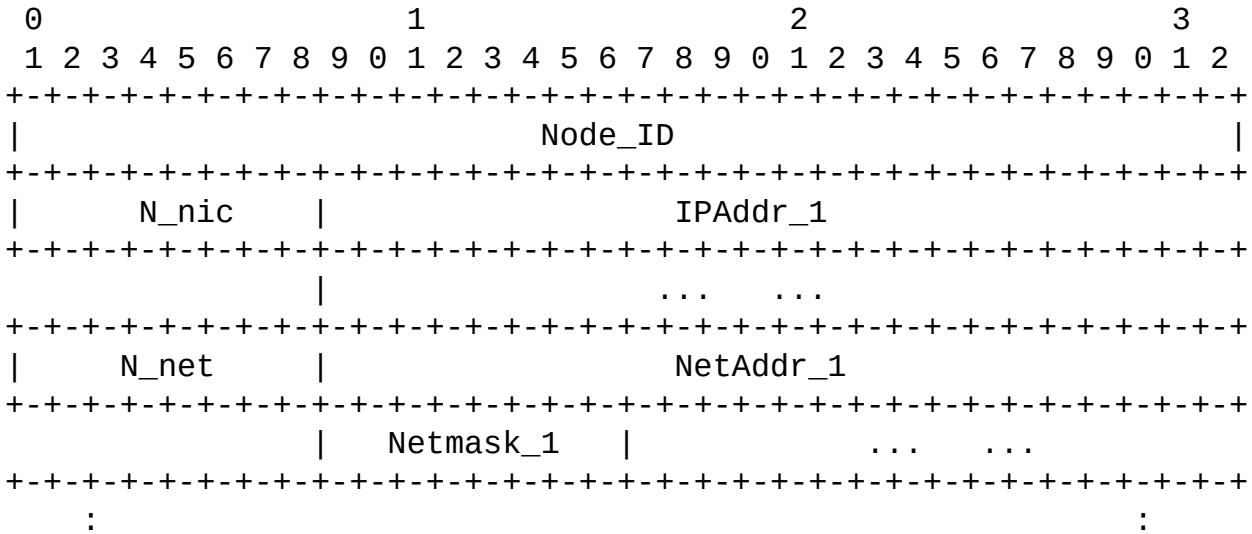
E is the number of edges in this list, Root_ID is the root of this list, ID_of_Neighbor_i is the i-th neighbor of this Root_ID node, S is the sequence number of this edge, and C is the cost of this edge.

Figure 4.10: Format of a Link List in a LDD message.



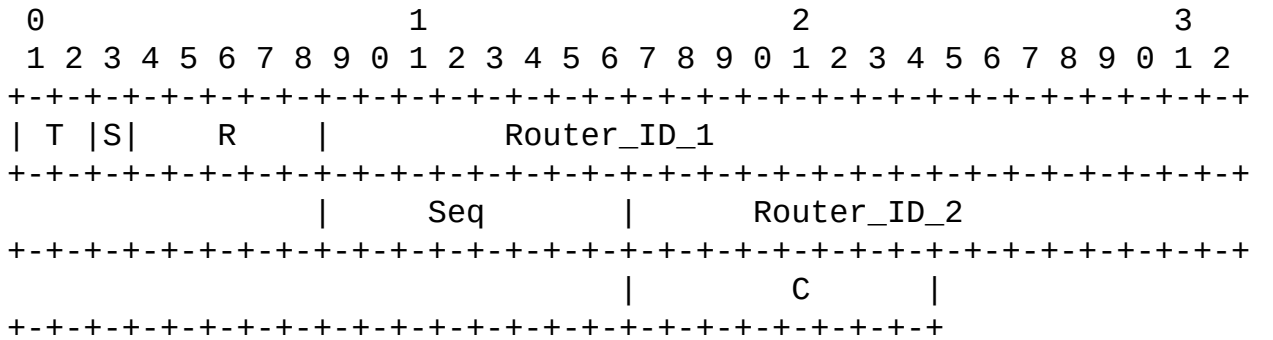
T is the message type for this control message, R is reserved field for future use, and N is the number of interface and subnet prefix descriptions in this message.

Figure 4.11: Format of an Interface and Prefix Description (IPD) message.



Node_ID is the ID of that node, N_nic is the number of interfaces associated with this node, IPAddr_i is the corresponding IP address, N_net is the number of subnets associated with this node, NetAddr_i and Netmask_i define the netmask for that network address.

Figure 4.12: Format of an entry in IPD messages.



T is the message type for this control message, S is the sub type field which is LINK_UP or LINK_DN, R is reserved field for future use, Router_ID_1 and Router_ID_2 are the IDs of two end nodes, Seq is the sequence number of this edge, and C is the cost of this link.

Figure 4.13: Format of a Link State Description (LSD) message.

We reduce the number of re-transmissions and the size of broadcast messages using the CDS algorithms and partial neighbor lists, respectively. Therefore, the routing protocol using the CDS algorithms can be applied to large networks compared with routing protocols using blind broadcasts. However, if the network is large enough and scalability becomes an issue, a clustering technique should be used, similar to the concept of an area used by OSPF [12]. The next section introduces the specifications of OSPF-MCDS.

4.3.3 Specification of OSPF-MCDS

In this section, we briefly present the specification of the OSPF-MCDS protocol, including the generation of the MCDS, the HELLO protocol, how link-up or link-down events are handled, how the periodic link state control message is generated, and how control messages are sent. Please refer to the appendix for a full description of the protocol.

4.3.3.1 Generation of MCDS

There are two candidate algorithms used in OSPF-MCDS, one is global and the other is local. The global algorithm is the GCDS algorithm presented in Section 4.1.5 and runs on a copy of the full topology. The local algorithm is developed and based on the GCDS algorithm and the ULRCDS algorithm presented in Section 4.1.4, which requires three-hop neighbor set information. The algorithm is described as follows.

Algorithm 4.3.1 *Input: The link state database within the 3-hop range of node c, say Link-Database(c).*

Output: Node c is a MCDS or non-MCDS node.

Initialization: For every node in LinkDatabase(c), initialize its state as a non-MCDS node. Define node set M = {c, neighbors of c}.

Notes:

1. If the network is fully connected (which can be examined by comparing the sum of degrees and the maximum possible number of links in the network times two), all nodes are non-MCDS.

2. In the case of a tie, the node with larger ID breaks the tie.

Stage I.

1. Randomly picks up a node in M, say i, if there is no node left in M, go to STEP 8. Otherwise, update M = M - {i}.

2. Construct a node set Nbr(i) = {node i's neighbors}. Mark all nodes in this set as un-visited nodes.

3. Find an un-visited node in Nbr(i), say node k. If there is no such node, go to STEP 1. Otherwise, Nbr(i) = Nbr(i) - {k}. Let node set N = {k} and mark node k as a visited node. Initiate the existing node set $N_{exit} = \phi$.

4. Find an un-visited node in Nbr(i), say node j, s.t. N ∩ {node j's neighbors} ≠ 0. Mark node j as visited and Nbr(i) = Nbr(i) - {j}.

5. If there is no such node j left in Nbr(i), check the size of N_{exit} and Nbr(i). If $|N_{exit}| = 0$ and $|Nbri| \neq 0$, node i is a primary MCDS node and go to STEP 1. Else if $|N_{exit}| \neq 0$ and there is a node k ∈ $|N_{exit}|$ s.t. k covers all i's neighbors and the degree of k is larger than the degree of i, node i becomes a non MCDS node; go to STEP 1. Otherwise, node i is a potential MCDS node; go to STEP 3.

6. N = N ∪ {j}.

7. If j is adjacent to node i, go to STEP 4. Otherwise, $N_{exit} = N_{exit} \cup P\{j\}$, where the node set P{j} = N ∩ {node j's neighbors}, and then go to STEP 4.

Stage II.

8. If node c is a primary MCDS or non-MCDS node, output the result and return.

9. Let N_{mcds_nbr} = {primary MCDS nodes in node c's neighbors} ∪ {potential MCDS nodes in node c's neighbors which have larger node degree than node c}. If N_{mcds_nbr} covers all node c's neighbor nodes and are connected only by nodes in N_{mcds_nbr}, c is set to be a non-MCDS node. Output the result and return.

Basically, the first stage of Algorithm 4.3.1 is the same as the GCDS algorithm except it only examines nodes in the two-hop range. The second stage is the same as the ULRCDS algorithm. Algorithm 4.3.1 has the following properties. Note that we do not consider fully-connected networks since there is no node selected as a MCDS node in such networks.

Lemma 4.1 *In Stage I in Algorithm 4.3.1, the node type for a node, say* i*, is examined at node* i *and all of* i*'s neighbors. The distributed results at all these nodes are identical. In other words, all these nodes assign the same node type to node* i *in Stage I.*

Proof: For any node investigated in Stage I of Algorithm 4.3.1, say node *i*, the input is its two-hop neighbors. We assume that this information is known to node *i* and all its neighboring nodes, via exchanging of control messages between neighbors. Therefore, the results are identical at distributed nodes.□

Lemma 4.2 *In Stage I of Algorithm 4.3.1 if a node is a non-MCDS node, one of its exit nodes that covers all its neighbors has to be selected as a MCDS node.*

Proof: According to the description in Stage I of Algorithm 4.3.1, a node, say *i*, is a non-MCDS node if and only if it has one open neighbor set, say node set O_i, and there is at least one node in the corresponding exit node set, say node set E_i, which covers all of i's neighbors and has larger node degree than node *i* (in case of a tie, that node has a larger node ID than *i*).

Let *e* be any node in E_i that covers all node *i*'s neighbors and node set *E* = {*all possible such e nodes*}. Assume node *j* is in *E* and has the largest degree in *E* (in case of a tie, we assume *j* has the largest node ID than other nodes with the largest degree). In the next paragraph, we prove that node *j* has to be selected as a MCDS node.

We prove the statement by contradiction. Assume node *j* is selected as a non-MCDS node. Similar to node *i*, node *j* has one open neighbor set, say node set O_j, and there is at least one node in the corresponding exit node set, say node set E_j, which covers all of j's neighbors and has larger node degree than node j (in case of a tie, that node has a larger node ID than j). Therefore, let node k be the node in E_j which covers all j's neighbors and has larger degree than j (in case of a tie, k has a larger node ID than j). Here, we have {neighbors of i} $\subset$ {neighbors of j} $\subset$ {neighbors of k}. Therefore, *k* is also in *E*. We assumed that node *j* has the largest degree, but now we have another node *k* in *E* which has larger node degree than *j*, or in case of a tie, has a larger node ID than *j*. This is a contradiction. Therefore, we prove that node *j*, which is one exit node for node *i* and covers all of node *i*'s neighbors, has to be selected as a MCDS node if node *i* is selected as

a non-MCDS node by Stage I in Algorithm 4.3.1. □

Lemma 4.3 *The node set selected, say node set* S, *by Stage I of Algorithm 4.3.1 in the network, if not empty, is a CDS.*

Proof: Firstly, we prove that every node is either in *S* or has at least one neighbor in *S*. According to Lemma 4.2, a node is a MCDS node or at least one of its neighbors is selected as a MCDS node. Therefore, the coverage property of the selected node set *S* is proved.

Secondly, we prove the connectivity feature, also by contradiction. Assume two selected node *i* and *j* are not connected in *S*. Let $(i, p_1, p_2, ..., p_m, j)$ be one of the shortest paths connecting node i and j in the network, in which the number of nodes in S is the maximum. In other words, the number of nodes in S in path $(i, p_1, p_2, ..., p_m, j)$, say Max_S, is the maximum possible value and $Max\_S \leq m + 2$. Since nodes i and j are disconnected in S, there is at least one node p_i in $p_n(n = 1, 2, ..., m)$ which is not in S. Otherwise, nodes i and j are connected in S and $Max\_S = m+2$, which is contradict with the disconnected assumption.

According to Lemma 2, node p_i has one MCDS neighbor, say p_{mcds}, which covers all its neighbors and is in S. Node p_{mcds} cannot be p_{i-1} or p_{i+1}, because, otherwise, we can reduce the length of the shortest path $(i, p_1, p_2, ..., p_m, j)$ by removing node p_i since p_{i-1} and p_{i+1} are connected. Because nodes p_{mcds}, p_{i-1} and p_{i+1} are all in p_i's neighbor set and p_{mcds} covers all p_i's neighbors including p_{i-1} and p_{i+1}, we can use p_{mcds} to replace node p_i in path $(i, p_1, p_2, ..., p_m, j)$ without increasing the path length. Thus, we construct another shortest path between nodes i and j, $(i, p_1, p_2, ..., p_{i-1}, p_{mcds}, p_{i+1}, ..., p_m, j)$, which has larger Max_S. This brings up a contradiction because we assumed the path $(i, p_1, p_2, ..., p_m, j)$ has the maximum number of nodes in S. Therefore, we proved the node set selected by Stage I in Algorithm 4.3.1 is a connected node set.

In summary, the node set selected by Stage I in Algorithm 4.3.1 covers all nodes in the network and is connected by nodes in it. Therefore, the node set selected by Stage I in Algorithm 4.3.1 is a CDS. □

Lemma 4.4 *There is a shortest path between any two nodes in the node set selected by Stage I in Algorithm 4.3.1 in the network contains only nodes in the node set selected by Stage I in Algorithm 4.3.1*

Proof: According to the discussions in the proof for the connectivity property of Lemma 4.3, for any two nodes i and j is S, they are connected via at least one shortest path in the network in which all nodes are in S. Therefore, we prove this lemma□

Lemma 4.5 *The node set selected by Stage I in Algorithm 4.3.1 is still a CDS after Stage II in Algorithm 4.3.1.*

Proof: For a potential node, say node *i*, let node set $N_1(i)$ be node *i*'s primary MCDS neighbors, node set $N_2(i)$ denote node *i*'s potential MCDS neighbors with larger node IDs, node set $N_3(i)$ represent node *i*'s potential MCDS neighbors with smaller node ids, and $N_4(i)$ be the node set containing node *i*'s non MCDS neighbors. If node i is removed according to Stage II in Algorithm 4.3.1, node set $N_3(i) \bigcup N_4(i)$ is covered by $N_1(i) \bigcup N_2(i)$.

We prove the consistency of coverage property by contradiction. Assume that there is a node, say *i*, such that it was covered by a potential MCDS nodes m_i and now is not covered by any MCDS nodes because nodes m_i are changed to non-MCDS nodes by Stage II in Algorithm 4.3.1. Then, node i should be covered by $\{N_1(m_i) \bigcup N_2(m_i)\}$. However, since node i is not covered by any MCDS node according to the assumption, the only nodes, say nodes n_i, that can cover node i have to be in $\{N_2(m_i)\}$ and are also removed by Stage II in Algorithm 4.3.1 when nodes n_i are examined. As we know, the removal of nodes n_i requires that $\bigcup_{all\ possible\ n_i}\{N_1(n_i) \bigcup N_2(n_i)\}$ covers all of node n_i's neighbors which include node i. Therefore, the discussion on nodes in $N_2(m_i)$ also applies to nodes

in all possible $N_2(n_i)$. The removal procedure continues and cannot be terminated because, otherwise, node i is covered by the last removal's MCDS neighbor set and this is conflict with our assumption. Therefore, Stage II in Algorithm 4.3.1 has to remove infinite potential MCDS nodes to keep our assumption valid. However, we know that the node set selected by Stage I in Algorithm 4.3.1 is finite since it is no larger than the network size. Therefore, this brings up a contradiction and we proved that every node is either a MCDS node or is covered by at least one MCDS node.

We can use the same argument to prove that the connectivity remains in the final node set after Stage II in Algorithm 4.3.1. To summarize, the final node set after Stage II in Algorithm 4.3.1 is still a CDS. □

Theorem 4.1 *The node set selected by Algorithm 4.3.1 is a CDS.*

Proof: According to Lemma 4.3, the node set selected by Stage I in Algorithm 4.3.1 is a CDS. Lemma 4.5 shows that the selected node set is still a CDS after the second stage in Algorithm 4.3.1. Therefore, the node set selected by Algorithm 4.3.1 is a CDS. □

4.3.3.2 Exchange of HELLO message

Nodes periodically broadcast HELLO messages to detect new neighbors and a time-out scheme is used to find expired neighbors if there is no HELLO message received from a neighbor for a certain amount of time. We use differential HELLO message, which only include the IP addresses of new neighbors. If the received HELLO message does not contain the receiver's node ID, the sender's node ID is included in the receiver's next HELLO message. Otherwise, the receiver detects a new bi-directional link. In this case, if the sender has a larger node ID, the sender's IP address will be included in the next HELLO message. If the sender has a smaller node ID, the receiver sends a Link UP LSD if there is no LDD message to be sent soon (See Section 4.3.3.3). After a two-way link is up between two neighbors, the neighbors' IP addresses are excluded from the HELLO

messages. The finite state machine for the HELLO protocol is demonstrated in Figure 4.14.

In the implementation, a counter is set for each neighbor, recording the number of missing HELLO messages. If a HELLO message is received from this neighbor, this counter is reset to zero. Each time when the center node generates a HELLO message, this counter increases by one. The neighbor expires once the counter reaches the maximum allowed value.

4.3.3.3 Handling of Link UP and Link DOWN events

Generally, when a new link comes up, the neighbor node with larger ID sets the sequence number and sends a LSD message to declare the corresponding Link UP event. When an old link goes down, both end nodes broadcast the Link DOWN event using a LSD message. Note that in the case of a link coming up, the neighbor node does not send a LSD message if there are only two nodes in its link state database. In other words, all known nodes in the network have the knowledge of this new link and there is no need to broadcast it. Similarly, in the case of a link down event, a neighboring node of that link does not send a LSD message if there is no neighbor node in its neighbor list.

Note that if a LDD periodic message is going be to sent soon (see Section 4.3.3.4 for the definition of the term "soon"), a LSD Link UP message is not generated or forwarded. Otherwise, when the GCDS algorithm is used, an MCDS node rebroadcasts any first-seen LSD messages. If the local CDS algorithm is used, an MCDS node rebroadcasts a LSD link UP message if this new link is within the three-hop range, i.e., the minimum hop counts from this MCDS node to any end node of this new link is less than or equal to three. This is used to guarantee the correct calculation results of MCDS nodes. If the LSD describes a link down event, it is forwarded by MCDS nodes even if the local CDS algorithm is used. This is used to guarantee that no one uses this broken link in its shortest path tree. In case of a link state change, the shortest paths to all destinations are recalculated.

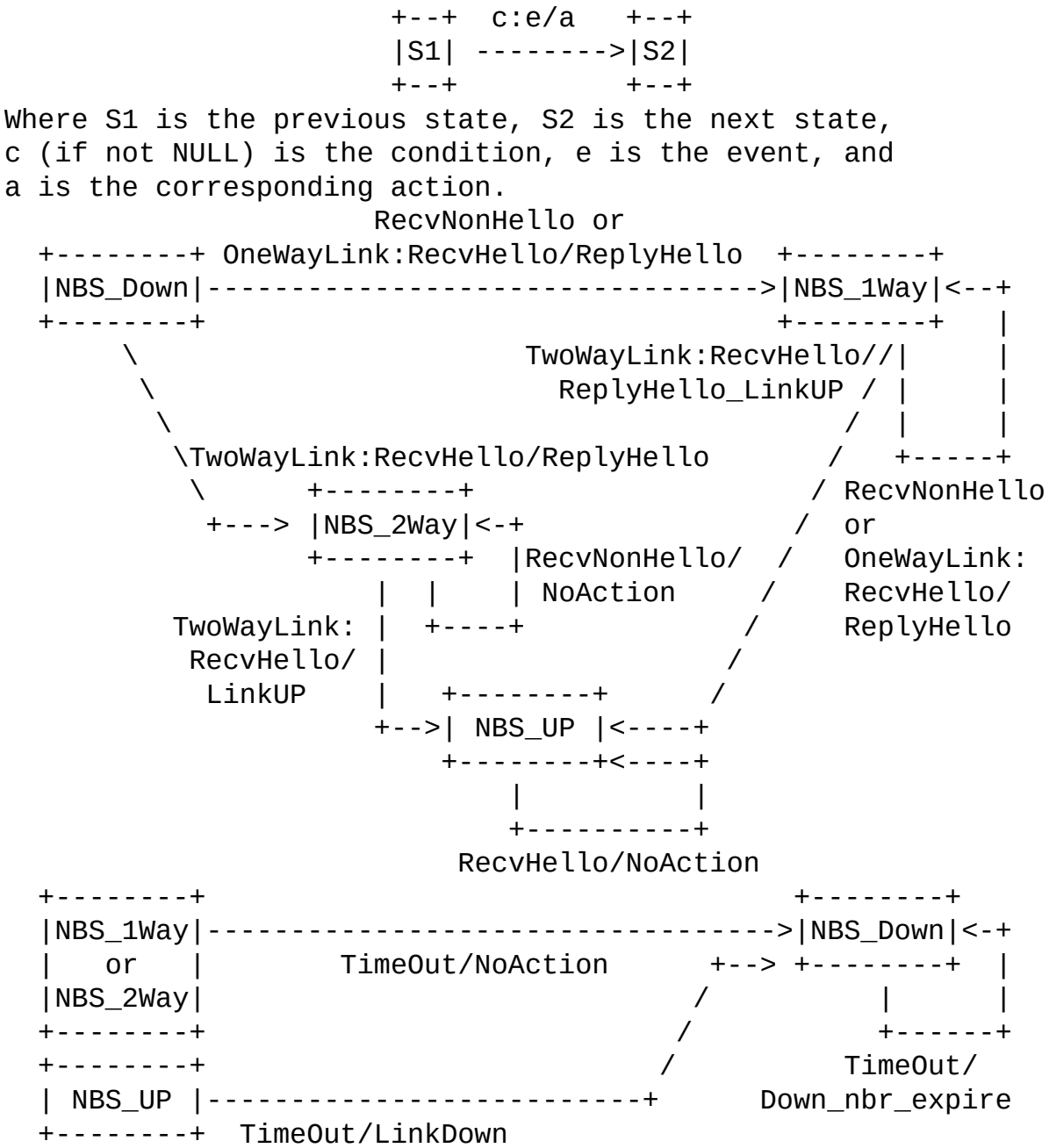


Figure 4.14: The finite state machine for the hello protocol.

When an edge is inserted into the local database via a LSD message, it is possible that the next synchronization message for this link may come later than expected since the LSD is transmitted immediately after the link state changes while the corresponding link synchronization message is propagated via MCDS nodes and has a longer delay. Therefore, the first synchronization message is expected to arrive later than ordinary next synchronization message. Thus, the link expiration for this link is set to be twice the normal expiration time. (Refer to Section 4.3.3.4 for the details of setting expiration time for edges).

4.3.3.4 Periodic Broadcast of Link State Database

There is a counter associated with the HELLO protocol. When a HELLO message is sent, the counter is increased by one. When the counter reaches a predefined maximum value, say three, the counter is reset and the node also sends a periodic broadcast of its local link state database. When the counter is one less than the predefined maximum value, the node is going to broadcast link state database "soon". In this case, there is no LSD Link UP message to be forwarded or sent by this node because the LSD Link UP message is included in the next periodic control message scheduled in a short period time. (Refer to Section 4.3.3.3 for more details.)

A non-MCDS node only broadcasts part of its neighbor list in which nodes are non-MCDS and have smaller node IDs. The sequence numbers of these edges are increased by one before the broadcast message is generated.

A MCDS node broadcasts its neighbor list and a full copy of its local link state databases using LDD messages. When the LDD is broadcast by a MCDS node, the sequence numbers of neighboring edges that are stored locally and connect to neighbors with small IDs are increased by one.

A neighbor node is not added to the LDD message if the counter for that neighbor

is one less than the maximum allowed number of missing HELLO messages. Similarly, if the counter for an edge is one less than the maximum number of allowed missing synchronization messages, this edge is excluded from the LDD message.

The sequence number used for link state descriptions has a maximum value. If such a number reaches the maximum value, it is set to zero. When a sequence number, say s_1, is larger than another, say s_2, and $|s_1 - s_2| < Maximum\_Sequence\_Number/2$, we say that s_1 is later than s_2. If $|s_1 - s_2| \geq Maximum\_Sequence\_Number/2$ and s_1 is smaller than s_2, we also say that s_1 is later than s_2.

When a node receives a LDD message, the sequence number of every link state description in this message is compared to the local copy. If a received one has a later sequence number, the local link state is updated. If there is no local copy for a link entry, this link is considered to be a new link and processed as though a LSD Link UP message is received.

Similar to the counter used for neighbors, there is also a counter for each edge. If an edge is refreshed by a LDD message, i.e., the received link state entry is more recent than the local copy, the counter is reset to zero. Each time when a node generates a periodic broadcast message, the counters for all edges are increased by one. If the counter for a link reaches the maximum allowed value, this edge expires in the local link state database. If any link expires before the generation of the next broadcast message, the MCDS is re-selected and the shortest paths to all possible destinations are re-calculated.

In the implementation, a temporary link database is used to keep tracking those recent expired link descriptions. This is because some links expired at a node may not expire at its neighboring nodes. This node may receive the link state information for those recent expired links from those neighboring nodes. The temporary link database can be used to avoid the case that those expired links are inserted into local database again. This temporary link database can be removed if, later, an improve scheme that can identify whether a link state information is out-of-date replaces the sequence number scheme we

used.

4.3.3.5 Sending Control Messages

Messages are buffered if there is no HELLO message waiting to be sent. In other words, only HELLO message can trigger the sending of control messages. This is used to combine multiple control messages into one packet and these messages can share the same IP and MAC headers which usually have large sizes compared to the control messages.

4.4 Summary

In Chapter 3, we proposed a framework and an analytical model to characterize routing protocols for MANET. Guided by the framework, we designed and developed a new routing protocol based on OSPF by replacing designated routers with MCDSs. In this chapter, we presented several approximation algorithms for finding minimal connected dominating sets. These algorithms have the subject interest from the MANET research community. Currently, research groups including Boeing, Cisco, SRI, and INRIA (France) are interested in extending OSPF for MANET routings using MCDSs and our framework and MCDS algorithm papers are cited in a recent IETF Internet draft [79].

This new routing protocol is expected to have low control overhead, maintain full topology information, and provide shortest path routing. A simulation study of different MANET routing protocols, including OSPF-MCDS, AODV, OLSR, and TBRPF, is discussed in Chapter 7. Refer to Chapter 7 for conclusions based on simulation results.

Chapter 5

Study of Node Mobility

According to the RNS framework proposed in Chapter 3, node mobility is one of the important factors that affect the operation and performance of MANET routing protocols. We studied the impact of node mobility on MANET routing protocols and summarize results in this chapter. Section 5.1 discusses our preliminary study of a widely used mobility model, the random waypoint model [80], in ns2 simulations of MANETs. This section is based on [81]. Section 5.2 proposes a quantitative study of link lifetime in a MANET. The material in this section was presented in [82]. We define and apply a neighbor stability metric, which describes the neighbor change rate, to adaptively control parameters used in MANET routing protocols in Section 5.3. Our study presented in this chapter suggests potential future research on mobility management and applications for MANET protocols.

5.1 Mobility Versus Link Stability in MANET Simulations

Simulation experiments are widely used to evaluate MANET routing protocols. Like simulations of traditional wired networks, these experiments must model the network

topology, network traffic, and the routing and other network protocols. In addition, the wireless and mobile nature of MANETs necessitates consideration of node mobility, physical layer issues, including the radio frequency channel, terrain, and antenna properties, and, perhaps, energy and battery characteristics. Node mobility, coupled with physical layer characteristics, determines the status of link connections and, hence, the network's dynamic topology. Link connectivity is an important factor, if not the most important factor, affecting the relative performance of MANET routing protocols. From the perspective of the network layer, changes in link connectivity trigger routing events such as routing failures and routing updates. These events affect the performance of a routing protocol, for example, by increasing packet delivery time or decreasing the fraction of delivered packets, and lead to routing overhead, e.g., for route discovery or route update messages. Therefore, for given physical layer assumptions, assumptions about link connectivity are critical to the significance of simulation results for MANET routing protocols. This section focuses on simulation issues related to link connectivity. We consider the rate of change of link connectivity. In other words, the number of link state changes from "up" to "down" or "down" to "up" per unit time, as the metric of interest.

Traditionally, simulation studies of MANET routing protocols have explicitly modeled mobility. At a given time, the model determines positions of nodes being simulated. The relative node locations of each pair of nodes, together with physical layer assumptions, then determine the link connectivity for that pair of nodes. Camp, et al. [80] and Liberatore [83] describe typical MANET mobility models. Mobility models can be classified as independent mobility models or group mobility models. Independent mobility models assign movement vectors independently to nodes without considering the movement of other nodes in the system. Group mobility models consider correlated movements of groups of nodes, so movement vectors are not independent among group members.

The random waypoint model is one the most commonly used mobility models for simulations of MANETs [14, 84, 85, 86, 87, 88, 89, 90, 91]. This model is implemented in

three popular simulation tools, ns2 [98], QualNet [99], and OPNET [100]. However, the random waypoint model has three shortcomings when used in simulation experiments to evaluate MANET routing protocols. First, the relationship between the random waypoint model's parameters and the rate of change of link connectivity is not well understood. Therefore, it is difficult to select mobility model parameter values to vary link connectivity properties, which exert significant influence on MANET routing protocol performance, in a controllable manner. Second, our simulation results show that the random waypoint model requires a long warm-up period for the average link-up or link-down lifetime to reach steady state. In fact, the necessary warm up time is much longer than simulation times typically used for studies reported in the literature. Finally, computational resources required to compute node positions and determine link connectivity can be relatively high, thus increasing the cost of simulation. (Of course, link connectivity can be computed once and saved for use in later runs.)

To overcome these shortcomings, we propose a link connectivity model that implicitly models mobility. The link connectivity model, in a simple form, uses a two-state Markov chain to model the link between each pair of nodes. The state of each link, UP (connected) or DOWN (disconnected), is aggregated in an $N \times N$ connectivity matrix that represents the state of an N-node MANET. This model overcomes the limitations of the random waypoint model cited above. The model can also be extended to implicitly model group mobility.

5.1.1 Analysis of the Random Waypoint Model

The random waypoint model, which has been mentioned in previous Chapters, has three parameters when used in simulations: (i) radio range, which assumes a free space propagation model to capture physical layer assumptions; (ii) maximum speed, with node velocities uniformly distributed from zero to the maximum speed; and (iii) a constant pause time. An extended random waypoint model uses the minimum speed as the fourth

definition [95]. Yoon, et. al. point out that the three-parameter definition may lead to zero speed problem and the four-parameter definition does not have this problem [96]. Therefore, we use the four-parameter definition in our study. In the random waypoint model, node locations are uniformly distributed within a two-dimensional space at the beginning of a simulation. A random destination, uniformly distributed within the two-dimensional space, is selected for each node. The node then moves toward the destination at a speed that is uniformly distributed from $[minimum\ speed, maximum\ speed]$. When a node reaches its destination, it pauses for a constant pause time. After the pause time expires, a new destination is selected and the node again moves toward the destination at a random velocity. At any given time, two nodes are connected if the distance between them is less than the radio range. Otherwise, the two nodes are disconnected.

Using this random waypoint model, the rate of change in link connectivity depends on the radio range, maximum speed, and pause time values. Camp, et al. [80], Bettstetter [92], and Bettstetter and Wagner [93] present results on the statistical properties of the random waypoint model, but do not quantify the relationship between the rate of change in link connectivity and these parameters. Therefore, it is difficult to design simulation experiments that control link connectivity properties using the random waypoint model's parameters.

It is known that the initial random distribution of mobile nodes in the random waypoint model is not representative of the manner in which nodes distribute themselves when moving [80]. Therefore, a warm-up or initialization period is required for the node mobility model to reach steady state. Camp, et al. [80] suggest a warm-up period of 1,000 seconds for the random waypoint model, but offer no justification. Warm-up periods less than 1,000 seconds are used in many simulation studies that use the random waypoint model [14, 84, 85, 86, 87, 88, 89, 90, 91].

We conducted simulation experiments specifically to study the warm-up period required for the random waypoint model to reach steady state with respect to statistical

measures for link stability. For the link between each pair of nodes, we define the link UP lifetime to be the duration of the link connection, i.e., the time from when the pair of nodes is connected (is within radio range) until the following time when the pair of nodes is disconnected (is beyond radio range). Similarly, we define the link DOWN lifetime to be the time from when a link goes from being connected to being disconnected until the link goes back to being connected. In our two-state Markov model, the UP lifetime is the time for one visit to the UP state and the DOWN lifetime is the time for one visit to the DOWN state. For the link between each pair of nodes, we define the link UP rate to be the number of times the link goes from being disconnected to being connected per unit of time. Similarly, we define the link DOWN rate to be the number of times the link goes from being connected to being disconnected per unit of time. For the two-state model, the link UP rate is the transition rate from the DOWN state to the UP state and the link DOWN rate is the transition rate from the UP to the DOWN state.

To study link connectivity with the random waypoint mobility model, we simulated 10 nodes in a 100×100 square unit area. The radio range was fixed at 20 units, the maximum node velocity was fixed at 20 units per second, and the pause time was fixed at 5 seconds. We used 0.01-second step intervals to analyze the trace file generated by ns2's *setdest*[1] command. Five independent trace files, with different seed values, were generated. For each trace file, we measured the average link UP lifetime and average link DOWN lifetime for each pair of nodes in the system. We also observed the link UP rate and link DOWN rate for the link between two selected nodes, node 0 and node 1. The results for these four metrics with 95% confidence intervals are shown in Figures 5.1, 5.2, 5.3, and 5.4. These results indicate that the link UP and link DOWN lifetimes and the link UP and DOWN rates become stationary only after a very long warm-up period. For example, the average link UP lifetime does not reach its steady-state value of 21.2 seconds until a warm-up period of more than 206,000 seconds has elapsed. This warm-up period

[1] *setdest* is an independent tool included in the CMU extension with ns2 2.1a7b and later distributions. It generates random waypoint trace files with user-defined parameters as inputs for ns2 simulations.

is much longer than the total simulation time of at most a few thousand seconds, including the warm-up period, reported in previous studies [80, 14, 84, 85, 86, 87, 88, 89, 90, 91]. The random waypoint model, at least for the parameters considered in our study, requires an excessive warm-up period to achieve steady-state behavior in link connectivity, which is required for robust simulation results (Note that all simulations using Random Waypoint model in this dissertation have long warm-up periods to ensure that the link connectivity change rates enter steady-states).

5.1.2 Two-State Markov Connectivity Model

To address the limitations of the random waypoint model and, likely, similar explicit models of node mobility, we propose a link connectivity model that implicitly models node mobility. This model has fewer parameters, thus simplifying the design of simulation experiments and can achieve steady-state values within much shorter warm-up periods.

The link connectivity model uses an $N \times N$ connectivity matrix, M, to represent the topology of a MANET with N nodes. The elements in the matrix are two-state random variables. Element M_{ij} specifies the status of link *(i, j)*, the link between node *i* and node *j*. (We assume symmetric links. Asymmetric links can be modeled using a relatively straightforward extension.) If $i \neq j$, $M_{ij} = 1$ when link *(i, j)* is in the UP state and $M_{ij} = 0$ when link *(i, j)* is in the DOWN state. We define $M_{ii} =$ the degree of node *i*, for all *i*.

The link UP lifetime and the link DOWN lifetime are random variables. We denote the mean values of the link UP lifetime and link DOWN lifetime for link *(i, j)* as $T_{ij,UP}$ and $T_{ij,DOWN}$, respectively. In our initial investigation, we assume that the link UP lifetime and link DOWN lifetime are exponentially distributed random variables. We also assume that all links have the same mean link UP lifetime and link DOWN lifetime, denoted as T_{UP} and T_{DOWN}, respectively. These two parameters, T_{UP} and T_{DOWN}, fully characterize

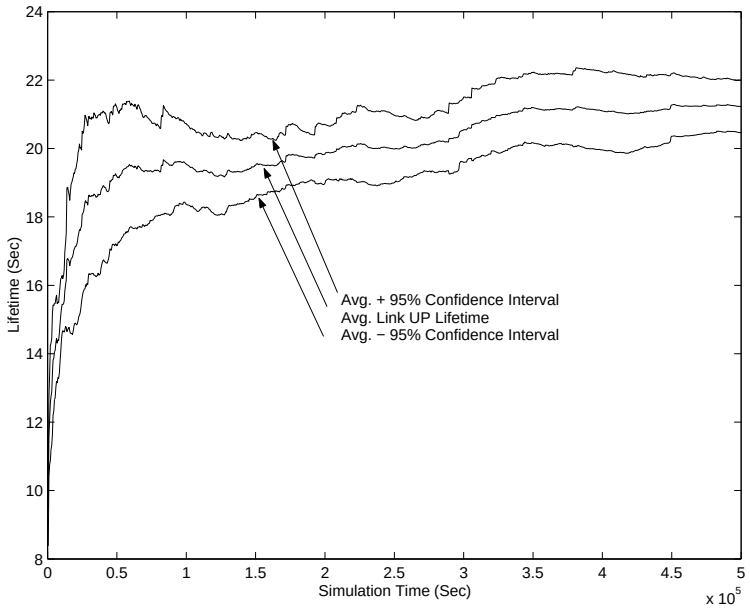


Figure 5.1: Average link UP lifetime with the random waypoint model.

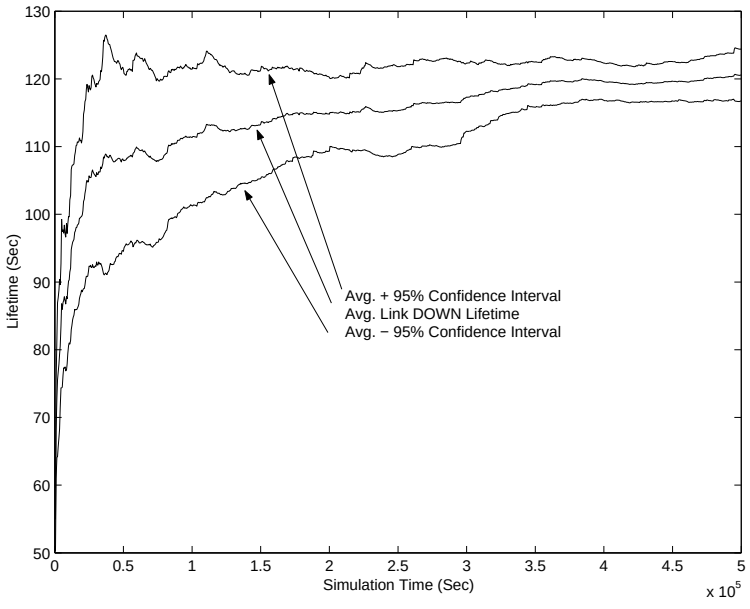


Figure 5.2: Average link DOWN lifetime with the random waypoint model.

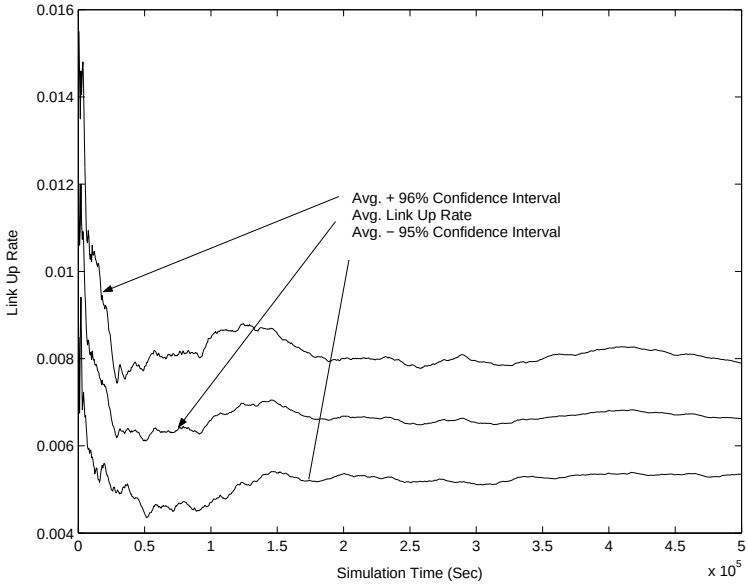


Figure 5.3: Average link UP rate with the random waypoint model.

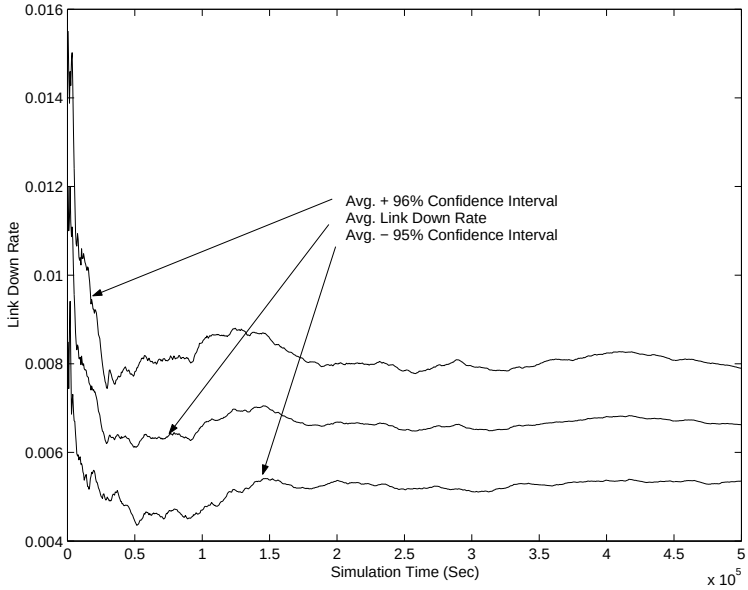


Figure 5.4: Average link DOWN rate with the random waypoint model.

this simple version of the link connectivity model.

The two-state Markov model for link connectivity is expected to have more controllable link stability characteristics and require shorter warm-up periods compared to the random waypoint model. The link connectivity model is also expected to require fewer computations and, thus, run faster than the random waypoint model. We conducted simulation experiments to examine the behavior of the link connectivity model. We assume that all links have the same mean link UP lifetime and link DOWN lifetime. To achieve the same steady-state average values for the link UP lifetime and the link DOWN lifetime shown in Figures 5.1 and 5.2 for the random waypoint model, we let T_{UP} = 21 seconds and T_{DOWN} = 120 seconds.

We are confident that the link connectivity model can be efficiently incorporated into common simulators for mobile networks. Most simulators that consider mobility, including ns2 and QualNet, read a trace file that specifies node movements or locations as a function of time. Therefore, we must modify the simulator if we want to use a link connectivity model, which specifies the connectivity state of all nodes in the network, instead of a node mobility model. This modification is possible and should not be too difficult since the simulator must, ultimately, convert the information in the trace file into connectivity information. For example, in ns2, we can integrate a link connectivity model using a slightly modified *GridKeeper* class (see [98] for details).

Results for the average link UP lifetime and the average link DOWN time using the link connectivity model, with 95% confidence intervals, are shown in Figures 5.5 and 5.6. Results for the average link UP rate and the average link DOWN rate using the link connectivity model, with 95% confidence intervals, are shown in Figures 5.7 and 5.8. The simulation results show the same steady-state values as observed for the random waypoint model. The two-state Markov model for link connectivity requires a warm-up period of only a few hundred seconds to achieve steady state values. This warm-up period is much shorter than the warm up period of over 200,000 seconds needed with the

random waypoint model for the parameters used in this experiment.

We also compared the computation time for simulations using the two-state Markov model for link connectivity and using the random waypoint model. Both cases use ns2 to simulate a 10-node network for the same period. The results in Figure 5.9 show that the computation time required by the link connectivity model is slightly less the time required by the random waypoint model. We have not made any attempt to optimize our implementation of the two-state Markov model for link connectivity or its integration with ns2. We believe that a more efficient implementation can further reduce computation time. Implementation issues are not discussed in this document since they are beyond the scope of this research.

5.1.3 Potential Extensions

As mentioned earlier, node mobility models can be classified as being independent or group mobility models [80, 83]. The simple link connectivity model described in the previous section and the random waypoint model are both independent node mobility models. In a group mobility model, nodes form groups and the rate of change of link connectivity is relatively low if two nodes are in the same group, i.e., if they are moving in a similar direction at a similar speed. We can extend our link connectivity model to implicitly model group mobility by using relatively larger values of mean link UP lifetime, $T_{ij,UP}$, for links where nodes i and j are in the same group and relatively smaller values where nodes i and j are not in the same group. We can also extend the link connectivity model to consider distributions other than exponential for link UP lifetime and link DOWN lifetime values.

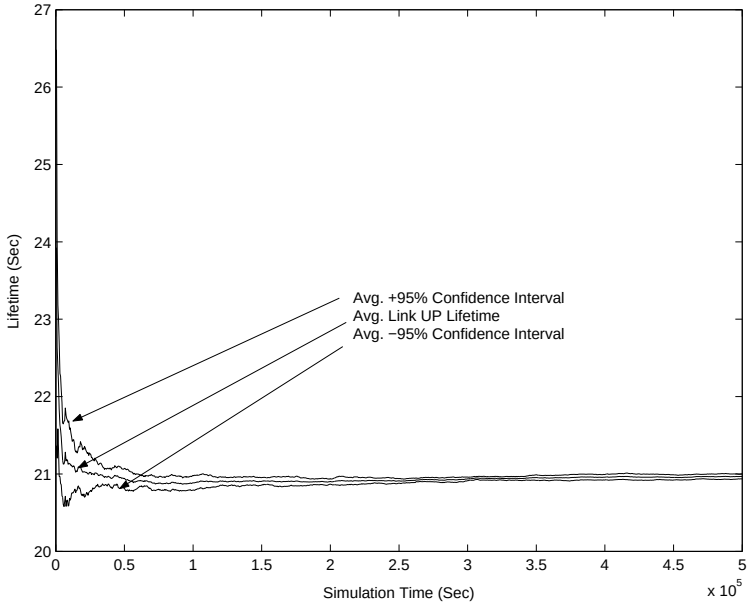


Figure 5.5: Average link UP lifetime with the link connectivity model.

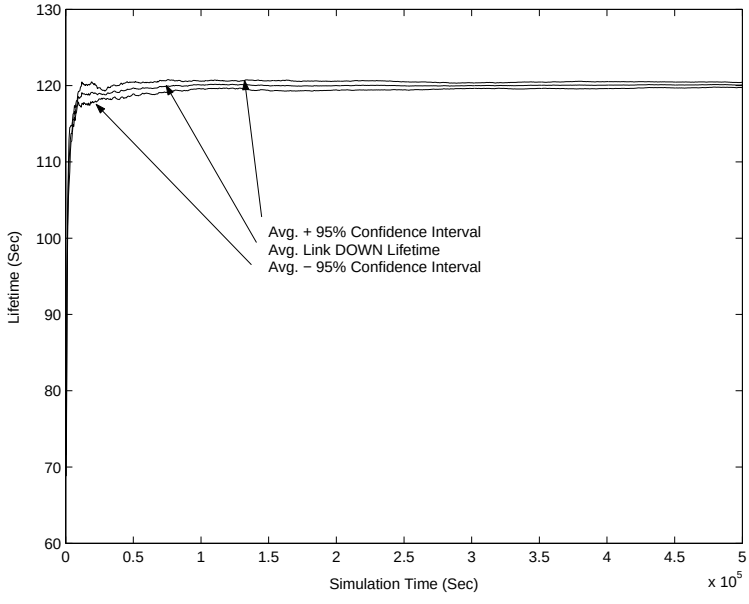


Figure 5.6: Average link DOWN lifetime with the link connectivity model.

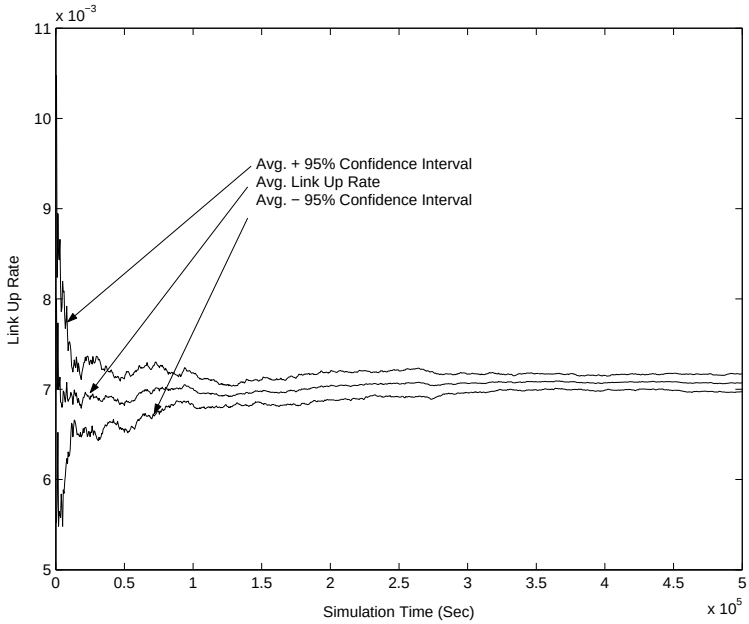


Figure 5.7: Average link UP rate with the link connectivity model.

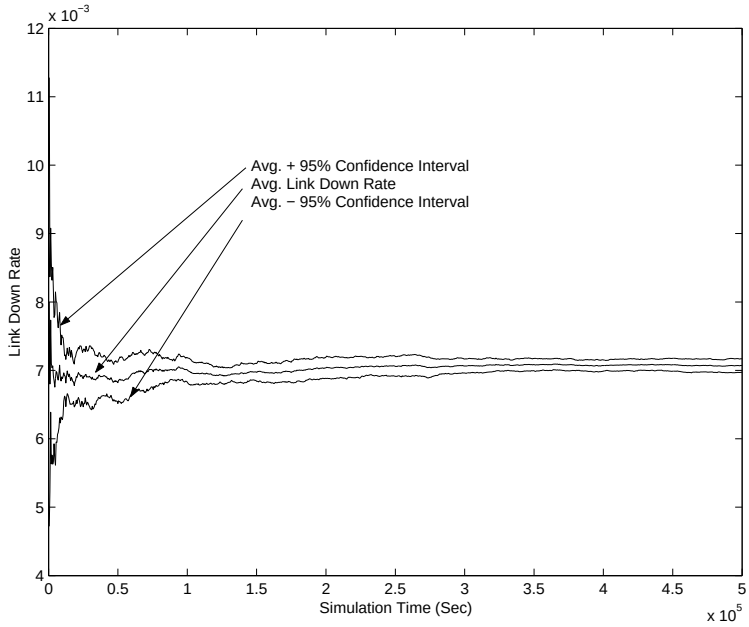


Figure 5.8: Average link DOWN rate with the link connectivity model.

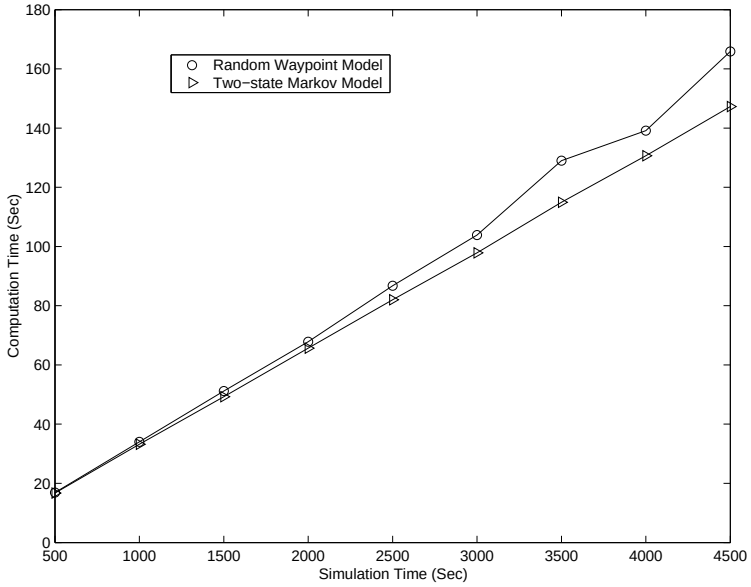


Figure 5.9: Comparison of computation time.

5.1.4 Conclusion of Study of Link Stability

This section presented a simulation study and analysis of the random waypoint node mobility model and proposed a link connectivity model based on a two-state Markov chain to address the weaknesses of the random waypoint and, presumably, other explicit node mobility models. The link connectivity model has three advantages when compared to the random waypoint model: (i) it is easier to vary link stability in a controllable manner; (ii) much shorter warm-up periods are required for link stability statistics to reach steady state; and (iii) simulations tend to require less computation time. Note that we have yet to verify the advantage of reduced computation time. The disadvantage of the link connectivity model is that it does not support high-fidelity models of mobility and physical layer characteristics. High-fidelity channel models have been developed and validated against real systems. However, there are presently no general benchmarks for mobility that would allow validation of a mobility model. So, for the near term at least, it is usually more important to subject MANET routing protocols to a broad range of mobility conditions and such conditions are most easily controlled by varying link connectivity rather

than node mobility.

Potential future work includes investigation of other mobility models such as group mobility models. We apply our results to conduct systematic evaluation of MANET routing protocols, as discussed in Chapter 7.

5.2 Quantitative Study of Link Lifetime

As we discussed in Section 5.1, for given physical layer assumptions, assumptions about link connectivity are critical to the significance of simulation results for MANET routing protocols. Thus, the design of controllable simulation experiments for MANETs requires an understanding of how to control link connectivity changes. Traditionally, simulation studies of MANET routing protocols have explicitly modeled node mobility. At a given time, the model determines positions of all nodes being simulated. The relative locations of each pair of nodes, together with physical layer assumptions, then determine the link connectivity for that pair of nodes.

There have been some prior studies of the mobility models used for MANET simulations. For example, Camp, et al. [80] present a survey of mobility models and corresponding simulation studies. Bettstetter [92] and Bettstetter and Wagner [93] present studies of the statistical properties of the random waypoint model. In Section 5.1 we present a simulation study of connectivity changes using the random waypoint model [81]. Bettstetter, et al. [94] discuss a quantitative analysis of the random waypoint model with emphasis on statistical properties, such as the estimated average epoch length and duration, the estimated direction for the next movement, and the number of cell changes. None of these papers quantify the relationships between the change in link connectivity and the parameters used in the mobility models. At present, these relationships are not well understood. Thus, it is difficult to select values for node mobility model parameters to cause predictable variations in link connectivity properties. And, since link connec-

tivity exerts significant influence on the performance of MANET routing protocols [1], it is difficult to create controllable, well-designed simulation experiments for MANETs. Since the MANET working group of the Internet Engineering Task Force (IETF) is pushing four routing protocols to the experimental RFC stage [2], comprehensive simulation studies of MANET routing protocols are becoming more important. Thus, a quantitative understanding of the relationships between link connectivity changes and node mobility models becomes increasingly important for the MANET research community.

This section presents an analytical study of the relationships between link connectivity changes and MANET mobility models. We assume that any two nodes are connected, i.e., the link is "up," when the distance between the two nodes is less than or equal to a certain radio transmission range. Here, we assume the free space propagation model, although results could be extended to consider other radio models. Two nodes are not connected, i.e., the link is "down," when the distance between these two nodes is greater than the radio range. We define the link up lifetime for a given pair of nodes to be the average time that the link between the two nodes remains in the "up" state. We define the link down lifetime for a given pair of nodes to be the average time that the link remains in the "down" state. The link life period is the sum of the link up lifetime and the link down lifetime. The link life period can be used to characterize the rate of link connectivity changes. In particular, we can define the link change rate, which is simply the inverse of link life period.

5.2.1 Analytical Study

In this section, we present an analytical study of connectivity changes in MANETs. Section 5.2.1.1 proposes a relative movement model for pairs of nodes in a MANET. A quantitative analysis of link connectivity based on this model is introduced in Section 5.2.1.2. We apply the results to a widely used mobility model, the random waypoint model in Section 5.2.1.3 and verify the conclusions using simulations in Section 5.2.2.

5.2.1.1 Relative Movement Model

Generally, nodes in a MANET move randomly and communicate with one another using wireless communications. Performance studies of MANETs typically use radio range as a parameter to limit the distance over which nodes can communicate. It is assumed that the transmitted signal is strong enough such that nodes within radio range of the transmitter can receive the data. Nodes that are beyond radio range from the transmitter cannot receive the data. Therefore, an analysis of link connectivity between a pair of nodes is equivalent to an analysis of the distance between the two nodes and the relative movement between them. Next, we introduce a model to describe the relative movement between two nodes that is the basis of our study.

When we designate one node, say node A, as a reference node, the movement of any other node, say node B, is actually a combination of node B's own movement and node A's movement. The movement of node B assuming that node A is the point of reference is known as the relative movement. We use the term "epoch" to define node B's relative movement. An epoch is the period during which node B maintains constant movement (direction and velocity) and the following period during which B pauses after the movement ends. Figure 5.10 illustrates the five possible forms of epochs for node B with respect to the reference node A for a given radio range. In the figure, the radio range is indicated by the radius of the circle with node A at its center. The area within the circle is covered by node A. There are only two possible types of starting location for node B's epochs, denoted B' and B", which represent the case that link A-B is "down" or "up" at the beginning of an epoch, respectively. When node B starts an epoch from location B', there are three possible epochs, labeled as 1, 2, and 3 in Figure 1. Similarly, when node B starts from position B", there are two possible epochs, which are labeled as 4 and 5 in the figure. Note that the ending position of node B for epochs 3 and 4 is within the radio range of node A and the next epoch starts from B". The ending position of node B for epochs 1, 2, and 5 is outside the radio range of node A and the next epoch starts from

B′. We assume that ending positions are independent of starting positions and that the average period of movement in epoch i (i = 1, 2, 3, 4, and 5) is T_i.

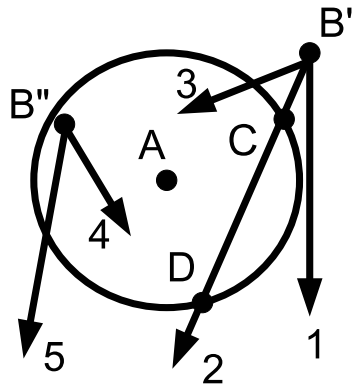


Figure 5.10: Possible forms of epochs of node B with respect to a reference node A.

Relative node movements lead to link connectivity changes. For example, link A-B in Figure 5.10 changes from the "down" state to the "up" state when node B starts from B′ and enters the coverage area of node A following an epoch of type 2 or 3. Link A-B changes from the "up" state to the "down" state if node B starts from B″ and leaves node A's coverage area following an epoch of type 5 or starts from B′ and departs the coverage area following an epoch of type 2. The connectivity state for link A-B does not change for epochs of type 1 or 4.

5.2.1.2 Analysis of the Relative Movement Model

Next, we study the connectivity relation between two nodes using the relative movement model described in Section 5.2.1.1. First, we define the average link life period. Figure 5.11 illustrates connectivity state changes between nodes A and B. "ON" and "OFF" indicate the period that link A-B is "up" and "down," respectively. Link A-B remains in the ON state for a certain time and then changes to OFF state for another interval. This process repeats as nodes continue their movements. We assume that T_{ON} and T_{OFF} are the average times for link A-B to stay in the ON and OFF states, respectively. Therefore,

the link life period is equal to $T_{ON} + T_{OFF}$. Thus, the link connectivity change rate is $1/(T_{ON} + T_{OFF})$.

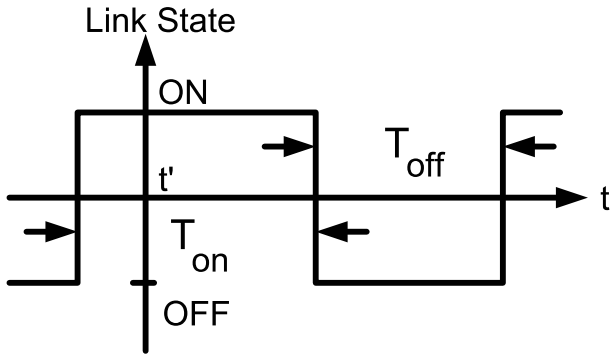


Figure 5.11: Connectivity state changes.

To simplify the following discussion, we define the probability that node B starts its epoch from B' and B" as P_{123} and P_{45}, respectively. Given that node B starts from position B', the conditional probabilities for it to follow epoch 1, 2, and 3 are P_1, P_2, and P_3, respectively. Similarly, P_4 and P_5 are the conditional probabilities for epochs 4 and 5, respectively, given that node B starts from position B". Note that $P_1 + P_2 + P_3 = P_4 + P_5 = P_{123} + P_{45} = 1$. For epoch i (i = 1, 3, 4, and 5), $T_{ON}[i]$ and $T_{OFF}[i]$ represent the length of time that link A-B is in the ON and OFF states, respectively, in the corresponding epoch. Note that $T_{ON}[1] = T_{OFF}[4] = 0$. For epoch 2, the state of link A-B changes from the OFF state to the ON state and then to the OFF state again. The time that link A-B is in the ON state in this epoch is denoted by $T_{ON}[2]$. We use $T_{OFF\_a}[2]$ and $T_{OFF\_b}[2]$ to represent the interval when node B is in the segment B'C and the segment CD (see Figure 5.10), respectively.

There are two possible cases for link A-B to be in the ON state between two adjacent OFF states. In Case I, node B starts an epoch of type 2, such that link A-B begins in the OFF state, is in the ON state when node B is between positions C and D, and then ends in the OFF state. In Case II, node B starts an epoch of type 3, which is followed by zero or more epochs of type 4, and then the sequence is terminated by an epoch of type

5. For Case I, the average time for link A-B to be in the ON state is $T_{ON}[2]$. The average time for link A-B to be in the ON state in Case II is shown in Equation 5.1 when $P_4 \neq 1$ ($P_4 = 1$ if and only if the radio range is infinite). The probability of Case I and of Case II are shown in Equation 5.2. The expected value of T_{ON}, based on the expected time in the ON state and the probability of Case I and Case II, is as shown in Equation 5.3.

$$\begin{aligned} T_{ON\_I} &= T_{ON}[2] \\ T_{ON\_II} &= T_{ON}[3] + (\Sigma_{k=0}^{\infty} k T_{ON}[4] p_4^k) + T_{ON}[5] = T_{ON}[3] + \tfrac{T_{ON}[4]P_4}{(1-P_4)} + T_{ON}[5] \end{aligned} \tag{5.1}$$

$$\begin{aligned} P_{ON\_I} &= (P_{123}P_2)/(P_{123}P_3 + P_{123}P_2) = P_2/(P_2 + P_3) \\ P_{ON\_II} &= (P_{123}P_3)/(P_{123}P_3 + P_{123}P_2) = P_3/(P_2 + P_3) \end{aligned} \tag{5.2}$$

$$T_{ON} = T_{ON}[2]\frac{P_2}{P_2 + P_3} + T_{ON}[3]\frac{P_3}{P_2 + P_3} + T_{ON}[4]\frac{P_4}{1 - P_4}\frac{P_3}{P_2 + P_3} + T_{ON}[5]\frac{P_3}{P_2 + P_3} \tag{5.3}$$

Similarly, we list all four possible cases for the occurrence of an OFF state as follows. Case I is the interval when node B leaves the coverage area using an epoch of type 5, followed by zero or more epochs of type 1, and then an epoch of type 3 that terminates the OFF state. Case II is the interval when node B moves out of the coverage area at position D using an epoch of type 2, followed by zero or more epochs of type 1, and then an epoch of type 3 where node B enters the coverage area. Case III is the interval when node B leaves the coverage at position D during an epoch of type 2, followed by zero or more epochs of type 1, and an epoch of type 2 where node B comes back into the coverage area at position C due to another epoch of type 2. Lastly, Case IV is similar to Case I except that node B uses an epoch of type 2 to terminate the corresponding OFF state. Equations 5.4 and 5.5 specify the average OFF times for each of these four cases and the corresponding probabilities of each case. This leads to the average OFF time, T_{OFF},

as shown in Equation 5.6. We assume that relative node movements are stationary (in steady state), we have $P_3P_{123} + P_4P_{45} = P_{45}$ and $P_{123}P_1 + P_{123}P_2 + P_{45}P_5 = P_{123}$. Therefore, Equation 5.7 specifies the link life period, T. Note that, in this study, we do not provide specific quantitative expressions for the probabilities and periods used in Equation 5.7. Derivation of quantitative values is complicated and is a topic for future work. However, without detailed quantitative knowledge, one can still apply Equation 5.7 to study the relationship between parameters used to define node movements and link connectivity changes. An example is provided in Sections 5.2.1.3 and 5.2.2.

$$\begin{aligned}
T_{OFF_I} &= T_{OFF}[5] + (T_{OFF}[1]P_1)/(1-P_1) + T_{OFF}[3] \\
T_{OFF_II} &= T_{OFF_b}[2] + (T_{OFF}[1]P_1)/(1-P_1) + T_{OFF}[3] \\
T_{OFF_III} &= T_{OFF_b}[2] + (T_{OFF}[1]P_1)/(1-P_1) + T_{OFF_a}[2] \\
T_{OFF_IV} &= T_{OFF}[5] + (T_{OFF}[1]P_1)/(1-P_1) + T_{OFF_a}[2]
\end{aligned} \tag{5.4}$$

$$\begin{aligned}
P_{OFF_I} &= \frac{P_{45}P_5}{P_{123}P_2+P_{45}P_5}\frac{P_3}{P_2+P_3} \\
P_{OFF_II} &= \frac{P_{123}P_2}{P_{123}P_2+P_{45}P_5}\frac{P_3}{P_2+P_3} \\
P_{OFF_III} &= \frac{P_{123}P_2}{P_{123}P_2+P_{45}P_5}\frac{P_2}{P_2+P_3} \\
P_{OFF_IV} &= \frac{P_{45}P_5}{P_{123}P_2+P_{45}P_5}\frac{P_2}{P_2+P_3}
\end{aligned} \tag{5.5}$$

$$\begin{aligned}
&T_{OFF} = T_{OFF_a}[2]P_a + T_{OFF_b}[2]P_b + T_{OFF}[1]\frac{P_1}{1-P_1}P_cP_d + T_{OFF}[3]P_d + T_{OFF}[5]P_c \\
&where \\
&P_a = p_2/(p_2+p_3) \\
&P_b = P_{123}P_2/(P_{123}P_2 + P_{45}P_5) \\
&P_c = P_{45}P_5/(P_{123}P_2 + P_{45}P_5) \\
&P_d = p_3/(p_2+p_3)
\end{aligned} \tag{5.6}$$

$$
\begin{aligned}
&T = T_1K_1 + T_2K_2 + T_3K_3 + T_4K_4 + T_5K_5 \\
&where \\
&K_1 = P_1P_3^2/(P_2 + P_3)^3 \\
&K_2 = P_2/(P_2 + P_3) \\
&K_3 = P_3/(P_2 + P_3) \\
&K_4 = P_3P_4/P_5(P_2 + P_3) \\
&K_5 = P_3/(P_2 + P_3)
\end{aligned}
\tag{5.7}
$$

5.2.1.3 Application and Discussion

In this section, we apply the analytical results of Section 5.2.1.2 to an example mobility model, specifically the random waypoint (RW) model (Refer to Section 5.1.1). In the random waypoint model, node locations are uniformly distributed within a two-dimensional space at the beginning of a simulation. A random destination, uniformly distributed within the two-dimensional space, is selected for each node. The node then moves toward the destination at a speed that is uniformly distributed from [minimum speed, maximum speed]. When a node reaches its destination, it pauses for a while. After the pause time expires, a new destination is selected and the node again moves toward the destination at a random velocity. At any given time, two nodes are connected (can communicate directly) if the distance between them is less than the radio range. Otherwise, the two nodes are disconnected (cannot communicate directly).

Generally, when the maximum speed of movements increases in the RW model, the time duration for each type of epoch (see Figure 5.10), T_i (i = 1, 2, 3, 4, and 5), decreases. This is because when nodes move at a high speed, they tend to finish an epoch in a shorter time. As a result, both the link up lifetime and the link down lifetime decrease. Therefore, the link life period, T, decreases and the link connectivity change rate increases. When the maximum pause time increases, the time in each type of epoch, T_i (i = 1, 2, 3, 4, and 5), may increase. Therefore, the link life period increases and the link

change rate decreases.

We first briefly discuss the relationships between the probability of occurrence of each type of epoch, P_i (i = 1, 2, 3, 4, and 5), and the radio range before we study the effect of the radio range on the link connectivity changes. Figure 3 illustrates trends of how the values of P_i (i =1, 2, 3, 4, and 5) change when radio range changes. (This figure shows trends, not actual values.) When the radio range is short, the area covered by node A (referring to Figure 5.10) is small. Thus, epochs of types 2, 3, and 4 are not likely to occur while epochs of types 1 and 5 are likely to occur. When the radio range increases, the possibility of epochs of types 2, 3, and 4 increases, while the possibility of epochs of types 1 an 5 decreases. Note that when the value of the radio range becomes large enough, the possibility of an epoch of type 2 is reduced since the area covered by node A becomes large enough that it is not likely for a node to enter and leave the coverage area within one epoch.

When the radio range is small, P_1 and P_5 are close to 1 while P_2, P_3, and P_4 are close to 0 as shown in Figure 5.12. Since the area covered by the center node is small, T_i (i = 1, 2, and 5) are large and T_i (i = 3 and 4) are small. According to Equation 5.7, K_1 is large and K_4 is small. Therefore, the value of the link life period, T, is large because of a large link down period and the link connectivity change rate is close to 0. This is reasonable because a small radio range leads to a high likelihood of long disconnected periods between pairs of nodes.

When the radio range is large, P_1, P_2, and P_5 are close to 0 while P_3 and P_4 are close to 1, as indicated in Figure 5.12. Since the area covered by node A is large, T_i (i = 1 or 5) is small and T_i (i = 2, 3, or 4) is large. According to Equation 5.7, K_1 is small and K_4 becomes large. Thus, the link life period, T, is large due to a long link up period and the link connectivity change rate is close to 0. Note that the period that a link is in the UP state is longer than the period that the link is in the DOWN state, which is reasonable since pairs of nodes are more likely to be connected with a longer radio range.

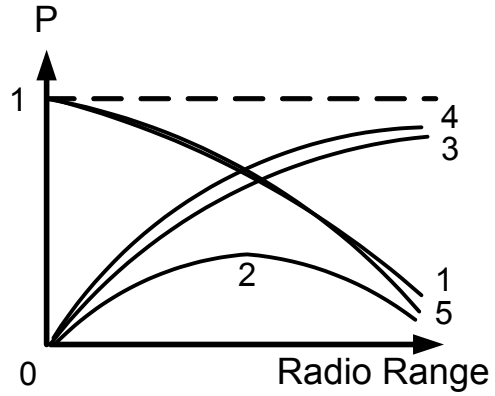


Figure 5.12: Trends of different probabilities.

When a medium value for radio range value is chosen, K_1 and K_4 also have medium values. The link life period has at least one minimum value when the radio range is in the middle since the link life period becomes long when the radio range becomes small or large. Therefore, there is at least one peak value in the value of link connectivity change rate as the radio range parameter is varied.

Minematsu, et al. describe the Characterized Environmental Indicator (CEI) in which they define the frequency of link failure (LF) as shown in Equation 5.8 [109]. The radius of transmission range refers to the radio range used in this paper.

$$LF = \frac{Speed\ of\ Nodes}{Radius\ of\ Transmission\ Range} \tag{5.8}$$

Minematsu, et al. indicate that the greater the radio range, the smaller the LF values. However, this is not always true based on the analysis discussed above. When the radio range is very small, LF should also be small because the possibility of a link being in the UP state is small. LF follows Equation 5.8 only when the radio range is larger than a certain value. In Section 5.2.2, we verify our conclusions using simulations.

5.2.2 Simulation Study

We simulated the random waypoint model to verify the analysis of Section 5.2.1. Ten independent replications are used in the simulation (ten pseudorandom seeds are generated using the algorithm presented in [110]). Two nodes are deployed in a 200×200 m^2 map. A long warm-up period of 360,000 seconds and a simulation time of 3,000 seconds are chosen according to [81]. The Bonn motion mobility scenario generator [95] is used to create trace files. In the first set of simulations, we fix the radio range to 40 m and observe the link connectivity change with respect to the maximum movement speed and pause time. The maximum movement speed ranges from [3, 30] m/s with a step of 3 m/s. The maximum pause time ranges from [10, 100] s with a step of 10 s.

Figures 5.13 and 5.14 show the link up lifetime and link down lifetime. Figures 5.15 and 5.16 show the link life period and link change rate. When the maximum speed increases, we see decreases of link up lifetime and link down lifetime in Figures 5.13 and 5.14 and link life period in Figure 5.15. We also observe increases in the link change rate in Figure 5.16, which agrees with the discussion in Section 5.2.1.3. When the maximum pause time increases, the link up lifetime and the link down lifetime also increase. Therefore, the link life period increases and, thus, the link change rate decreases.

In the second set of simulations, we fix the maximum pause time at 40 seconds and study the link connectivity changes with respect to varying radio range and maximum node speed. The radio range is varied from 10 m to 160 m with a step of 10 m. The link up lifetime and link down lifetime are shown in Figures 5.17 and 5.18. The link life period and link change rate are shown in Figures 5.19 and 5.20. When the radio range is small, we can observe small link up lifetime and large link down lifetime in Figures 5.17 and 5.18. When the radio range increases, the link up lifetime increases while the link down lifetime decreases. Therefore, the overall link life period is large when the radio range is small or large. The link life period reaches its smallest value when the radio range is at a medium value. Therefore, the link change rate reaches its peak value when the radio

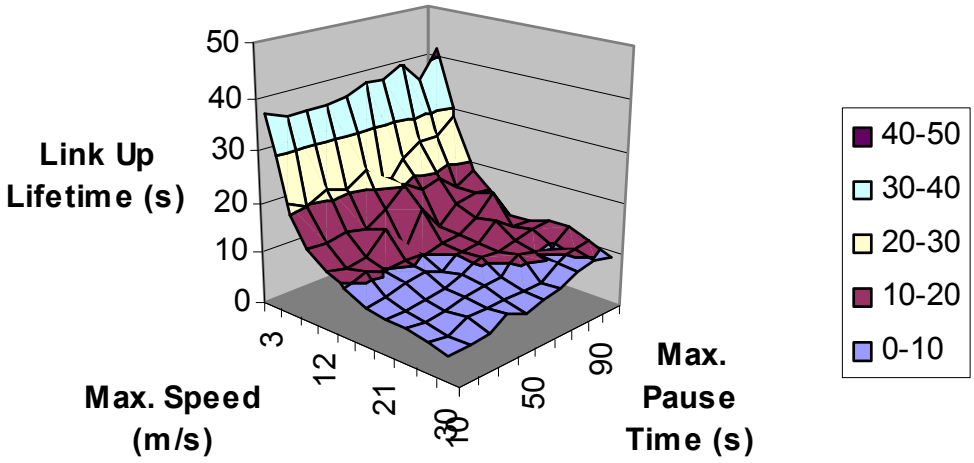


Figure 5.13: Link up lifetime with fixed radio range of 40 m.

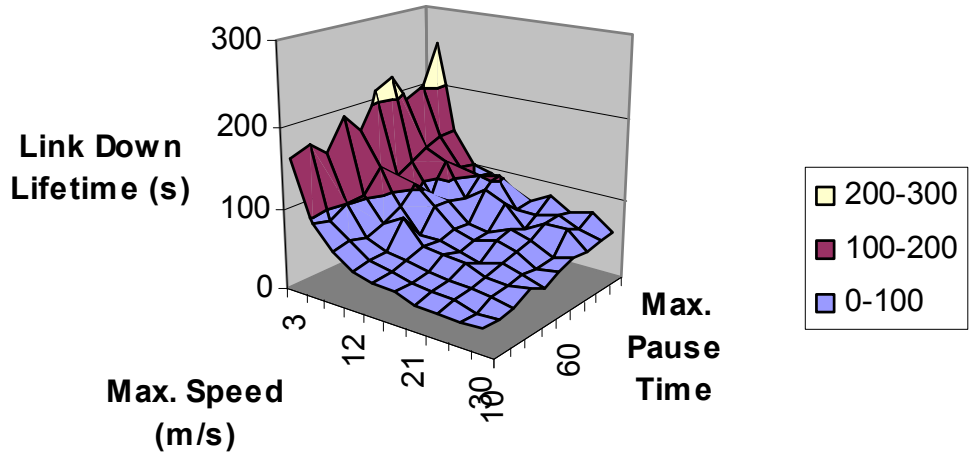


Figure 5.14: Link down lifetime with fixed radio range of 40 m.

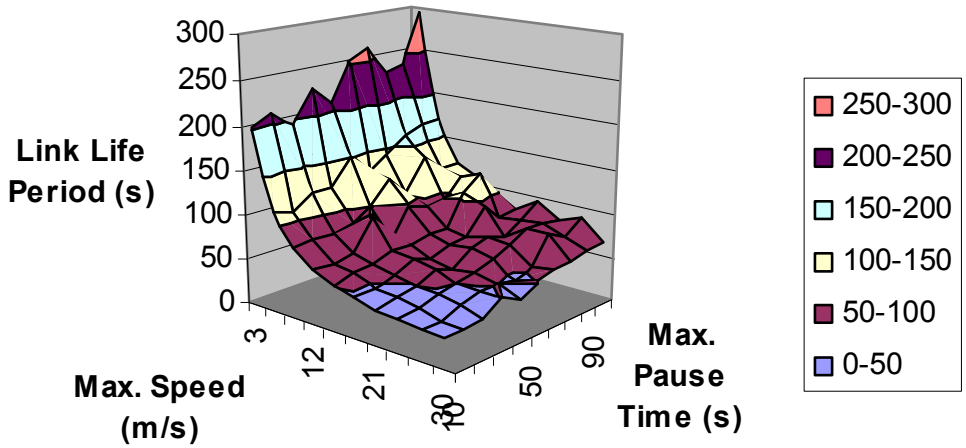


Figure 5.15: Link life period with fixed radio range of 40 m.

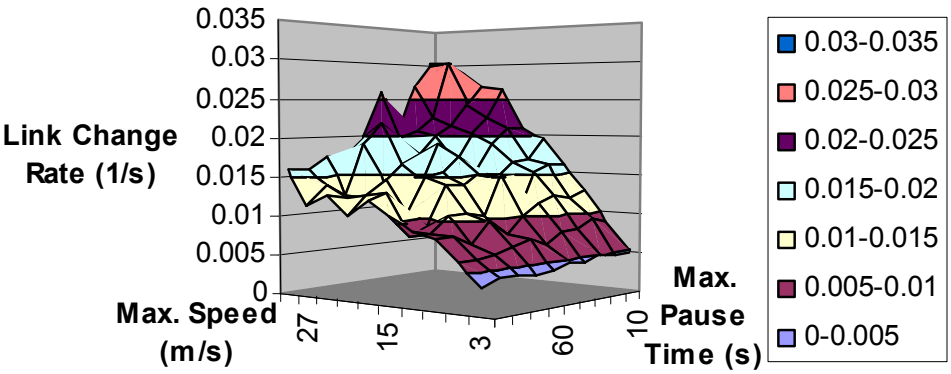


Figure 5.16: Link change rate with fixed radio range of 40 m.

range is at a medium value. These results agree with the analysis of Section 5.2.1.

As the radio range increases, we obtain scenarios that have long link life periods with low connectivity (due to long link down lifetime), medium link life periods (due to low link up lifetime and link down lifetime), and then long link life periods with high connectivity (due to long link up lifetime). In other words, various radio ranges can produce various combinations of the average node degree, the link connectivity rate change, and the average length of paths. These factors all affect control overhead in MANET routing protocols [1]. This implies that using various radio ranges for the simulation experiments is a good choice for MANET evaluations and is better than varying the parameters of the random way point model, specifically the maximum node movement speed and the maximum pause time.

5.2.3 Conclusion of Study of Link Lifetime

Link connectivity changes are among the most important factors that affect the operation and performance of MANET protocols. A better understanding of link connectivity in a MANET can be helpful for evaluation of MANET routing protocols and, potentially, be applied to improve MANET routing protocols. However, little research has been done on the topic. We provided an analytical study of link connectivity in MANETs. We verified our results using simulation experiments with the random waypoint mobility model. Results of the quantitative study indicate that varying radio range is a good approach to evaluation of MANETs. This is used to guide our simulation study in Chapter 7.

Currently, quantitative statistical values for the parameters of the relative movement model are not available. A future step is to study those parameters and the relationships between them and mobility models. Future work also includes application of the analysis, including selection of simulation parameters and evaluation and improvement of MANET routing protocols. Another application of this work is to apply the study of

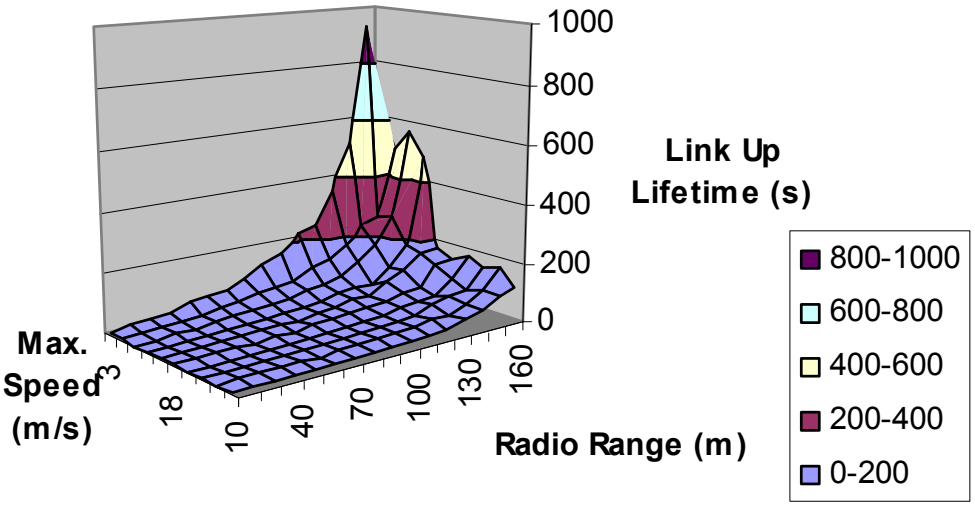


Figure 5.17: Link up lifetime with fixed maximum pause time of 40 s.

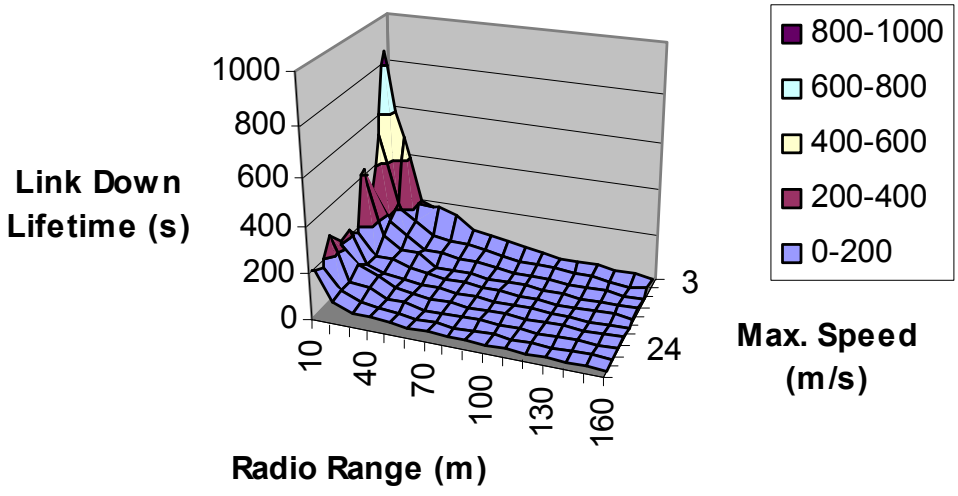


Figure 5.18: Link down lifetime with fixed maximum pause time of 40 s.

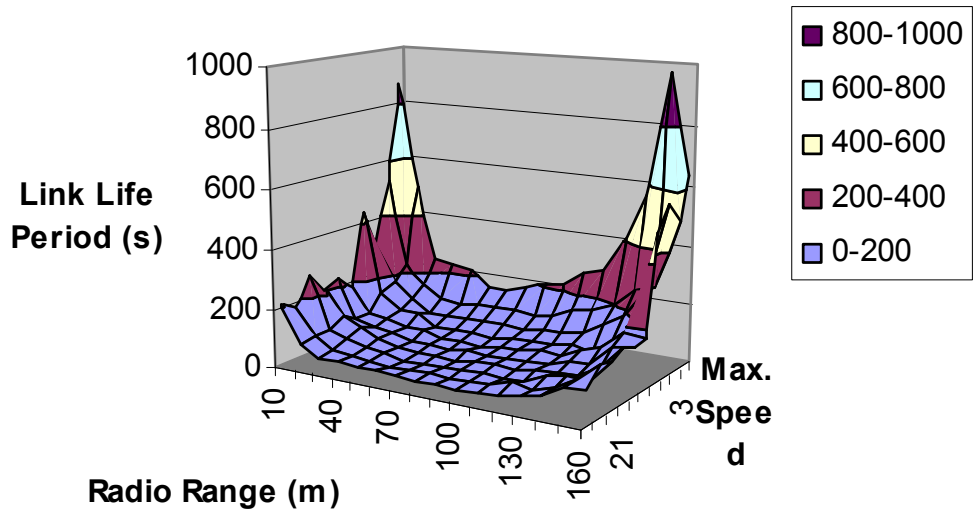


Figure 5.19: Link life period with fixed maximum pause time of 40 s.

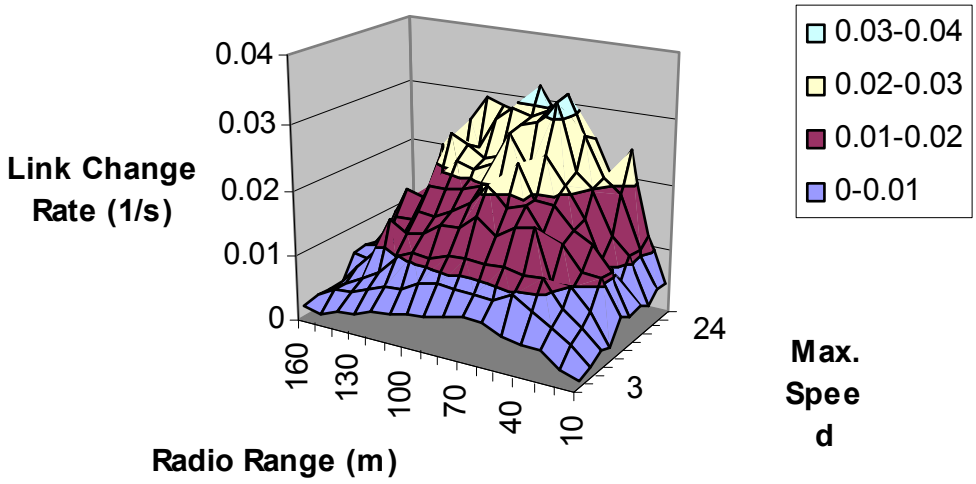


Figure 5.20: Link change rate with fixed maximum pause time of 40 s.

mobility to the control of MANET protocols, for example, as in the application discussed in the following Section.

5.3 Neighbor Stability and Adaptive Control of Periodic HELLO Intervals

According to the RNS framework proposed in Chapter 3, an N-node MANET can be considered as a system with input parameters and output metrics as shown in Figure 5.21. The MANET itself contains N mobile nodes and is supported by a certain set of protocols, including routing protocols. There are three types of input parameters for such a MANET, which include the network, user traffic, and protocol profiles. The network profile defines the physical environment such as the radio range, how nodes move, and, thus, how the network topology changes. The user traffic profile describes different types of user traffic flows in the network, for example, the number of user data flows and the parameters of UDP or TCP traffic flows. The protocol profile defines the parameters for the protocols running in this system. Examples include the constant interval between HELLO messages defined in a hello protocol that supports MANET routing. There are many possible outputs for a MANET, depending on users' interest. Examples include the delivery ratio for user data, bandwidth consumed by control messages, throughput of user data, end-to-end delay, or power consumption.

Based on this model, it is possible to measure some outputs and/or some internal variables in a MANET and use them as feedback to tune the inputs of the system so that the entire system can have better performance. To our knowledge, there is no prior study of this kind in the literature.

In this chapter, we present a study of the use of feedback control in a MANET. Specifically, we control the interval time between HELLO messages adaptively for a MANET

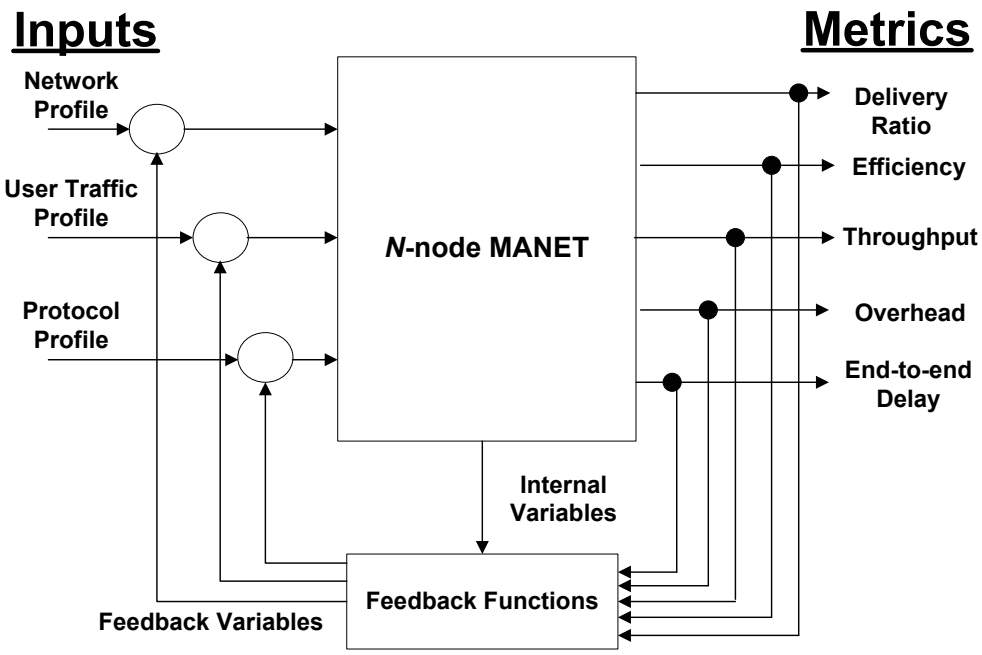


Figure 5.21: A model of a MANET system.

using the feedback control idea and a novel concept of neighbor stability. Section 5.3.1 introduces the novel neighbor stability concept and Section 5.3.2 presents an adaptive hello scheme using neighbor stability. A simulation study is summarized in Section 5.3.3. This study shows that the adaptive scheme we used can improve some, but not all, performance metrics. This study can potentially lead to future research topic that applies control theory to MANET protocols.

5.3.1 Adaptive Hello Using Neighbor Stability

Traditionally, a hello protocol sends HELLO messages periodically with a constant time interval. The time interval is defined by users. For example, the OSPF standard recommends a ten-second interval and the TBRPF draft suggests a one-second interval. However, to our knowledge, there is no prior study on the proper interval for a MANET. In this section, we present a novel model to adaptively adjust the time interval according to network parameters measured at run time.

Before we describe the model, some terms are defined as follows. When a node wants to send out a HELLO message, it takes a snapshot of its neighbor list at this very moment. The neighbor stability (NS) for a node is defined as the number of neighbor changes, including new neighbors found and old neighbors lost, divided by the larger node degree in the two adjacent snapshots. In case a node does not have any neighbors in the two adjacent snapshots, the NS is defined as 2. Therefore, a larger NS implies more rapid neighbor and link connectivity changes. Note that the value of NS is always within the range of [0, 2] according to the definition. A NS value that is close to zero implies that a stable neighbor node set while a NS value that is close to two indicates that the neighbor nodes change rapidly. Nodes will share node stability information in HELLO messages. Quantized NS (QNS) is defined as the smallest integer greater than or equal to $(NS/2) * (2^n - 1)$, where n is the number of bits that are used to store the QNS value in HELLO messages. Rule 5.3.1.1 defines how the QNS value is generated in a node that runs the adaptive HELLO protocol investigated in this section.

Rule 5.3.1.1 *Nodes calculate and broadcast their own QNS values according to the neighbor lists in two adjacent snapshots of neighbor lists. The interval between adjacent HELLO messages for a node depends on the selected QNS value, which is the maximum value in the set {QNS values from new neighbors $\bigcup$ the node's own QNS}. The selected value is used as the index to select the interval between adjacent HELLO messages. Larger QNS values indicate the use of smaller intervals. This maximum QNS value may vary from time to time. Therefore, the chosen interval is adaptive based on the real network environment.*

If a group of nodes move in a similar pattern, the link connectivity change rate in this group is relatively low and the QNS values at these nodes are small. The intervals between adjacent HELLO messages chosen by these nodes are going to increase and eventually reach the maximum allowable value. If a node moves into an area in which nodes are relatively static with respect to each other, the QNS value of this node is going to affect all the original nodes in this area by reducing the intervals used by those nodes.

This is reasonable since all nodes need to detect the connectivity changes caused by the new node as soon as possible.

Generally speaking, in the traditional hello protocol with constant intervals, if a neighbor's HELLO message is missing for a certain number of successive intervals, this neighbor is considered to be lost. We have the following rule to guarantee that the same statement is valid with our new adaptive hello protocol.

Rule 5.3.1.2 *A node finds a new neighbor when it receives the first hello message from this neighbor. The estimated QNS assigned for that new neighbor equals to the QNS value of the node or the QNS broadcast by that neighbor node, whichever is larger. This estimated value is usually less than or equal to the maximum QNS chosen by this neighboring node according to Rule 5.3.1.1. Therefore, the estimated interval for this neighbor is guaranteed to be greater than or equal to the real interval used by this neighbor. If the neighbor is not a new neighbor, the estimated QNS value of this neighbor is the value broadcast by this neighbor. We can prove that the actual QNS value chosen by this neighbor is always greater than or equal to the estimated value.*

A node assumes that a neighbor is lost if the HELLO message from this neighbor is missing within m successive intervals. Here the interval is associated with the corresponding estimated QNS.

According to the rules we defined above, we have two theorems and corresponding proofs as follows.

Theorem 5.1 *The estimated QNS value for a new neighbor, denoted EQNS, is always less than or equal to the selected QNS (SQNS) value of that neighbor.*

Proof: Assume that a center node's QNS is Q_{center} and the new neighbor's QNS value is Q_{nbr}. Therefore, $EQNS = \max\{Q_{center}, Q_{nbr}\}$. According to Rule 5.3.1.1, $SQNS$ for the new neighbor is the maximum one in {neighboring QNS values recorded by this

neighbor, Q_{nbr}}. Beause $Q_{center} \in$ {Neighboring QNS values recorded by this neighbor}, $EQNS \in$ {Neighboring QNS values recorded by this neighbor, Q_{nbr}}. Therefore, $EQNS \leq SQNS$. □

Theorem 5.2 *The EQNS value for an old neighbor is always less than or equal to the SQNS of that neighbor.*

Proof: Similar to the proof of Theorem 5.1, we assume that the QNS value broadcast by the old neighbor is Q_{nbr}. So $EQNS = Q_{nbr}$. The $SQNS$ of this neighbor is the maximum value in {Neighboring QNS values recorded by this neighbor, Q_{nbr}}. Therefore, $EQNS \in$ {Neighboring QNS values recorded by this neighbor, Q_{nbr}}, in which the maximum value is $SQNS$. Thus, we prove that $EQNS \leq SQNS$. □

There is a common problem with maintaining neighbor lists in hello protocols. Assume that a node moves out of its neighbor's range after the link between them is considered to be up. This node stays out of the radio range of its neighbor and comes back after a certain amount of time. It is possible that one node is considered to be lost at the other node while the other node is still in this node's neighbor list. This problem is likely to occur in the hello protocol with adaptive intervals. This can be solved in the same way used by traditional protocols with constant intervals, for example, to force the neighboring link information to expire at both nodes if this case occurs.

When a node receives a HELLO message from a new neighbor, the node sends a reply message immediately if the interval it chose is too long, as described in the following rule. (The definition of the term "too long" depends on user's preference. For example, if a value that is larger than the default hello interval is considered as "too long".)

Rule 5.3.1.3 *A node replies to a HELLO message immediately if a new neighbor's HELLO message is received and the interval chosen is longer than a user defined length.*

This rule is used to avoid a possible reaction to a HELLO message from a new

neighbor with a long delay. The next section present a study of applying this adaptive scheme to OLSR, a proactive routing protocol.

5.3.2 Adaptive HELLO in OLSR

A MANET with our adaptive scheme is shown in Figure 5.22. Note that the function $nbr(i, j)$ returns 1 if node i is node j's neighbor or $i = j$. Otherwise, the function returns 0. Therefore, a node only collects QNS values from its neighbors and its own QNS. The feedback function for a node, say k, selects the maximum known QNS and calculates the change in interval between adjacent HELLO messages using Equation 5.9. The selected interval is equal to the default interval minus Δ.

$$\Delta = [\max_{i \in \{new\ neighbors\ of\ k\} \bigcup \{k\}} \{QNS_i - Normal\_QNS\}] * Interval\_Step \tag{5.9}$$

The adaptive scheme intends to use long intervals for networks with low mobility hosts. Therefore, it could have a potential late-detect problem in that nodes will not be able to detect a lost neighbor early enough due to a long interval. Thus, the throughput may be reduced. Our simulation results in the following sections also show that the delivery ratio, which is proportional to the throughput, is slightly reduced with the adaptive hello scheme. To solve this problem, we propose the following rule.

Rule 5.3.2.1 *When a node receives a HELLO message from a neighbor, if the signal power of this hello message is reduced, compared with the previous signal power from the same neighbor, this neighbor is considered to be leaving relative to the receiving node. If the power is lower than a predefined value, the neighbor is counted as lost when the QNS is calculated. We can also quantize the power to get QP, and use QP^{-1} as the weight to calculate a weighted QNS.*

This rule requires cross-layer design between the network layer and lower layer

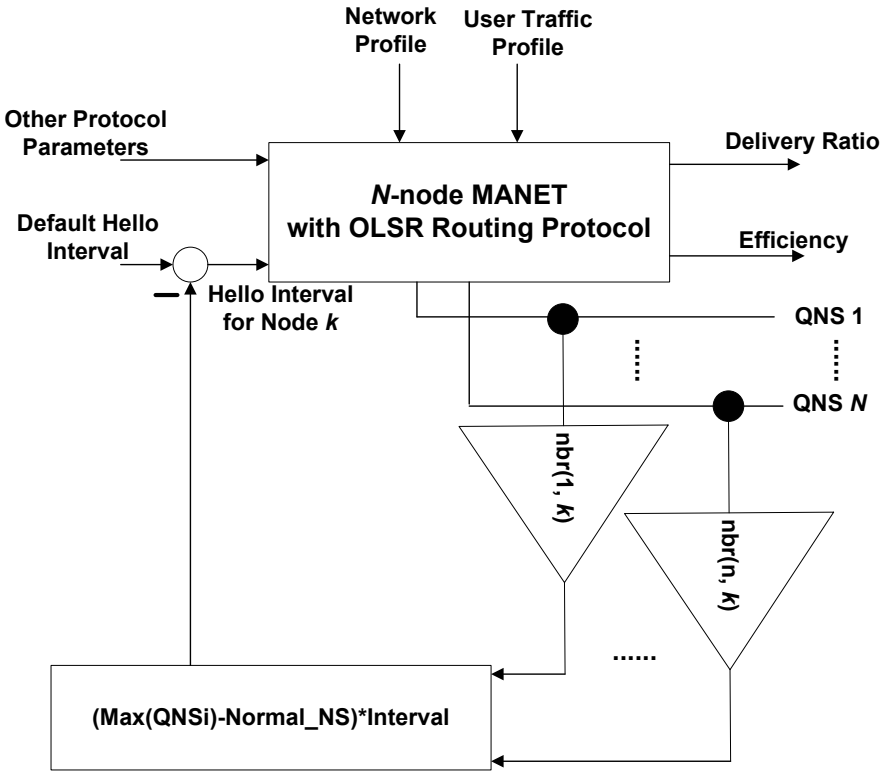


Figure 5.22: Feedback control model for OLSR.

protocols. To our knowledge, there is no product on the market can achieve this. However, this approach should be feasible. Using this rule, when a neighbor starts to depart, the center node reduces the hello interval to more quickly detect the potential loss of the neighbor. Thus, it can avoid the late-detect problem and improve the throughput of the network.

5.3.3 Simulation Study

We use the OLSR protocol to study the adaptive scheme. The HELLO messages used in OLSR have reserved fields (OLSR has more than 1 byte of reserved fields). Therefore, we can exchange node mobility values with $2^8 = 256$ possible values using the default packet format without extra overhead. In this study, we set $n = 8$ (Recall that n is the number of bits that are used to store the QNS value in the hello messages). A

$Normal\_QNS$ is defined so that if the selected QNS equals the $Normal\_QNS$, the hello interval is the default interval, say 2 seconds. The QNS is set to the minimum integer that is larger than $(NS/2.0) \times 255$ and is also within the range [0, 255]. The hello interval is defined by $the\ default\ interval - step\_interval \times (selected\ QNS - Normal\_QNS)$, where $step\_interval$ is a user defined step interval. Simulation experiments are used to compare the performance of OLSR with the adaptive hello scheme with the performance of the original OLSR. The traffic load is a UDP CBR flow that is generated at node 1 with destination 0. The data packet size is 1400 bytes and the interval between user data packets is 0.1 second. Random Waypoint and Reference Point Group Mobility (RPGM) models are used. (Simulation results using RPGM are similar to what we observed using the random waypoint model. Therefore, we do not include results using RPGM in this dissertation.) The warm up time is 1,000 seconds and five independent replications are used.

The metrics we observed are overall control overhead and user data delivered in terms of megabytes. Figure 5.23 compares control overhead for the original OLSR and the OLSR that uses the adaptive hello scheme. In some scenarios, the modified OLSR has higher overhead than the original OLSR, while for some other scenarios the original OLSR has higher overhead. The reason is the overhead of OLSR with the adaptive hello scheme depends on the relative stability among nodes. The throughput using the adaptive hello scheme is better than the original OLSR in most cases except for the smallest maximum possible speed (refer to Figure 5.24). This is because, in most cases, the adaptive scheme can detect the neighbor state changes more efficiently than the original OLSR. Therefore, more user data packets can be delivered. We noted that the adaptive scheme has lower throughput with the smallest maximum speed. To further investigate the reason, we simulated another scenario with low node mobility.

The results of the low mobility scenario are summarized in Figures 5.25 and 5.26. The overhead is reduced using the adaptive scheme, but the percentage of delivery of data packet is also reduced. This is because when the relative mobility among nodes is low, the adaptive scheme tends to increase the hello interval and thus, is less sensitive

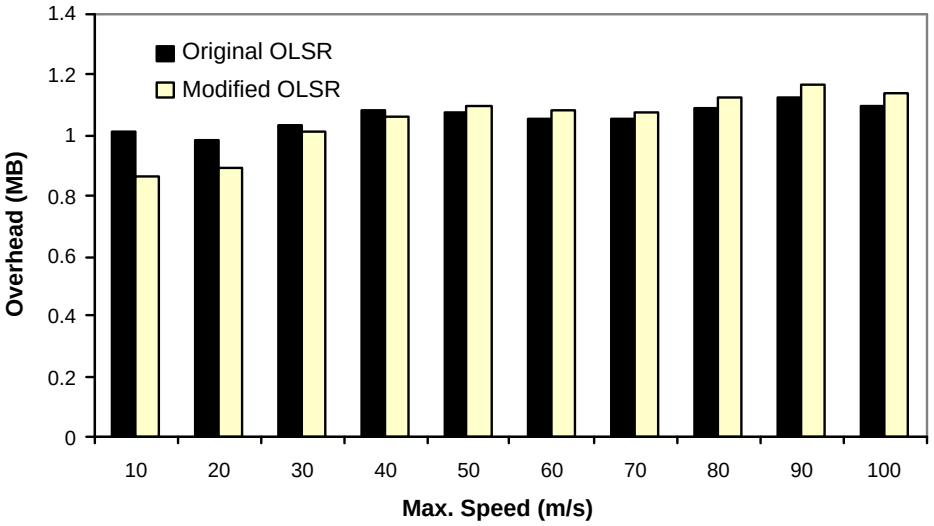


Figure 5.23: Overhead comparisons using the random waypoint model.

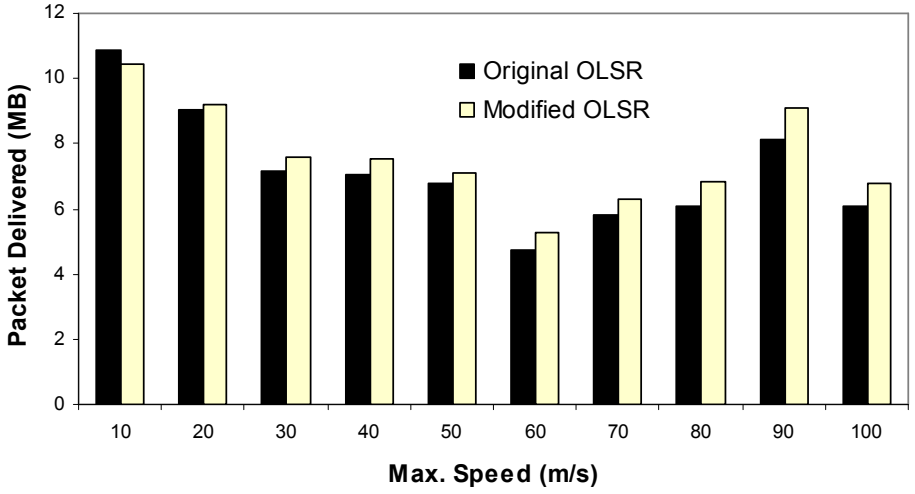


Figure 5.24: Throughput comparisons using the random waypoint model.

to link failures. Therefore, the overhead and throughput are both reduced. This suggests that the adaptive scheme we proposed should work well in higher mobility scenarios, but it needs improvement for lower mobility scenarios.

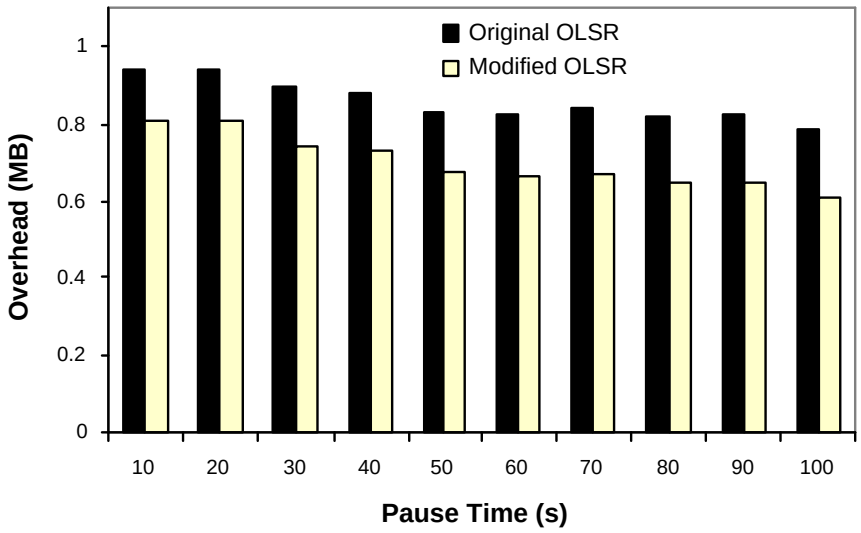


Figure 5.25: Overhead comparisons with lower mobility using the random waypoint model.

5.3.4 Conclusion of Study of Neighbor Stability

We conducted a study of an adaptive hello scheme based on neighbor stability. This adaptive scheme estimates neighbor movements based on neighbor change rates and adaptively adjusts the interval between adjacent HELLO messages. Simulation results show that this adaptive scheme can improve throughput, at the cost of extra overhead, for high mobility scenarios. In other words, the scheme can be used in high mobility scenarios in which power and bandwidth are not limited. The scheme does not improve the throughput in low mobility scenarios since the hello protocol becomes less sensitive to link failures due to very long hello intervals set by the adaptive hello scheme. Future work includes improvement of the proposed adaptive scheme, such as using more

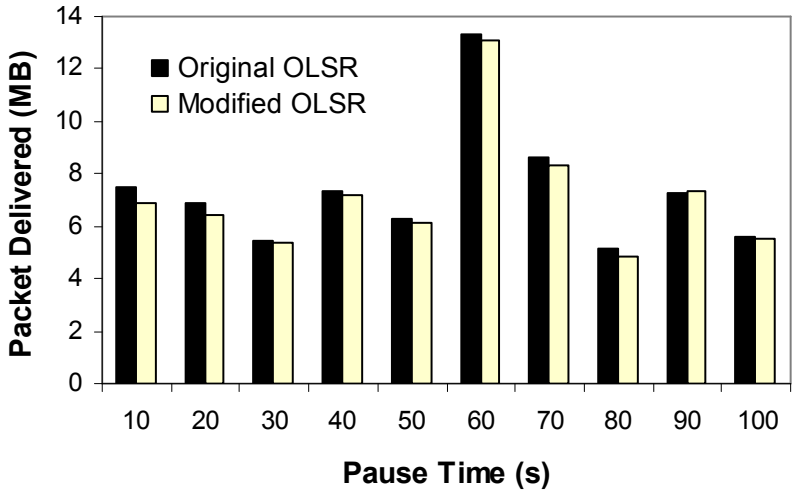


Figure 5.26: Throughput comparisons with lower mobility using the random waypoint model.

variables to control the interval instead of just the neighbor stability, and more extensive comparison study using different mobility models.

5.4 Summary of Study of Node Mobility

We presented a study of a widely used mobility model, the random waypoint model. We discovered some disadvantages of this mobility model and proposed a new link connectivity model for simulations of MANETs to provide a more easily controlled mobility model. We also conducted a quantitative study of link lifetimes in MANETs using the concept of relative movement and studied an adaptive hello scheme for MANET routing using the concept of neighbor stability in this chapter. There is increasing interest in research on link lifetime prediction and its application to MANET routing. Link lifetime information should be helpful for protocols and applications at different layers, including the network layer, because node mobility is one of the fundamental characteristics of a MANET and knowledge of node mobility may be used to adaptively adjust control protocols and improve the performance of MANETs.

Chapter 6

Simulation and Emulation Tools

We used simulation and emulation tools to verify the new protocol and to compare it with other routing protocols. This section introduces network simulator (ns2) models and a Linux test bed we developed for emulation of MANET topologies. Section 6.1 presents the network simulator ns2. Section 6.2 discusses problems with protocol implementations and fair comparison assumptions we used in our simulation experiments summarized in Chapter 7.1. An emulation tool we developed for emulation of MANET is proposed in Section 6.3, which is based on [111].

6.1 Network Simulator and Network Animator

Ns2 and network animator (nam) are popular simulation tools used by many researchers for network simulation [98, 112]. They are "freeware", coming with support for different routing protocols. They are developed in C++ and TCL script language for Microsoft Windows, Linux, and FreeBSD operating systems. Users can develop new protocols or revise existing protocols easily using the concept of class in C++. TCL can allow users to configure simulations, including parameters and protocols, so that re-compiling is not

necessary.

When simulating with ns2, one can choose different combinations of a MAC layer protocol, a network layer protocol, and a transport layer protocol. Nam can animate the simulation scenarios using output trace files from ns2. There are many packages available for various wireless ad hoc routing protocols. Therefore, we can develop ns2 models for our routing protocol and compare its performance with that of other protocols.

6.2 "Bugs" and Fair Comparison Issues in ns2

Simulation of MANETs is widely used in the literature [86, 90, 101, 102, 103, 104, 105, 106]. Conclusions are drawn based on simulation results. Recently, MANET simulation studies receive increasing attention since the IETF standardization procedure requires more experimental results on MANETs, using different routing protocols [2]. However, there is lack of study, verification or validation, of network simulators for MANETs. This section presents our study of the simulator, ns2, itself.

6.2.1 Introduction

Simulations and corresponding conclusions are accepted because of the assumption that the simulation model and protocol implementations used can actually characterize and represent what happens in the real world in a random manner. However, there are only few papers discuss this assumption for MANET simulations. We discussed the long convergence time issue for link durations in Section 5.1 and [81]. Liu, et al. address the problem in the RW model that the steady state of node speed may converge to zero [96]. Besides these known issues, we also discovered some other problems and issues in MANET simulations that should receive attention. For example, the group mobility models used in the literature may generate improper node speeds. Other issues include

the fact that some protocols are not modeled properly in the ns2 simulator. These issues may invalidate simulation results, especially for the performance evaluation of performance of MANET routing protocols or higher layer protocols. Moreover, there are some different assumptions used in the models of different MANET routing protocols in simulators, which may lead to unfair comparisons of different MANET routing protocols. This is also an important issue in simulation of MANETs. Thus, we believe that it is necessary to share our experience in MANET simulations with other researchers.

In this section, the problems and bugs we discovered are addressed and possible solutions are presented. Although some issues presented relate specifically to ns2, other simulators for MANET should also receive scrutiny since the same issues may be present when they are chosen to do MANET simulations. The problems and issues discussed in this section are mainly for simulation of MANET routing protocols. They are also important for the evaluation of other MANET protocols that rely on a MANET routing protocol, such as transport layer protocols or application layer protocols.

6.2.2 Unexpected Speed Problem in Group Mobility Models

Mobility models are used in MANET simulations to model physical node movements. These models can be categorized into individual mobility models and group mobility models, which define node movements in different ways [80]. Group mobility models usually assign a group leader for each group. Group leaders follow certain moving patterns and the movement of a group member is the combination of the movement of its group leader and its own relative movement with respect to its group leader [80, 101, 102, 105, 106]. A typical group mobility model is the reference point group mobility model (RPGM), which was first introduced by Hong, et al. [101]. This type of mobility models tends to be used to simulate rescue teams or squads of soldiers that move in groups. It is widely used in many MANET simulation studies [80, 101, 102, 105, 106]. The links between group members change less frequently than the links that connect two

nodes not in the same group. Figure 6.1 illustrates how group mobility models define node movements. O_1 and O_2 are the locations of a group leader O before and after moving. The maximum group distance r is defined as the maximum distance between the group leader and the individual nodes in the group, as shown in Figure 6.1. Locations S and S' are the starting and ending positions for an individual node. For individual nodes, N_s and N_d are the the furthest possible positions to the ending and starting positions with respect to the group leader O. This node's movement with respect to its group leader is defined by the arrow $S'D$. Therefore, the movement for this individual node is the combination of movements O_1O_2 and $S'D$, which is SD, as shown in Figure 6.1.

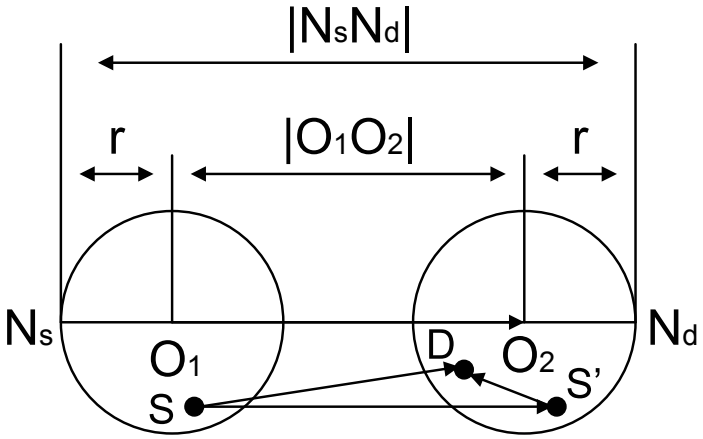


Figure 6.1: Group mobility model notation.

We use RPGM to illustrate the improper speed problem in group mobility models. Assume that the time interval for the group leader to move from O_1 to O_2 is t. According to the definition of RPGM [101], the speeds of the group leader O and the individual node S, denoted as S_{leader} and S_{node}, respectively, can be determined as follows.

$$\begin{aligned} S_{leader} &= |O_1O_2|/t \\ S_{node} &= |SD|/t \leq |N_sN_d|/t = (2r + |O_1O_2|)/t = \tfrac{2r}{t} + S_{leader} \end{aligned} \tag{6.1}$$

Since the interval *t* ranges from 0 to a certain maximum time, it is possible that S_{node}

becomes very large when the value of t is small enough with respect to the value of r. For example, S_{node} can be larger than 100 m/s if $t = 0.01 \times r$. In fact, we observed that some individual nodes may move at 500 m/s or more for a short period of time in simulations of movements of squads of soldiers defined by RPGM. Besides RPGM, other group mobility models summarized in [80], such as the column mobility model, nomadic community mobility model, and pursue mobility model, may also generate this kind of improper node speed because these models define individual node movements in a similar manner. Therefore, results of MANET simulations using these mobility models might not be valid, especially for simulations of MANET protocols that study link stability or path stability based on speed estimations.

This problem can be solved with a slightly revised definition with a maximum individual node speed. For example, the potential destination set for an individual node can be defined as the intersection area of two circles, one with O_2 as the center and r as the radius and the other one with S as the center and the maximum node speed times the time interval t as the radius. The destination location can be uniformly distributed in this intersection area. We discussed this issue with C. de Waal [97], the author of one mobility trace file generator named BonnMotion [95]. The revised BonnMotion does use this new definition to fix the improper speed bug in the RPGM model.

The improper speed problem affects simulations that use the original group mobility model in spite of the simulators that are used. In the next section, we address two "bugs" that reside in a widely-used simulator, ns2. If other simulators use code ported from ns2, these bugs should be checked as well.

6.2.3 ARP Protocol and IP Layer Bugs in ns2

This section discusses two bugs in ns2, which we call the "ARP bug" and the "IP header bug". To clearly state the problems, we first describe the processing procedures for a

packet in ns2. The internal structure of ns2 is shown in Figure 6.2. The routines to process user data packets and routing control packets are slightly different.

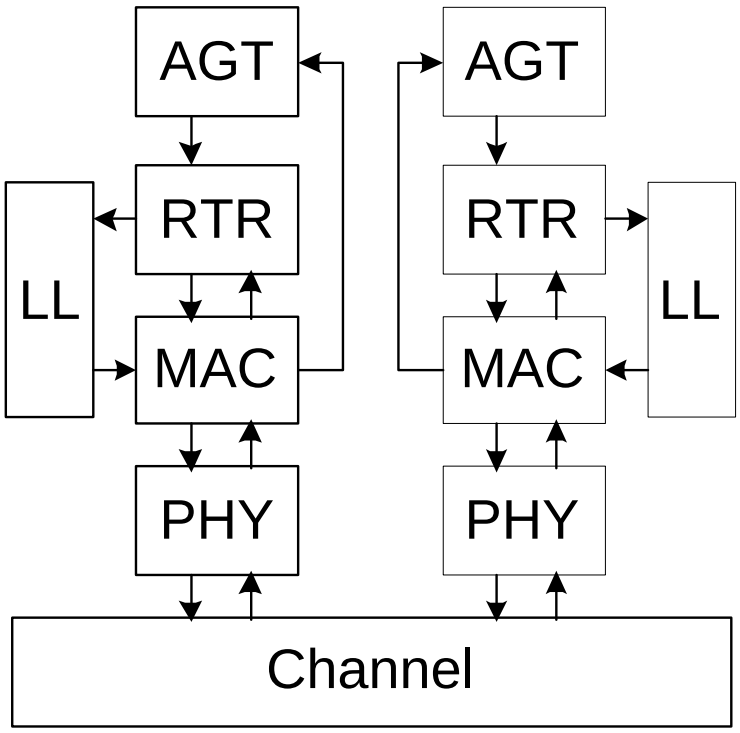


Figure 6.2: The internal organization of ns2.

In ns2, a user layer packet, such as a TCP or UDP packet, is first generated at the user agent (AGT) layer, which encapsulates the transport and higher layers. This packet is then passed to the routing agent (RTR) layer, which simulates the network layer, searching for the corresponding next hop. If there is no route for this packet, it is either dropped or queued for a short period of time allowing the RTR protocol to discover a proper route. If the route is available, the user packet is forwarded to the link layer (LL). The LL runs the ARP protocol for all unicast packets that do not have their next hops' MAC address. If the MAC address of the next hop node is known, the packet is forwarded to the physical layer such as wired or wireless physical layer depending on the configuration. The channel model is used to connect the physical layer of neighboring nodes. A user packet traverses the channel and is captured by the neighbor's physical layer. It is then passed to

the neighbor's MAC layer. If the destination MAC address in this packet is not a broadcast address or does not match the receiver's MAC address, the user packet is dropped by this neighbor. Note that if the destination IP address matches the receiver's IP address, this user packet is passed to AGT directly. Otherwise, the packet is passed to the RTR protocol to use the IP forwarding service.

The processing of routing control messages is slightly different. A routing control message can only be generated by the RTR protocol. Broadcast control messages can skip the LL layer since they do not need the MAC address of next hops. Unicast control messages follow a procedure similar to that for user packets. All routing control messages are passed to the RTR directly.

The ARP and IP header bugs are described in the following sections.

6.2.3.1 ARP Bug

The standard ARP protocol has a time-out scheme for any ARP REQUEST messages it sends. ARP resends a REQUEST message if no reply comes back before the timeout and gives up an ARP request operation after a certain number of retries. However, the ARP protocol implemented in ns2 does not implement a time-out and retry scheme. In the simulation of a wired network, the ns2 ARP protocol may not cause significant problems since destination neighbor nodes usually can receive any ARP REQUEST and send REPLY messages back in time without difficulties. In other words, a single ARP REQUEST may be enough in wired network simulations. However, in a MANET, it is certainly possible that ARP REQUEST and REPLY messages may be lost due to collisions or node movements. In the ARP model in ns2, a packet associated with an ARP REQUEST is dropped if and only if a second packet to be sent to the same next hop arrives. This packet can be held by the ARP protocol forever unless an ARP REPLY from the destination node is heard. This leads to potential extremely long delays. We use an example MANET using the OLSR routing protocol [53] with nodes A, B, C, and D, to illustrate this problem.

We assume that two nodes are connected if and only if the distance between them is less than or equal to the defined radio range. In other words, the free space radio propagation model is used. The example is shown in Figure 6.3. The locations in this figure are defined by (x, y) axes and the radio range is 180 units. We assume that at the beginning, the network is static for a period of time until all nodes set up proper routing tables and then node C starts to move to location C_1. When the link between AC is down due to C's movement at time t_1, node A sends a UDP packet to node D. Since there is a delay for OLSR to detect neighbor loss, node C is still in A's neighbor list and links AC and CD are still in node A's topology control database. Node A uses C as the next hop to destination node D. But, there is no ARP REPLY from node C and the outgoing UDP packet is held at A's ARP function. Assume node C moves back to its original location from location C_1 after a long period of time t_2, say $5,000$ seconds. After link AC and BC are up again, if node B sends a UDP packet to E, node B generates an ARP REQUEST asking for C's MAC address and then C sends an ARP REPLY as the reply to this ARP REQUEST. Node A also receives this ARP REPLY message. The ARP protocol at node A then sends the UDP packet that is generated $5,000$ seconds before to node D. We observed this behaviour for UDP packets. Such packets experience very long delay in ns2 simulations. Note that the values of t_1 and t_2 may be random. Therefore, simulation results, including end-to-end delays and throughput, may not be valid if the ARP bug is not fixed.

We propose two solutions to fix this bug. One is to implement a time-out and retry scheme in the ns2 ARP model. The other solution is to use a priority buffer queue that holds the outgoing unicast packets and bypasses the LL layer in ns2. Since MANET nodes can learn the MAC addresses of their neighbors via routing control messages, the RTR protocol in ns2 can provide the MAC addresses of the next hops instead of using ARP. The priority queue approach can also support some advanced features, for example, assigning different priorities for different types of packets or rescheduling the next hops for buffered unicast packets when the local routing table is updated. Therefore, the priority queue solution is the one we recommended and used in our simulation experiments discussed

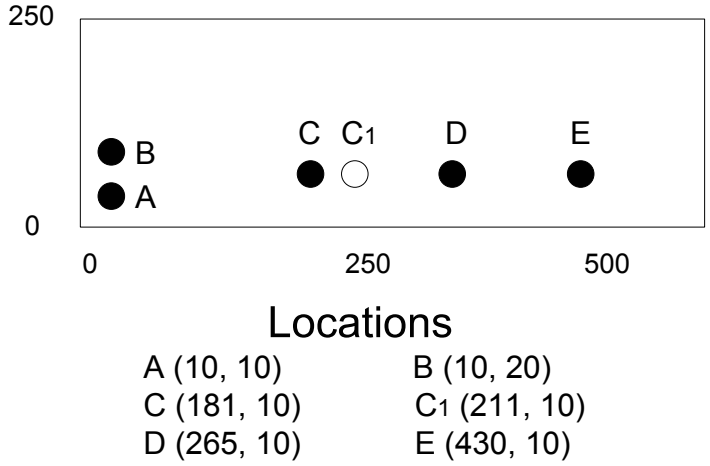


Figure 6.3: An example showing the ARP bug.

in Chapter 7.

6.2.3.2 IP Header Bug

The "IP header bug" presented in this section is due to the mis-calculation of the size of user packets. This is because the IP layer model implemented in ns2 is not a duplex model. In other words, the send and receive procedures of a user packet at the networking layer are different. Only the RTR routing agent adds an IP header when a user packet is sent while no entity removes the IP header when the packet is received at a node. Therefore, the size of user packets sent by the senders is different from the size of user packets received at destinations. This problem can be seen in trace files generated by ns2. This bug affects simulation results, such as the throughput. To fix the problem, one can implement a full-duplex IP layer by adding another class that is similar to the RTR or just skip the IP header handling part in the RTR protocol when a user packet is processed. In our simulation experiments presented in Chapter 7, we simply skip the IP header handling part in the models for RTR protocols.

The bugs discussed in this section are for ns2 only. However, if some other simula-

tors port the corresponding sections of code from ns2, these two bugs may also appear in those simulators. The next section discusses some issues regarding the fair comparisons of different MANET routing protocols. This discussion applies to all MANET simulators.

6.2.4 Fair Comparisons among MANET Routing Protocols

To compare different MANET routing protocols, one should use the same and reasonable assumptions for all protocols that are studied. However, current models of different routing protocols do use different assumptions. For example, a MANET routing protocol known as AODV holds outgoing user packets in a buffer for up to a certain maximum holding time before the packet is sent or dropped [50]. AODV needs time to determine a route if there is no known route to the destination. As long as the route is discovered within the maximum buffer holding time, this user packet can be delivered even if the packet is generated at a time when there is no known path between the source and the destination nodes. However, models of some other routing protocols, such as OLSR, do not have this kind of buffer. In OLSR, a user layer packet is dropped immediately if the node's routing agent has no route for it. Therefore, AODV and other protocols that have this buffer can have, potentially, higher throughput than OLSR and other protocols that do not have this feature. Our recommendation is to install the same buffer scheme in all MANET routing protocols so that they can be compared fairly. Note that in ns2, the processing procedure for data packets originated from a node, as we described above, is different from that for forwarding data packets. The buffer can also be applied to the procedure for forwarding data packets. In this dissertation, we do not investigate whether we should use the buffer for forwarding data packets, but ensure that all protocols use consistent assumptions. In our simulation experiments discussed in Chapter 7, we only incorporate the buffer at senders, not at intermediate nodes, for all investigated MANET routing protocols.

Another issue that may lead to unfair comparisons of different routing protocols

is the existence of transit loops. This problem occurs when a node receives a packet from a neighbor and the next hop is the neighbor asked for the forwarding. If this loop exists for a transit time and then disappears after the routing protocol determines the correct routes, we call it a transit route loop. It is not a desirable scenario that a packet can be sent back and forth between two neighboring nodes until a correct route is found. Transit route loops are likely to occur in a MANET, especially with proactive routing protocols. However, there is no loop route detection implemented at intermediate nodes in ns2 for MANET routing protocols such as AODV-UU [107], the recommended implementation of AODV, and OLSR [108] (The implementations only check transit loops at source nodes). Therefore, to compare different protocols fairly, a loop detection scheme that can check for routing loops for user packets should be implemented for all routing protocols. If a routing loop is detected, the corresponding user packet should be dropped immediately.

There are also bugs in the ns2 models of different protocols. For example, in the ns2 model for OLSR, a user packet is dropped when a transit loop is detected at the sender. However, this packet is still forwarded based on the local routing table after the drop function according to the ns2 OLSR source code [108]. Another example is the recommended AODV-UU implementation [107]. In this implementation, all timer events are global variables and are referenced by pointers. The timer event list is a bidirectional global list. When some timers expire, a temporary timer list is generated containing all expiring timers removed from the global timer list. The events in this temporary list are then processed one at a time until all are handled. It is possible that an expiring timer's handler inserts some other expiring timers back into the global timer event list before they are processed. The temporary list may be broken due to removal of expiring timers according to the source code of AODV-UU. In this case, the simulation program either experiences a segment fault or loses all the events behind the broken point in the temporary timer list. As a result, the performance of the AODV protocol may be degraded because some route requests cannot be processed and corresponding user packets are delayed or dropped accordingly. The third example is the time calculation problem in

AODV-UU 0.7.2. The time is exchanged using the timeval structure. However, the function in AODV-UU that converts a simulation time which is a double variable to a timeval variable uses a long variable to pass the simulation time. The long variable has a range limitation. When the simulation is longer than $2,147$ seconds, the long variable becomes negative because of overflow. For simulations that last longer than $2,147$ seconds, AODV-UU may malfunction. If a simulator other than ns2 ports the implementations of these protocols, they have the same problem.

We discussed fair comparisons among different MANET routing protocols in this section, focusing on the ns2 simulator. If one studies the performance of a protocol that relies on the underlying MANET routing protocol or if one uses other network simulators that use similar models, the results may not be valid.

6.2.5 Conclusions and Summary

We summarized several problems in the simulation of MANET routing protocols in this section. These issues include the definition of group mobility models, implementations of ARP and IP in ns2, buffer support to improve throughput, transit route loop detection, and some other implementation bugs. Some problems must be fixed before one can carry out valid simulation experiments. Some issues should be addressed before one can fairly compare different MANET protocols. All these issues are important for simulation of MANETs, especially for routing and upper layer protocols. In this section, we also described our approaches with respect to those bugs and fair comparison assumptions for our simulation experiments summarized in Chapter 7.

6.3 A Dynamic Switch for Emulation of MANET Topologies

As for traditional wired networks, experimental test beds are valuable tools for studying the performance and behavior of routing and other protocols for MANETs. Test beds enable researchers to realize and investigate real implementations of protocols and applications. However, deploying a real MANET test bed can be expensive and time consuming. Another significant concern is that a test bed using real mobile nodes is hard to control. It is difficult to "replay" node movements and to ensure equivalent channel conditions to repeat controlled experiments. Emulation is an efficient approach to solve these problems.

It is relatively easy to set up a traditional wired network in a research laboratory. While such an environment may be suitable for initial development of protocols intended for a MANET environment, the fixed topology, low packet error rate, and high data rate of the wired network do not represent a typical MANET environment which is characterized by a dynamic topology and wireless connections with relatively high error rates and relatively low data rates. Traditional switches for wired networks, such as Ethernet switches, Asynchronous Transfer Mode (ATM) switches, and IP routers, rely on multiple access control (MAC) or Internet Protocol (IP) address information to determine forwarding and we cannot alter the emulated connectivity without altering MAC or IP level addressing. Further, such switches often cannot emulate the effects of packet loss or data rate limitations. Thus, we cannot directly use a wired network and a traditional switch to emulate a mobile ad hoc network.

To meet this need, we developed a special switch that connects multiple nodes according to a controllable dynamic topology with a controllable packet error rate and a controllable data rate on the links. Our dynamic topology switch is implemented in the Linux operating system and includes modifications to the Linux kernel. The switch

emulates a MANET using standard Ethernet or other wired physical connections and requires no changes to the network's nodes.

Our primary objective was to create a reasonable emulation of a MANET environment that required no changes to the "mobile" nodes. We want to test different types of mobile nodes, including nodes running proprietary operating systems. We also want to make the emulation transparent to the real protocols. This transparency includes both functional transparency as a first priority and performance transparency, at least to the extent permitted by the emulated environment, as a second priority. This requires that the switch achieve high performance to match wireless link data rates, including for new higher data rate wireless local area network standards such as IEEE 802.11a [5] and IEEE 802.11g [114]. We also wanted to use standard "off-the-shelf" personal computers for the switch and, clearly, needed an open source operating system.

In Section 6.3.1, we describe models to emulate the topology changes, bit error rate, and data rate of a MANET environment using a wired network. In Section 6.3.2, we discuss the implementation of the model as the dynamic topology switch. The partial validation of this emulator through comparisons with ns2 simulation results for the OLSR MANET routing protocol is presented in Section 6.3.3. Section 6.3.4 compares our dynamic topology switch to related work in network emulation. Section 6.3.5 presents conclusions and future work.

6.3.1 Model Description

This emulator has three functions, emulation of a dynamic topology, emulation of packet loss, and emulation of constrained link capacity, as discussed in Sections 6.3.1.1, 6.3.1.2, and 6.3.1.3, respectively.

6.3.1.1 Emulation of a Dynamic Topology

In a wireless network, a node can transmit directly (in one hop) to another node only if the sending node is within a certain range of the receiving node. Because nodes in a MANET can be mobile, the connectivity of the network can change at any time. Conceptually, this dynamic connectivity can be described by a function of time and location, as shown in Equation 6.2.

$$C_{i,j}(t) = F_e(< x_i(t), y_i(t) >, < x_j(t), y_j(t) >) \tag{6.2}$$

Here, $C_{i,j}(t)$ represents the status of the connection between nodes i and j. If nodes i and j are connected, then $C_{i,j}(t) = 1$. If they are disconnected, then $C_{i,j}(t) = 0$. The coordinates of node i at time t in two-dimensional space are $< x_i(t), y_i(t) >$. Function $F_e(\bullet)$ maps the coordinates of two nodes to a connection status value for a given environment e. $F_e(\bullet)$ is a complex function that depends on a variety of factors, such as radios, anntenas, coding, transmit power, capture effects, terrain, and atmospheric conditions. In Section 5.1.2, we used the M array to represent the link connectivity between pairs of nodes. The elements in the M array are, in fact, the outputs of the connectivity function for pairs of nodes. In other words, different $C_{i,j}(t)$ functions result in different distributions for $M_{i,j}$ in the M array. The dynamic topology switch does not evaluate the connectivity function, but, rather, relies on an external source to specify the connectivity, $C_{i,j}(t)$, for all pairs of nodes, i and j, as a function of time. This information can be derived from a mobility simulation, which has been our approach, or by some other trace file that might be based on measurements of a physical system or derived in some other way.

The basic concept of operation for the dynamic topology switch is to control the connectivity of "mobile" nodes using the central hub in a star network. Figure 6.4 shows a simple example with three mobile hosts and a single dynamic topology switch. The mobile nodes can be any device running any software, as long as they have an appropriate network interface card. The switch is implemented using a standard personal computer running Linux. It has multiple network interfaces realized by using multiple interface

cards and/or one or more multiple-port interface cards. Implementation details and related performance issues are provided below.

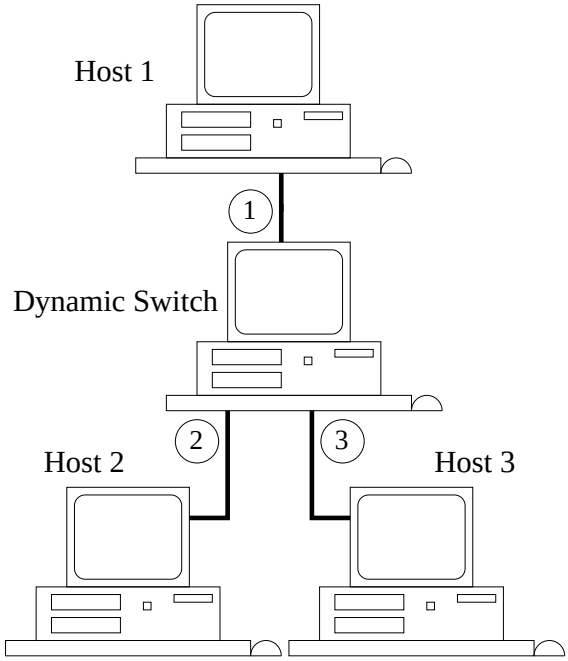


Figure 6.4: Example test bed topology.

The dynamic topology switch can switch traffic between any set of connected hosts based on a local switch connectivity table that can change dynamically. The switch is transparent to all the other nodes at and above the MAC layer. All incoming frames are switched based solely on the input interface and the switch connectivity table information. The switch does not alter the MAC frame or IP datagram information in any way and, in particular, it does not add any address information of its own to the MAC frame or IP datagram.

Discussed next is an example wireless ad hoc network realized by a wired test bed using the dynamic switch. The topology of the wireless ad hoc network is shown in Figure 6.5, where Nodes 1 and 3 are connected to Node 2, but not to each other.

Table 6.1 shows the switch connectivity table used in the dynamic switch. Note that the ports in the table denote the network interface ports of the dynamic topology

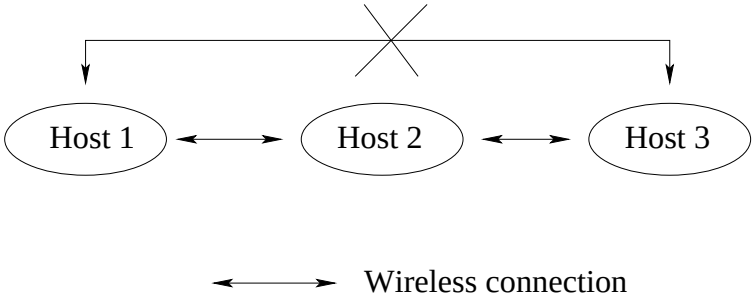


Figure 6.5: Wireless ad hoc network realized by the switch table of Table 6.1.

switch (See Figure 6.4). In other words, the switch relies on incoming or outgoing network interfaces, not the MAC or IP addresses, to specify forwarding. With this switch table, the wired test bed realizes the wireless ad hoc network shown in Figure 6.5.

Table 6.1: Example of a Switch Table

Incoming port	1	2	3
Outgoing port(s)	2	1 and 3	2

The dynamic topology switch can update the switch connectivity table in real time. Thus, we can, in effect, generate a sequence of switch connectivity tables to emulate the connectivity of a mobile ad hoc network in real time.

6.3.1.2 Emulation of Packet Loss

Mobile ad hoc networks are implemented using wireless links where packet loss due to bit errors may be likely. In the dynamic topology switch, we control the packet loss rate for each connected channel. We use the Gilbert model [115], a two-state Markov model, for packet loss.

In the two-state Markov model, a channel can be in one of two possible states, "good" or "bad." The state transition diagram is shown in Figure 6.6. The probability of dropping a packet, i.e., the probability of a packet error, is different in each state. Given

a present state, a channel may transit to the other state or stay in the present state with certain probabilities. P_1 and P_2 are the probabilities of staying in the good and bad states, respectively. $1 - P_1$ is the probability of a transition from the good to the bad state and $1 - P_2$ is the probability of a transition from the bad to the good state.

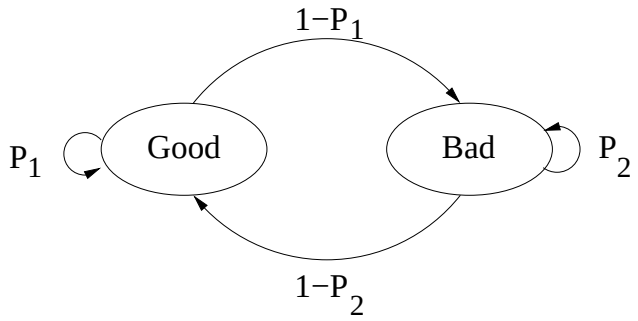


Figure 6.6: Two-state Markov chain for the packet loss model.

6.3.1.3 Emulation of Constrained Capacity

The capacity of wireless links may be constrained and, in particular, may be less than the capacity of the wired links used in the test bed. We enforce constraints on available bandwidth using a leaky-bucket token buffer model.

In the leaky-bucket token buffer model, no packet can be sent unless there is a token in the token buffer or a new token arrives. There is an upper bound on the size of the token buffer. We use a token arrival rate of r tokens per second, a token buffer size of B tokens, and an allowable transmission size of μ bytes per token to determine the bandwidth constraint [116]. Equation 6.3 specifies the maximum allowable data rate, where C is the transmission rate or emulated capacity.

$$C = \mu \times r \; bps \tag{6.3}$$

Since the emulated system does not accumulate transmission "credits," i.e., there is no history, we use buffer size $B = 1$. Selection of r involves a tradeoff between accuracy and

processing overhead. A high token arrival rate r results in transmissions being spread out over a longer interval that more closely mimics a low data rate link. However, minimum timer intervals supported by the operating system of the switch node and the overhead timer interrupts drives us to reduce r. Based on tests, we found a suitable token arrival rate to be $r = 1,000$ tokens per second. Thus, the allowable transmission size per token is selected to be $\mu = C/1000$ bytes.

6.3.2 Implementation

We developed the dynamic topology switch in Redhat 7.0 with Linux kernel 2.2.16 and Redhat 9.0 with Linux kernel 2.4.20. The source code is under GNU copyright protection.[1] Users need to have root privilege to install the code on a Linux system. It requires re-compilation of the kernel.

The software can be divided into three parts: user space program, broker program, and kernel space program. The user space program is responsible for interactions with users. The kernel space program handles kernel interrupts and received packets. The broker program contains a character device driver, which is used to exchange information between user space and kernel space.

The user space program first translates user inputs or command files into the proper command format. The translated commands are written into the character device in the broker program. If users require debugging information from the kernel, the user space program sends the corresponding command to the broker program and reads returned data from that character device.

The broker is a module that can be loaded in super-user mode. The broker creates a character device and sets all network interfaces to promiscuous mode during initializa-

[1]The software for the dynamic switch can be downloaded from www.sourceforge.net/projects/dynamic-switch.

tion. The broker program also maintains the switch connectivity table and token buffer queues. It continues to listen for input/output interrupts from the character device and calls the proper procedures to handle requests from the user space program. This allows users to use commands or input files to control the dynamic topology switch in real time. The broker is also responsible for moving outgoing packets into the proper buffers of the network devices.

The kernel space program deals with packet capture and dynamic forwarding. Once a packet is captured, the kernel procedure enters the dynamic switch block if the character device driver is loaded. The kernel space program looks up the outgoing port(s) for each incoming packet in the switch connectivity table via the broker program. The switch does not examine packets, but they are duplicated, if necessary, so that one incoming packet can be delivered to multiple output ports. The kernel space program forwards packets to the proper device(s) using the sending procedure in the broker program.

6.3.3 Verification of a Test Bed with the Dynamic Switch

In this section, simulation experiments to verify test bed operation using the dynamic switch are presented. To at least partially determine the validity of the dynamic topology switch for use in network performance studies, we compare measured values obtained using the switch to those produced by ns2, a widely used network simulator. In particular, we compare results from an actual implementation of the OLSR protocol [53] running on four "mobile" nodes connected via the dynamic switch to results from an OLSR simulation model running in the ns2 network simulator [98] for the same configuration.

6.3.3.1 Description of Experiments

Wireless routing protocols can be classified as either proactive or reactive. Mobile nodes in a proactive routing protocol periodically broadcast "hello" messages and link state

changes. Mobile nodes in a reactive protocol find a route to another node on-demand when that node is the destination of a data packet. OLSR, the protocol considered here, is a proactive protocol. The authors of the OLSR protocol distribute both a Linux implementation and an ns2 model [127], which we believe adds confidence that the real implementation and the ns2 model are consistent.

Our dynamic topology switch only emulates topology changes and wireless channel properties, specifically the bit error rate and link capacity. Therefore, we use control message overhead as the basis for comparing results from the emulation and the simulation. This metric is independent of the specific underlying MAC protocol and should be consistent across the two realizations.

Parameters for the Linux implementation of OLSR are the same as those for the ns2 model of OLSR except for jitter time. The ns2 model of OLSR introduces jitter to slightly randomize the time at which control packets are generated to reduce the likelihood of MAC-level collisions. Without jitter, the tight synchronization of nodes in a simulation model would result in multiple nodes attempting to transmit at the same time, thus leading to pessimistic performance because of an increased number of collisions. In a real network, including the network emulated by the dynamic topology switch, nodes are not tightly synchronized. Thus, the jitter parameter in the ns2 OLSR model accounts for the jitter that occurs implicitly in a real system. The jitter parameter for the ns2 model is set to 0.1 seconds based on observations from the Linux implementation.

We use the same mobility assumptions in both the dynamic topology switch and the ns2 model. The mobility model we used is a variation of the random waypoint model, which is similar to the one recommended by Bettstetter [92]. It considers a four-node network, with the mobile nodes moving in a 100×100 unit square map. (All length and velocity parameters are normalized to "units.") Nodes start at random positions within this area. Each node moves at a random speed for a random length of time. Both the speed and the duration of the movement are exponentially distributed. Nodes pause for a

constant time when movement ends. We assume that the previous direction of movement for a node is θ. Its next direction of movement is chosen uniformly from $[\theta - \alpha, \theta + \alpha]$, where α degrees is the maximum change (or "delta degree") in the direction of movement. Following the work of Bettstetter [92], we allow nodes to bounce at the borders instead of wrapping around or leaving the network. The radio range of a node is used to decide the connectivity between all pairs of nodes. As described below, we vary the radio range and examine its effect on control message overhead.

There are five parameters that characterize mobility with this model: average speed, average moving time, fixed pause time, α, and radio range. Initial experiments showed that changes in the radio range have the most significant effect on the total number of control messages sent by the nodes. Therefore, we experimented with scenarios with different radio ranges from 20 to 90 units. All parameter values are shown in Table 6.2.

Table 6.2: Mobility Parameters Used for the Experiments

Pause time	10 seconds
Average speed	20 units/second
Average moving time	10 second
α	0.0001 degrees
Radio range	20-90 units

The simulation time for all runs is 300 seconds. A simulation time of 600 seconds shows similar results. Five replications were run for each set of parameters, with the random seed set to 1, 2, 3, 4, and 5.

6.3.3.2 Validation Results

The four graphs in Figures 6.7, 6.8, 6.9, and 6.10 show results from both the ns2 simulation and the dynamic switch-based emulation for each of the four nodes. Each point

represents the average of the values from the five replications.

For OLSR and other MANET routing protocols, especially those that are proactive, the number of control messages that are sent increases as the topology changes more frequently. When the radio range is small, say 20 units, or large, say 90 units, the ns2 simulation and the switch-based emulation show that the number of control messages is relatively small and, thus, the network topologies change infrequently. For short radio ranges, nodes are almost always disconnected from their neighbors, i.e., there are few viable links, and mobility leads to few changes in the topology. For long radio ranges, nodes are usually connected to other nodes and extreme movements are needed to break a link. When the radio ranges are from 40 to 80 units, the network topology changes more frequently and both the ns2 simulation and switch-based emulation results indicate that more control messages are sent.

We observed the number of control message sent at every node. The observed results for Node 1 are listed in Table 6.3. We observed similar results at the other nodes. The difference, as a percentage, between results for the ns2 simulation and the switch-based emulation is calculated as the difference between the number of control messages reported by each method divided by the number of control messages reported by the ns2 simulation.

Table 6.3: Comparisons of Results from ns2 and Test Bed

Radio range	N_{ns2}	$N_{testbed}$	Percentage Difference (%)
20	148.4	150	1.08
30	155.6	152.2	2.19
40	162.8	156.6	3.81
50	167.2	162.8	2.63
60	170.4	159.2	6.57
70	166	158.2	4.70
80	161.2	152.4	5.46
90	153	149.8	2.09

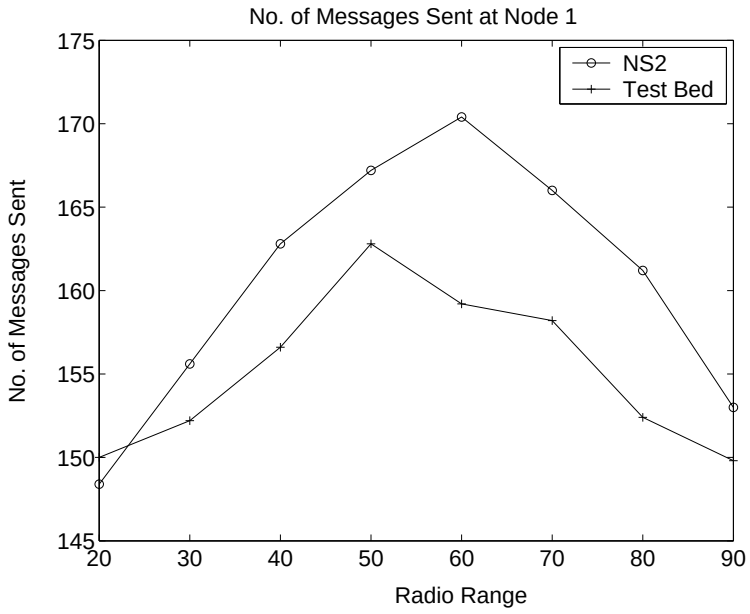


Figure 6.7: Number of control messages versus radio range at node 1.

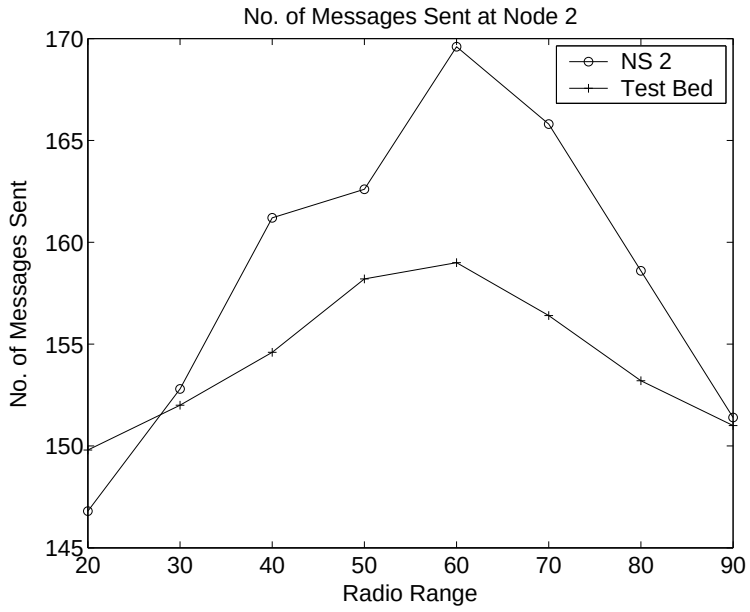


Figure 6.8: Number of control messages versus radio range at node 2.

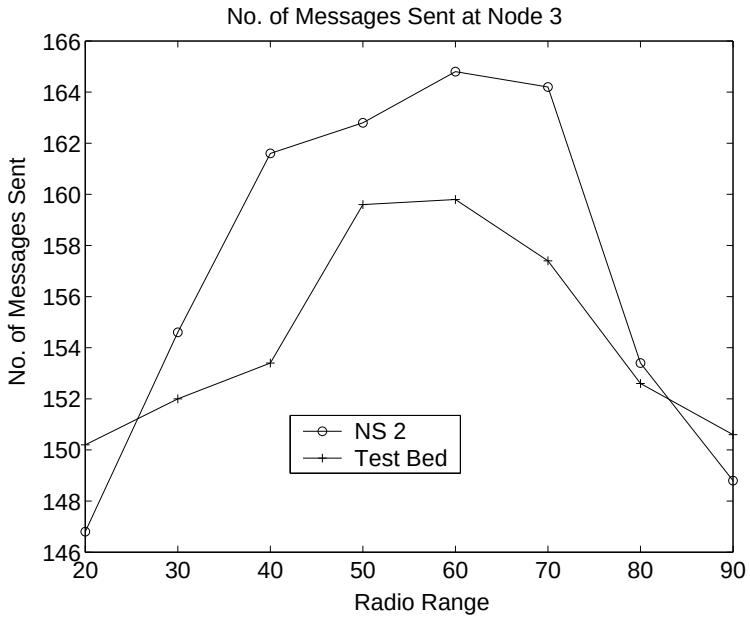


Figure 6.9: Number of control messages versus radio range at node 3.

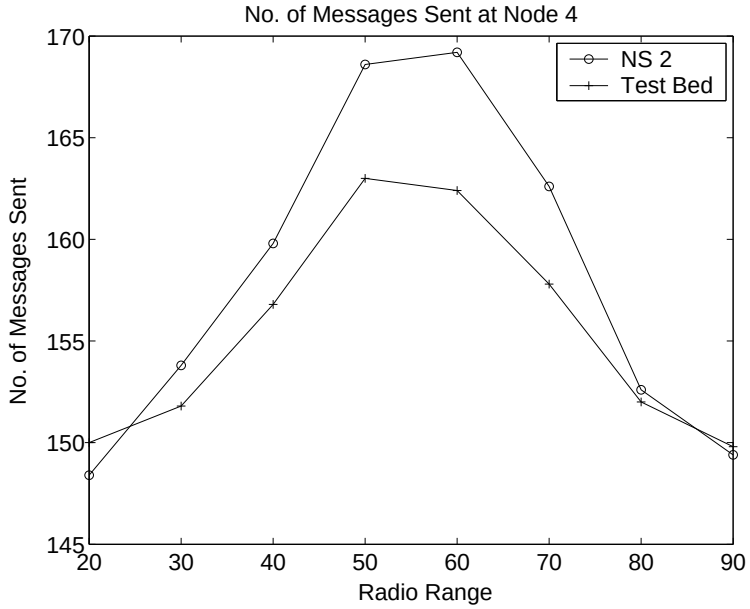


Figure 6.10: Number of control messages versus radio range at node 4.

The data shown in this table indicates that the percentage difference is less than 7 percent for all values of radio range. This small difference may be accounted for by the jitter parameter setting in the ns2 simulation model or by practical differences between the real implementation and the ns2 model of OLSR. For example, all nodes are identical in ns2, while processing delays may not be the same for all nodes in the switch-based emulation. Based on the close correspondence between results for the dynamic topology switch and the ns2 simulation, we have a high level of confidence that the switch is accurately emulating the topology of a mobile ad hoc network.

As indicated in Section 6.3.2, we use Redhat 7.0 with Linux kernel $2.2.16$ and Redhat 9.0 with Linux kernel $2.4.20$ as the operating systems in the dynamic topology switch. We use three ZNYX network cards [128], with each card containing four 10-Mbps Ethernet ports. Testing showed that one Linux node can support up to 11 ports. The actual performance that is achieveable by the dynamic topology switch depends on the specific hardware configuration, including factors such as processor clock rate, bus throughput, and memory size. Since we are studying network layer protocols in MANETs, the current implementation of this dynamic switch should be able to meet the requirements for MANET emulation.

In addition to using ports to connect mobile nodes, the dynamic topology switch is able to receive normal network traffic on a designated interface. This allows us to exchange messages with mobile nodes to synchronize the overall system for testing purposes. For example, to generate repeatable experiments with reactive routing protocols, we need to synchronize topology changes at the switch with startup activities at the mobile nodes. Using this approach, we can design repeatable scenarios for emulations of different MANET routing protocols, including both reactive and proactive protocols.

6.3.4 Comparison to Prior Work

There are other approaches with somewhat differing objectives that, like our approach, combine physical implementations of protocols with network emulation in different ways.

Nguyen, et al. [117] collect traces from physical systems for the purpose of modeling the behavior of wireless channels. Simulation is used to compare results using the traces to results using the derived model. The objective was to build a wireless channel model for use with simulation, which is different from the objective our work in that we want to emulate the underlying network and utilize actual nodes running actual protocol stacks.

Noble, et al. [118] extend the work of Nguyen, et al. [117] to create an approach they call "trace modulation." Traces are first collected from a physical system. The traces are then distilled to build a network model that is representative of the physical wireless mobile ad hoc environment that was measured. Finally, the distilled model is used to modulate the behavior of the protocol stack in a physical system. A modulation layer is inserted between the IP and Ethernet layers. The modulation layer delays and drops packets according to the model derived from the trace. A special process running on each node supplies the modulation layer with time-varying parameter values. This approach has been shown to be effective for evaluating throughput, but temporal ordering is affected so it is not useful for considering detailed effects of latency [119]. Trace modulation within the protocol stack can provide higher fidelity than our system, but requires alterations of the operating system kernel of the mobile nodes. With our centralized dynamic switch, we move the locus of control for modulation outside of the mobile nodes, but do lose some fidelity in the process. However, extensions to our approach could provide similar fidelity.

An extension of trace modulation is called "trace emulation" by Johnson [119]. In trace emulation, the trace of the network's behavior is generated through simulation

rather than experiments with a physical system. Generating the trace file through simulation is comparable to our approach of using a mobility simulator to generate a trace file that controls the dynamic topology switch. The generated trace is applied to the modulation layer as in Noble, et al. [118] and, thus, requires modification of the mobile node's kernel.

Johnson also developed a "direct emulation" method [119] that is similar to our approach. As in our system, packets from a real system are sent to a centralized node. The centralized node is running a simulation model that controls packet loss and delay. There is a fundamental trade-off between fidelity and efficiency. Direct emulation provides greater fidelity than our dynamic topology switch, but at the cost of extra overhead that can reduce the supported data rate for each mobile node and/or limit the number of mobile nodes that can utilize the emulated network. Our scheme needs to execute little code to move a packet from an input port to zero or more output ports.

Direct emulation is similar to Fall's use of the ns network simulator for emulation of traditional networks [120] and to work by Xu, et al. where physical elements are integrated with a sensor network simulation running in the GloMoSim simulator [121]. Fall's approach and Xu's approach both require extra overhead to manage the interface between the physical device and the simulation model. For example, in Fall's ns-based system, packets from real systems must be encapsulated as they are processed by the simulator to ensure that all packet information is preserved. In our approach and Johnson's [119], this interface overhead is eliminated.

Chao, et al. developed a test bed using an iptable[2] INPUT chain to control link connectivity [122]. Their approach requires manually configured MAC addresses at end nodes, which is not required by our dynamic switch. Moreover, the MAC header of the packets forwarded by their topology controller is changed. In other words, the topology controller is not transparent to end nodes. Chao, et al.'s model does provide a good GUI

[2]Iptable is software that can filter network packets based on their MAC or IP headers.

for users.

Like Chao, et al.'s test bed, M. Fjällström and J. Nielsen designed and implemented a similar test bed named the ad hoc protocol evaluation (APE) test bed [123]. Different from Chao, et al.'s model, APE uses a PREROUTING chain in iptable [122]. Therefore, this model also requires a manually configuration of MAC information at end nodes and the topology controller is not transparent to end nodes.

6.3.5 Conclusions

A dynamic topology switch was designed and implemented to forward packets based on the incoming network interface rather than on the packet's IP or MAC address. The switch efficiently forwards incoming packets to zero or more outgoing network interfaces that are specified by a switch connectivity table. This table can be dynamically updated so that the switch can emulate dynamic mobile ad hoc network topologies using fixed nodes and a wired network. The dynamic switch also emulates the properties of wireless channels, specifically by dropping packets and limiting link data rates.

The dynamic topology switch allows researchers to experiment with real implementations of full protocol stacks for MANETs without changing the mobile nodes and without impediment by the underlying network that is emulated by the switch. The switch can be used to evaluate routing protocols in terms of routing overhead, average length of routes, and relative latencies. However, the switch does not emulate a wireless MAC layer protocol, so absolute delays and throughput in a MANET routing protocol cannot be accurately measured since these metrics, certainly in absolute terms, are sensitive to the performance of the MAC layer.

6.4 Summary of Study of Simulation and Emulation Tools

In this chapter, we discussed a study of simulation and emulation tools. We investigated a widely-used network simulator, ns2, presenting bugs and problems in fairly comparing MANET routing protocol performance. Simulations discussed in Chapter 7 are based on our work presented in this chapter. To emulate a MANET in a wired test bed within a laboratory, we designed and implemented a dynamic switch. Using the switch, we can verify MANET routing protocols in a wired test bed with controllable and repeatable experiments.

Chapter 7

Simulation and Emulation Study of MANET Routing Protocols

This chapter presents a comprehensive study of MANET routing protocols using simulation and emulation. Simulation results comparing different MANET routing protocols are based on Sections 4.3.3 and 6.2. Section 7.2 briefly describes the integration and interoperatbility of different MANET protocols and applications, using a hybrid testbed using the dynamic switch.

7.1 Simulation Study

This section presents a comparison of MANET routing protocols using simulation. Protocols investigated are OSPF-MCDS, AODV, OLSR, and TBRPF. DSR is similar to AODV except a node list is used in DSR instead of hop count used in AODV. Moreover, there is no further development on the simulation model for DSR since it was first proposed. Similarly, the Internet draft for TORA expired and there is no up-to-date development on simulation model for TORA. Thus, we do not compare DSR and TORA with other

MANET routing protocols in this dissertation. We use ns2 to do the simulations. Several known bugs in the ns2 simulator were fixed and assumptions to ensure fair comparisons were applied to the implementations of different protocols (refer to Section 6.2). We use the latest AODV-UU ns2 implementation for AODV simulations [107].

7.1.1 Comparison Models

As mentioned Section 5.1, we developed a two-state connectivity (TSC) model as an alternative to traditional mobility models. We use both the TSC model and the random waypoint model in our experiments.

We use a two-state on/off UDP traffic model in our simulation experiments. The time for a UDP flow to be in the ON or OFF state is exponentially distributed. When the flow is in the ON state, fixed-size UDP packets are sent from one node to the other with constant intervals between packets. The flow stops when the flow is in the OFF state. We use the ON/OFF moodel because using only constant bit rate (CBR) traffic favors reactive protocols such as AODV and, CBR traffic is probably not a realistic traffic model for typical MANET applications.

This section discusses simulation models that we use to compare different routing protocols. The next section presents simulation results and draws conclusions.

7.1.2 Simulation Results

In our simulation experiments, we consider four metrics, capacity consumed by control messages, percentage of packet delivered, average hop count, and average end-to-end delay. The capacity consumption is defined as the size of the overall routing control messages at the MAC layer sent in one unit of time. Simulation parameters are set as follows. The user UDP packet size is 1,000 bps with a 1-second interval between packets. The

average time for a traffic flow to be in the ON or OFF state is 50 seconds. The interval between HELLO messages for hello protocols used by OSPF-MCDS, AODV, OLSR, and TBRPF is 1 second. We assume AODV does not have link layer notification and uses a hello protocol to detect neighbor changes. A neighbor expires after three missing HELLO messages. Proactive protocols broadcast link state messages every three seconds and a link state entry expires after two missing periodic refreshes. The simulation time is 4,000 seconds, of which the first 2,000 second is the warm-up period. Three scenarios are discussed in this section. For each data point, we have five independent replications. The first two scenarios presented in Section 7.1.2.1 use the TSC mobility model. The random waypoint model is used in the third scenario, which is summarized in Section 7.1.2.2, to verify our conclusions based on the TSC model.

7.1.2.1 Experiments using the TSC Model

In the first scenario, we fix the average link change rate and study the relationship between the performance metrics and the average node degree. There are 30 nodes in the network. Two different traffic loads are used, 5 and 150 UDP flows. A legend for the simulation is given in Figure 7.1.

Figure 7.1: Legend for the following graphs of simulation results.

Generally, when the average node degree increases, the link density becomes higher. Therefore, the percentage of delivery increases, which can be seen in Figure 7.2(a). AODV and OLSR both have better throughput than TBRPF and OSPF-MCDS when the traffic load is low. However, Figure 7.2(b) shows that OLSR consumes the most capacity for control messages compared to the other three protocols. In other words, OLSR trades capacity for throughput.

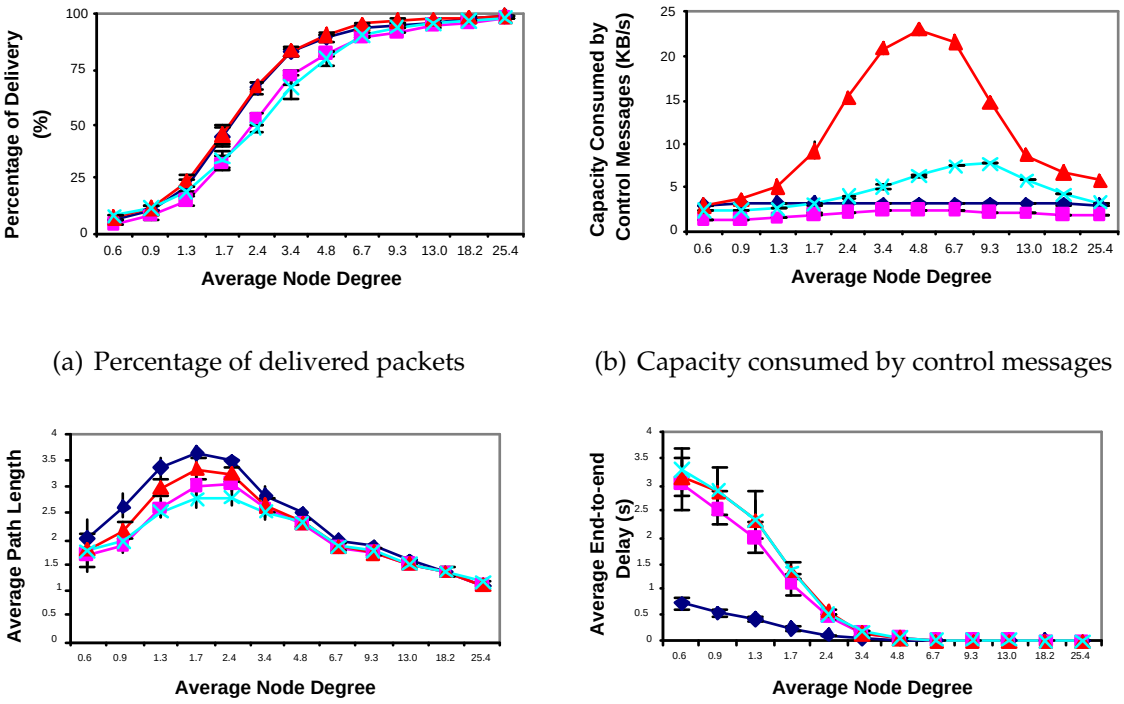
As shown in Figure 7.2(c), OSPF-MCDS has the smallest average hop count because it reports link state changes immediately after link changes are detected. This implies that OSPF-MCDS requires fewer transmissions per user data packet and, thus, consumes less power, on average, to transfer user data. According to Figure 7.2(c) AODV has the highest hop counts because reactive protocols are not sensitive to link-up events. When a new and shorter route is available due to a link-up event, reactive protocols cannot detect it and switch the better route.

Interestingly, we can see that the average hop count in OLSR is always greater than those in TBRPF and OSPF-MCDS, as shown in Figure 7.2(c). It is even longer than those in AODV in some of the following simulation results. We believe that this is due to the delay for distributed topology databases maintained by proactive routing protocols to converge. In other words, distributed link state databases need a certain time to converge following a link state change. The higher average hop count implies that OLSR cannot setup proper shortest paths quickly following a link state change, compared to TBRPF and OSPF-MCDS. This also explains why OLSR has higher throughput than TBRPF and OSPF-MCDS. Because of the longer convergence delay in OLSR, some user packets, which should be discarded, are transmitted to neighboring nodes. This allows a user data packet to remain in the network for a longer time and thus, the possibility that there is a route for this packet to reach the destination is higher. To our knowledge, the feature of holding packets "in the air" is not investigated before. A thorough study of the influence of transient delays for distributed databases to converge in proactive routing protocols could be an interesting future research topic.

The results for average end-to-end delay are presented in Figure 7.2(d). AODV has the smallest average end-to-end delay compared to other protocols since it sends out user packets in a TCP-like way. It first validates an existing path using a hand-shaking process using route request and route reply messages between the initiator and the destination (or an intermediate node that has the route to the destination). After that, AODV keeps using this path unless this path is broken. This guarantees that once a packet is sent, the

end-to-end delay is small. As we can see, the 95% confidence interval for AODV curves in Figure 7.2(d) is small. This feature suggests that AODV can provide predictable end-to-end delays when the average link change rate is or is close to constant. Thus, AODV may be suitable for real-time applications over wireless links in MANETs with relative stable link change rates. The end-to-end delays in proactive routing protocols are usually larger than in AODV except when the number of traffic flows is large. We believe that this is also due to the convergence delays we discussed above. When the number of user traffic flows is large and the link density is high, OSPF-MCDS can provide the smallest end-to-end delay, on average, compared to other protocols (which is shown more clearly in the following scenarios). We believe the reason is that OSPF-MCDS uses a MCDS to broadcast control messages and the nodes in MCDS are not necessary to be the gateways for user data. This reduces the possibility of collisions between user data packets and control packets.

The simulation results with 150 UDP flows are summarized in Figure 7.3. The percentage of delivered packets decreases after the average node degree reaches a certain value as shown in Figure 7.3(a). This is because when the user load and connectivity increase, the possibility of packet collisions becomes higher and then, throughput is reduced. According to Figures 7.3(a) and 7.3(b), OLSR still provides the best throughput among all four protocols and, also, has the highest capacity consumption. AODV's average percentage of packet delivered degrades to that of OSPF-MCDS and TBRPF as shown in Figure 7.3(a). This is because that when the number of traffic flows increases, the number of times that AODV discovers a route increases and, thus, the capacity consumed by AODV increases (see Figure 7.3(b)). Because of the increasing AODV overhead, the possibility of packet collisions increases. Therefore, the user data throughput is reduced. As we can see in Figures 7.2(b) and 7.3(b), the capacity consumption by AODV changes dramatically when the number of traffic flows increases. The reason is that AODV's overhead is more sensitive to the number of traffic flows than other three protocols. Therefore, AODV suffers scalability problems with respect to the number of traffic loads. According



(a) Percentage of delivered packets

(b) Capacity consumed by control messages

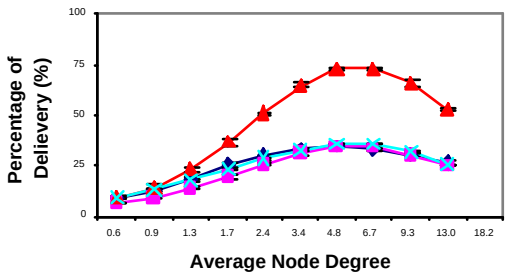
(c) Average path length

(d) Average end-to-end delay

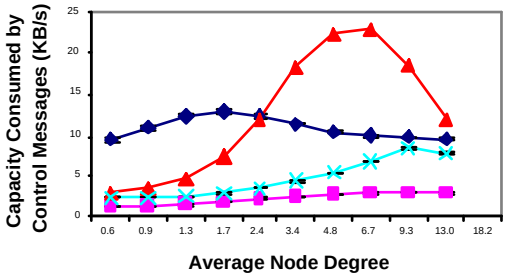
Figure 7.2: Results for a 30-node MANET with 5 UDP traffic flows and constant link change rate using the TSC model.

to Figures 7.2(b) and 7.3(b), TBRPF has the best performance in terms of capacity consumption compared to the other protocols. This suggests that TBRPF is a good candidate for the applications that have limited bandwidth. OSPF-MCDS consumes less capacity than OLSR, while still providing full topology information to all nodes. Therefore, OSPF-MCDS can be used to efficiently support load-balancing and other applications that require full topology information in a MANET with link capacity constraint.

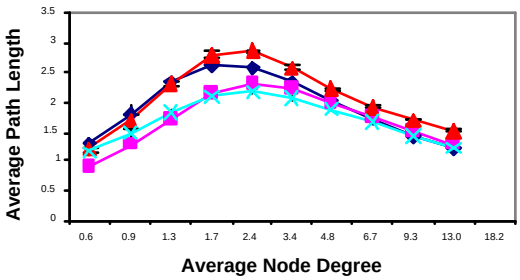
The results for the average hop count and the average end-to-end delay shown in Figure 7.3 are similar to those in Figure 7.2. Therefore, we can draw similar conclusions as discussed in the previous paragraphs.



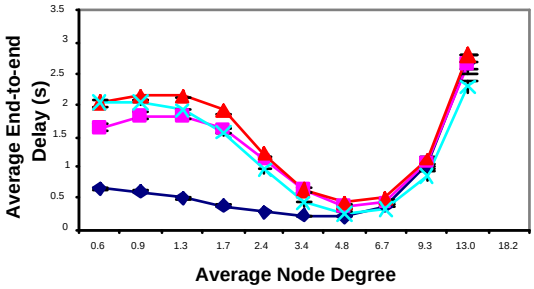
(a) Percentage of delivered packets



(b) Capacity consumed by control messages



(c) Average path length

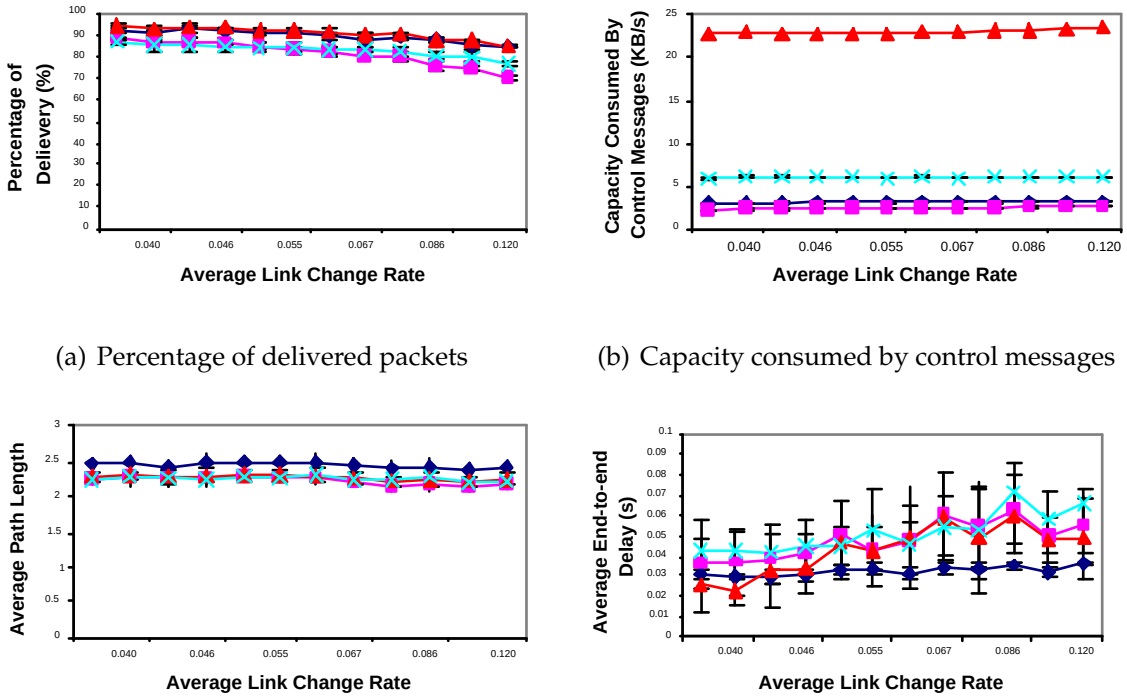


(d) Average end-to-end delay

Figure 7.3: Results for a 30-node MANET with 150 UDP traffic flows and constant link change rate using the TSC model.

The second simulation scenario is similar to the first one except that the average link change rate varies and the average node degree is a randomly picked up constant,

which is 4.78. Figures 7.4 and 7.5 show the results when the number of UDP traffic flows is 5 and 150, respectively. As we can see in Figures 7.4 and 7.5, the percentage of packet delivered decreases when the link change rate increases. The average hop count and the average end-to-end delay do not vary significantly when the link change rate changes. According to the results shown in Figures 7.4(b) and 7.5(b), however, it seems that the link change rate does significantly affect capacity consumption.



(a) Percentage of delivered packets

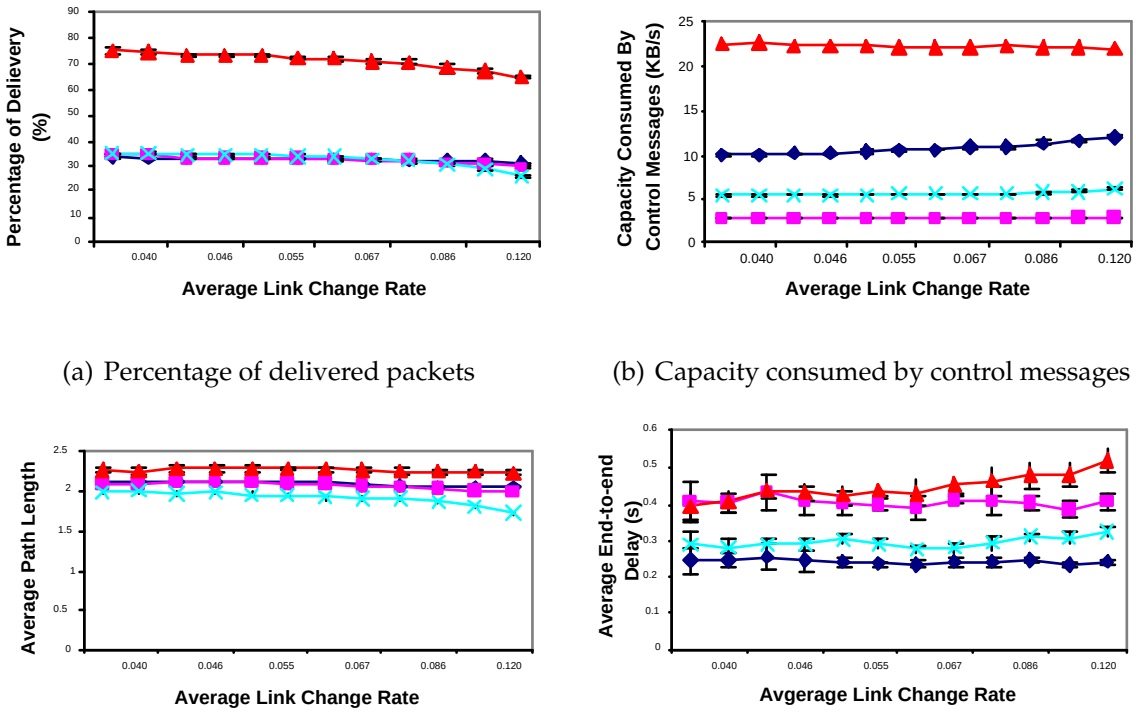
(b) Capacity consumed by control messages

(c) Average path length

(d) Average end-to-end delay

Figure 7.4: Results for a 30-node MANET with 5 UDP traffic flows and constant average node degree using the TSC model.

To more clearly show the influence of link change rate, the percentages of increased capacity consumption as a function of the average link change rate changes are shown in Figures 7.6(a) and 7.6(b). When the traffic load is low, the capacity consumed by TBRPF has the most significant changes mainly because its hello protocol only reports part of a node's neighbor list according to the link state changes. Control message collisions are



(a) Percentage of delivered packets

(b) Capacity consumed by control messages

(c) Average path length

(d) Average end-to-end delay

Figure 7.5: Results for a 30-node MANET with 150 UDP traffic flows and constant average node degree using the TSC model.

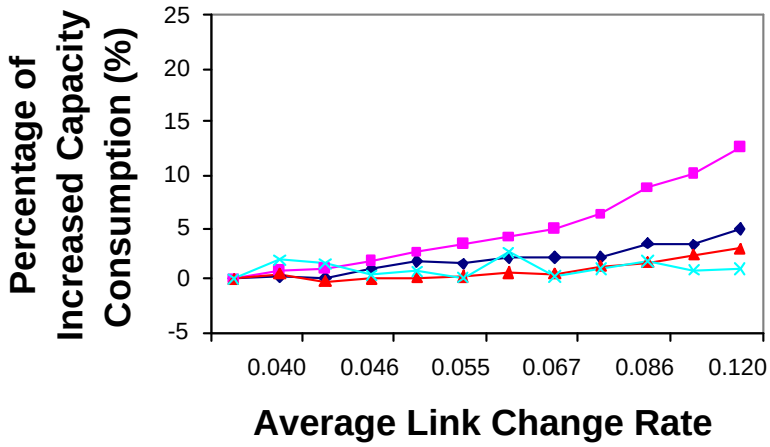
also possible reasons. Similarly, the same reasons also explain results for OSPF-MCDS shown in Figure 7.6(b). Results for AODV in Figure 7.6(b) show that the link change rate affects AODV's capacity consumption the most among all four protocols studied. The reason is that the AODV protocol usually needs to restart a route discovery procedure once a link down event breaks a path being used. When the number of user flows is large, the link change rate can affect the AODV overhead more significantly. The graphs in Figures 7.4, 7.5, and 7.6 suggest that the average link change rate is among one of the most important factors that affect the capacity consumed by AODV, especially with a large number of traffic flows. Figures 7.4 and 7.5 also show that although not significantly, the link change rate does affect the performance of OSPF-MCDS, TBRPF, and OLSR.

To study the impact of the network size, we simulate MANETs with different sizes such as 15, 25, and 35 nodes using the same settings used in the first scenario (we are unable to study larger networks due to limitations on the ns2 trace file size in Linux and Microsoft Windows). The results are similar to those in Figures 7.7, 7.8, and 7.9. Therefore, the conclusions discussed above also apply to the different network sizes that we simulated.

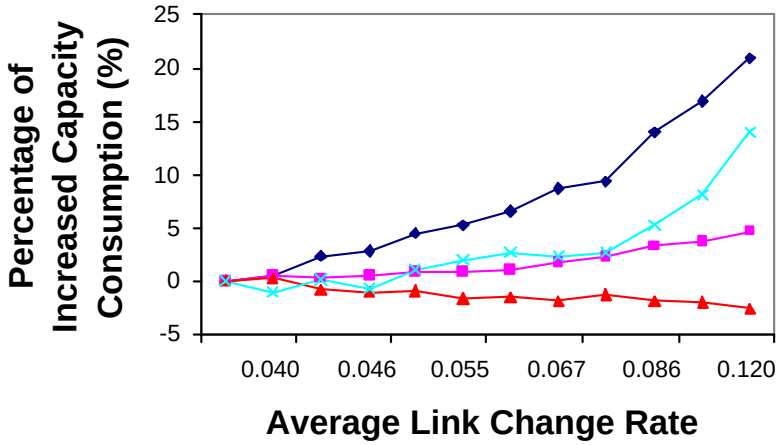
7.1.2.2 Experiments using the Random Waypoint Model

To further verify our conclusions with the TSC model, we use the random waypoint model, a traditional mobility model, in the third scenario. Table 7.1 lists the parameters we used for Bonnmotion [95] to generate the random waypoint trace files.

In this set of simulations, different radio ranges are used since the average node degrees and average link change rates exhibit clear trends with respect to different radio ranges. Figure 7.10(a) shows that the average node degree increases when the radio range increases. This is because a longer radio range implies a larger cover area and, thus, a larger number of neighbor nodes. According to Figure 7.10(b), the average link change rate increases when the radio range increases from a small value, say 40 units.

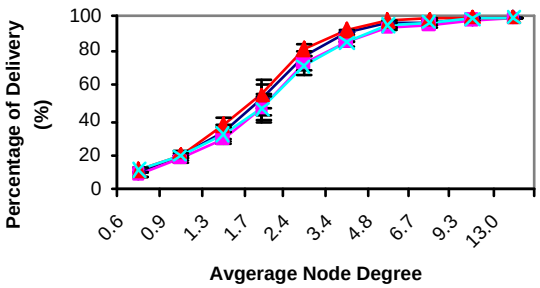


(a) Percentage of delivered packets

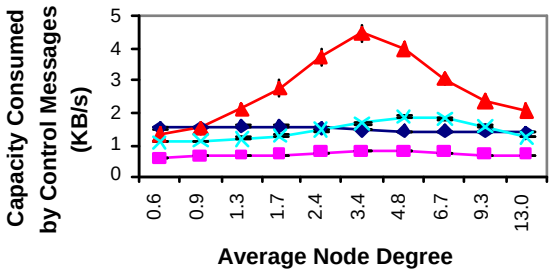


(b) Capacity consumed by control messages

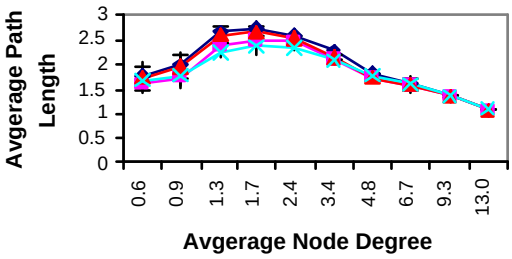
Figure 7.6: Percentage of increased capacity consumption.



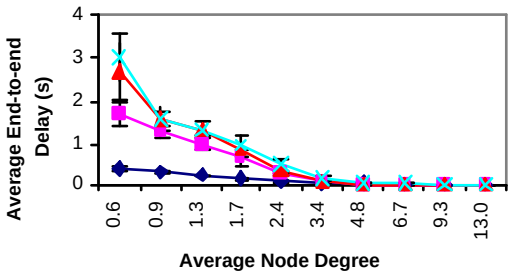
(a) Percentage of delivered packets



(b) Capacity consumed by control messages

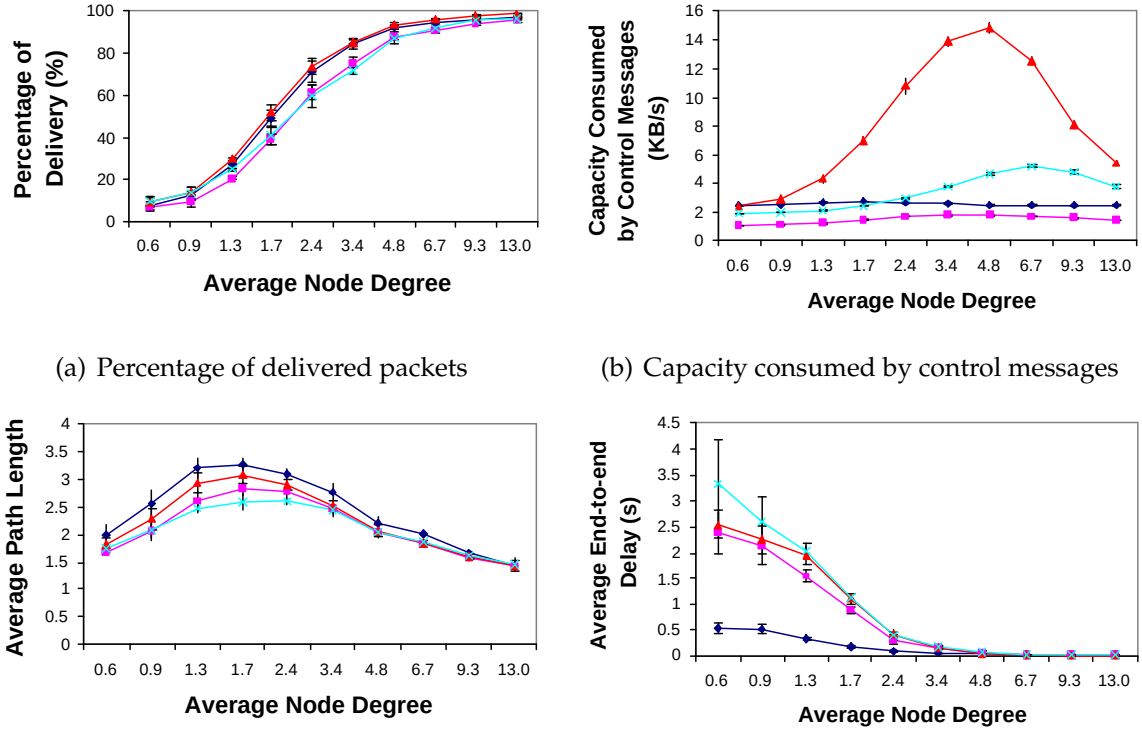


(c) Average path length



(d) Average end-to-end delay

Figure 7.7: Results for a 15-node MANET with 5 UDP traffic flows and constant link change rate using the TSC model.



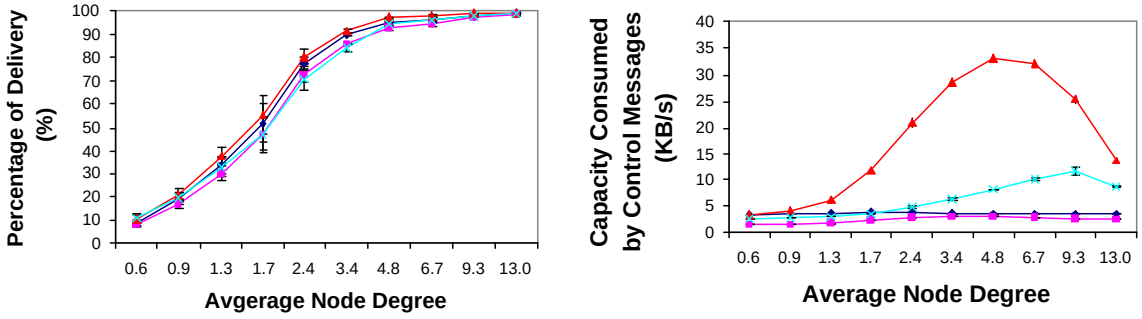
(a) Percentage of delivered packets

(b) Capacity consumed by control messages

(c) Average path length

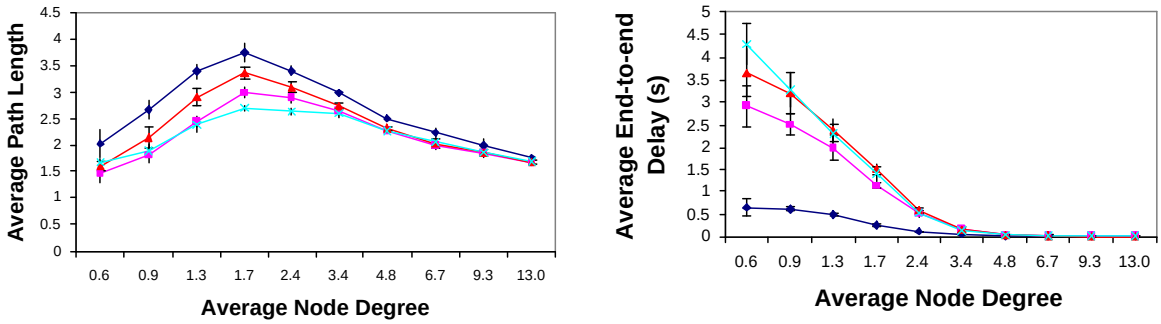
(d) Average end-to-end delay

Figure 7.8: Results for a 25-node MANET with 5 user UDP traffic and constant link change rate using the TSC model.



(a) Percentage of delivered packets

(b) Capacity consumed by control messages



(c) Average path length

(d) Average end-to-end delay

Figure 7.9: Results for a 35-node MANET with 5 UDP traffic flows and constant link change rate using the TSC model.

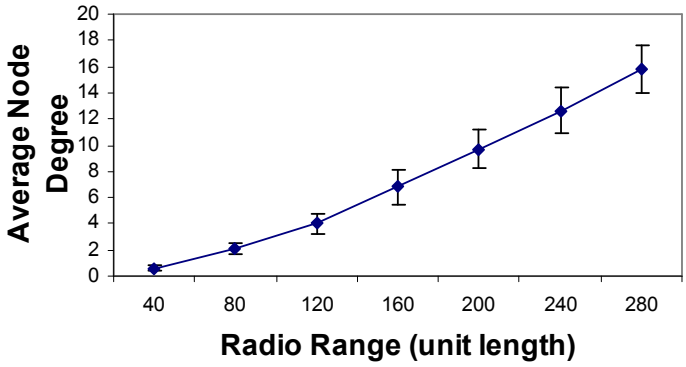
Table 7.1: Random Waypoint Model Parameters

Variables	Values
Map Size	$600 \times 600 unit^2$
Minimum Speed	0.5 unit/s
Maximum Speed	10 unit/s
Warm-up Time	36,000 s
Simulation Time	4,000 s
Number of Nodes	30

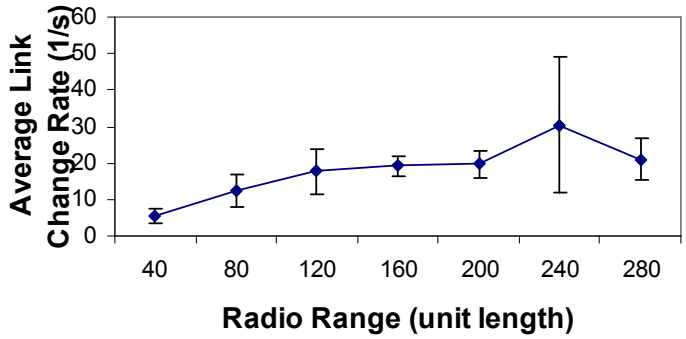
The reason is that the link density increases when the radio range increases. Therefore, the possibility of link changes increases, too. The link change rate reaches a peak value with a certain radio range and after that, the link change rate decreases. This is because a node is unlikely to move out of another's radio range if the radio range is long enough. As shown in Figures 7.10(c) and 7.10(d), the average link up lifetime increases and the average link down lifetime decreases when the radio range increases because nodes are likely to remain connected with a larger radio range.

Figure 7.11 summarizes the results for a 30-node MANET with five UDP traffic flows using the random waypoint model, which are close to the results shown in Figure 7.2. The major reason for the slight differences between Figure 7.2 and Figure 7.11 is the link change rate varies in the random waypoint model while it is a constant in the TSC model. Similar to the discussion of Figure 7.2, we can still draw the same conclusions based on Figure 7.11.

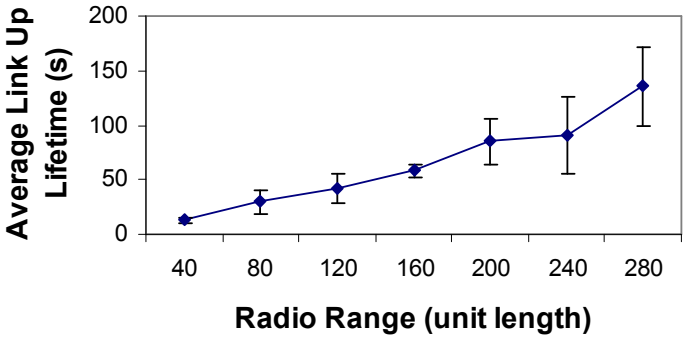
Figure 7.12 shows the results when the number of UDP flows is 150. The conclusions based on Figure 5 also apply to the curves in Figure 7.12. Therefore, our conclusions based on the TSC model also hold for the results using the random waypoint model.



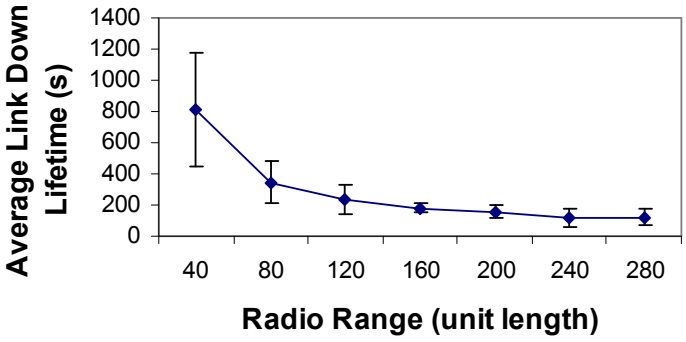
(a) Average node degree



(b) Average link change rate

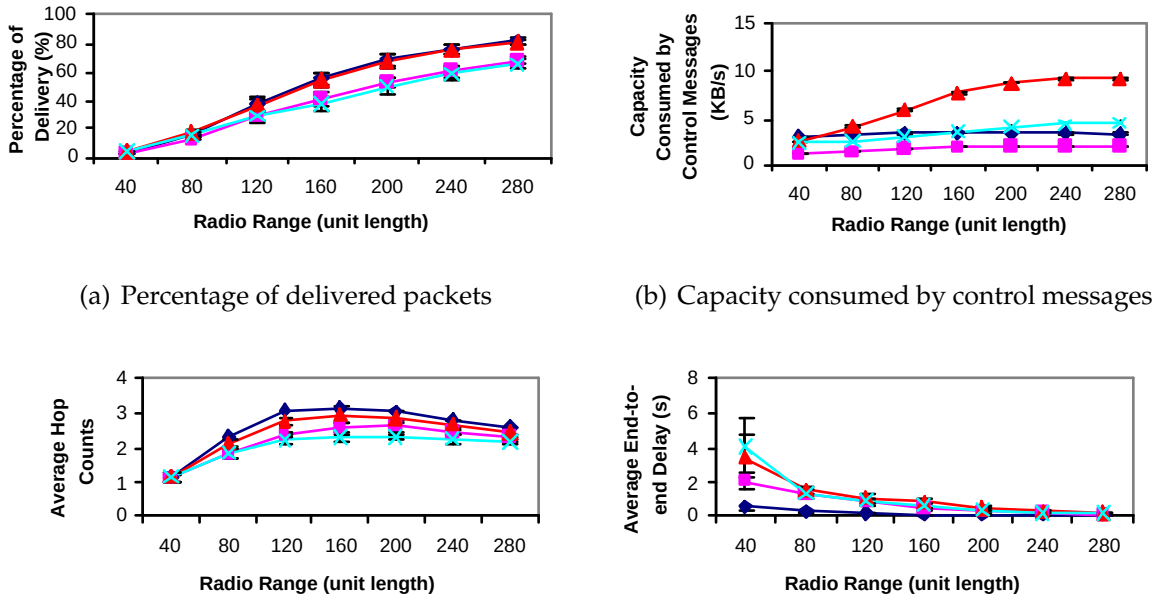


(c) Average link up lifetime



(d) Average link down lifetime

Figure 7.10: Network parameters for the random waypoint model trace files.



(a) Percentage of delivered packets

(b) Capacity consumed by control messages

(c) Average path length

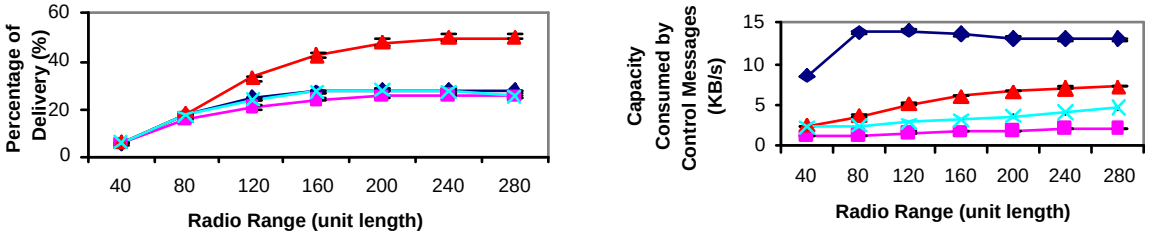
(d) Average end-to-end delay

Figure 7.11: Results for a 30-node MANET with 5 UDP traffic flows using the RW model.

7.1.3 Summary and Conclusions

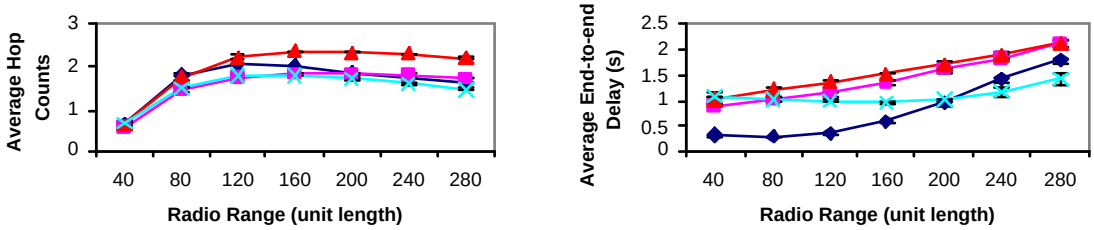
In this section, we presented a comparison of four different MANET protocol, including OSPF-MCDS, our new protocol based on OSPF, AODV, OLSR, and TBRPF. Simulations show that the performance of AODV, in terms of control overhead, is not scalable because it is not only affected by the network environment but also by the user traffic parameters such as the number of traffic flows. The overhead in AODV can be even larger than for the proactive protocols when the number of UDP traffic flows increases. Thus, AODV is not suitable for capacity-limited or power-limited MANETs. The end-to-end delay for AODV is better than the other three protocols when the link change rate is stable and relatively small, which suggests that AODV is suitable for real-time applications over MANETs when the link change rate is stable and relatively small.

The delay for the distributed link state databases to converge in OLSR is longer



(a) Percentage of delivered packets

(b) Capacity consumed by control messages



(c) Average path length

(d) Average end-to-end delay

Figure 7.12: Results for a 30-node MANET with 150 UDP traffic flows using the RW model.

than for OSPF-MCDS or TBRPF according to our results. The throughput of OLSR is higher than for OSPF-MCDS and TBRPF, partially because of the long convergence delay. Therefore, this suggests the potential influence of future research on the convergence delays on performance of different routing protocols.

OSPF-MCDS and TBRPF have similar delays for their distributed databases to converge, which are much smaller than those of OLSR. Therefore, the average hop count in TBRPF and OSPF-MCDS is shorter than for OLSR. OSPF-MCDS and TBRPF's overall performance, unlike that for AODV, is not sensitive to application parameters such as the number of traffic flows. OSPF-MCDS has similar or better performance than TBRPF in terms of hop count, end-to-end delay, and throughput. Although OSPF-MCDS has worse capacity consumption for control messages when compared to TBRPF, it provides full topology information while TBRPF does not. Therefore, OSPF-MCDS is a good candidate for a power-constrainted and capacity-constrainted MANET, especially when the full topology is required by other MANET protocols. Besides, for heavy loaded MANETs in terms of the number of traffic flows, OSPF-MCDS provides small end-to-end delays than for other three protocols. Thus, it is suitable for real-time applications in this kind of MANET.

OSPF-MCDS, like other proactive routing protocols, consumes a certain bandwidth for control messages in spite of traffic flows in the network. Therefore, for low traffic load MANETs, OSPF-MCDS is not a good candidate.

Simulation experiments also show that proactive routing protocol cannot work properly when the link connectivity change rate is close to the frequency of periodic broadcast used by proactive protocols. Reactive routing protocols fail to operate properly when the time for the initiator to receive the route reply is close to the average link up lifetime.

Future work based on this study can include, but is not limited to, improving the performance of OSPF-MCDS by tuning sub-protocols, extending OSPF-MCDS to support

MANET multicast, studying potential security attacks and intrusion detection methods in OSPF-MCDS, supporting Quality of Service (QoS) and other applications in MANETs.

7.2 Emulation Study

To verify the OSPF-MCDS protocol and to study the interoperability among different MANET protocols in a MANET, we implemented OSPF-MCDS protocol in the Linux operating system and setup a wireless and wired hybrid testbed. The Linux model for OSPF-MCDS is based on the ns2 model we developed for the simulation experiments described in Section 7.1. Unlike the model for ns2, the Linux model needs to schedule timer events, capture and send control packets by itself. Multi-thread programming is also required in the OSPF-MCDS Linux model. Since traditional user signal interrupt is not thread-safe [130], time expirations are implemented by Palaniappan Annamalai, a MS student in our group, using "select()" function. Two widely-used libraries for networking applications, Libpcap [131] and Libnet [132], are used to capture and send control messages, respectively. (Raw socket is not used because it is said to have potential long run problem [133].) Recent study shows that Libpcap is not thread-safe [134]. The longer time OSPF-MCDS runs, the higher possibility that libpcap may crash. This is the limitation for current Linux implementation. This problem can be solved by revising the receiving function using new released Libpcap which is thread-safe.

The architecture of the hybrid testbed we setup is shown in Figure 7.13. The testbed contains both wireless and wireline parts. Gateway nodes in the wireless part are connected using 802.11b networking cards. Gateway nodes in the wireline side are connected via the dynamic switch we developed. There are two copies of OSPF-MCDS running in this testbed, one running in the wireless domain and the other is in the wireline domain. Gateway G_6 has both wireless and wired networking cards. It acts as the edge gateway connecting machines in different sides. The edge function that connects

two OSPF-MCDS domains G_6 is implemented by Palaniappan Annamalai too. OSPF-MCDS runs at all gateway nodes, labelled as G_i in Figure 7.13. The default routes at subnet nodes, labelled as S_i, are manually set to corresponding gateway nodes. In the emulation experiments, Gateway G_{23} is the one that has mobility. Originally, the routing table at G_{23} sets G_{21} as the next hop to any other destination in the testbed since G_{21} is the only node that connects G_{23} to other nodes in the network. When G_{23} moves to the new location such that it connects to G_{22} and disconnects from G_{21}, a link-up event and a link-down event are propagated to all nodes in the network by OSPF-MCDS protocol. Then the next hop to any other destination at G_{23} is changed to G_{22} accordingly. We used a video conference software, named VIC [135], to demonstrate the continuous connectivity maintained by OSPF-MCDS between G_{23} and G_6 when G_{23} moves. Video streams can switch smoothly to the new routes after the network topology changes, showing that the connectivity in the MANET is maintained properly by OSPF-MCDS.

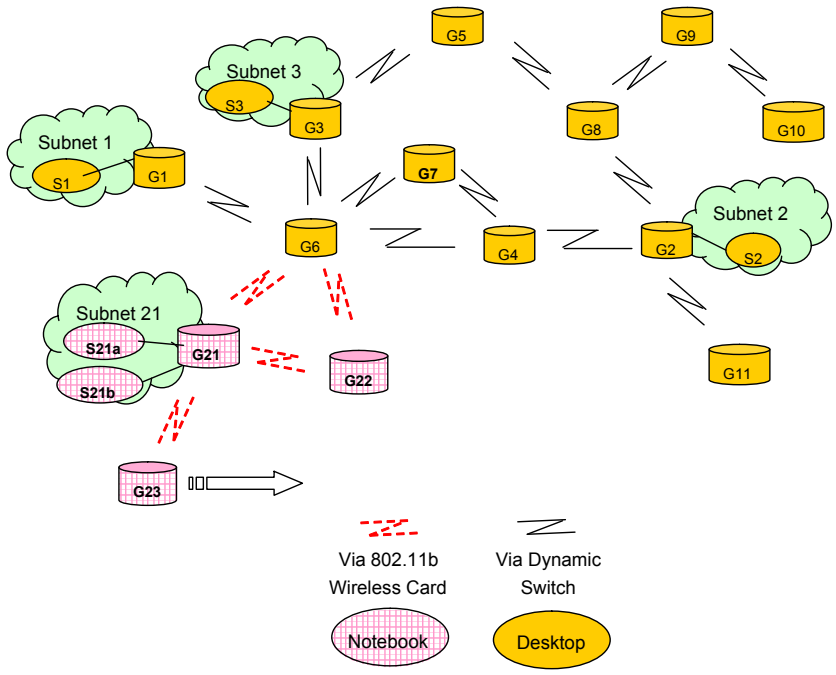


Figure 7.13: The hybrid testbed.

Besides OSPF-MCDS, a Policy Based Network Management (PBNM) Quality of Service (QoS) system, IP Security (IPSec), a real-time application, and Simple Network Management Protocol (SNMP) based network topology monitor are all running in the emulation testbed, which is illustrated in Figure 7.14. This work is summarized in [129]. Basically, OSPF-MCDS maintains the network connectivity in the MANET testbed and provide full topology information for the PBNM QoS system. The communication server in OSPF-MCDS for the PBNM QoS is developed by Palaniappan Annamalai. Communications between any two subnet nodes behind two gateway machines are based on IPSec tunnelling controlled by the IPSec protocol. QoS system is responsible to allocate proper bandwidth for traffic flows with different priorities. The integration results verify that our routing protocol can support backbone routing in this MANET testbed. Potential future work includes extending OSPF-MCDS to support interoperability with traditional routing protocols, such as OSPF, so that a MANET network can also be used to access a wireline network, including the public Internet. Another possible future research topic is to test the interoperability among different implementations of OSPF-MCDS developed by different groups in a testbed.

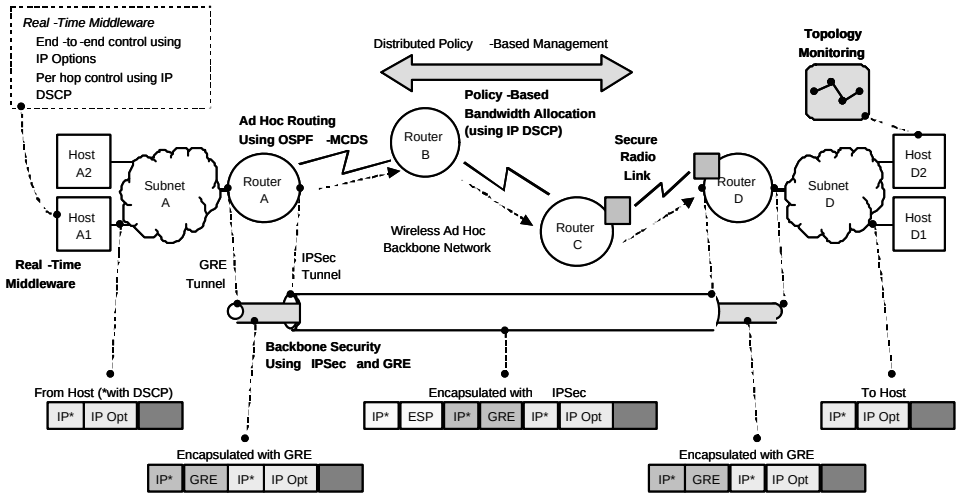


Figure 7.14: Integration of network management, routing, QoS and security in a MANET.

Chapter 8

Summary and Future Work

In this chapter, we summarize our research, highlight contributions, and discuss potential future research.

8.1 Summary

The network topology in a MANET can be dynamic and unpredictable. Traditional routing protocols used for wired networks cannot be directly applied to most wireless networks because some common assumptions are not valid in this kind of dynamic network. For example, one assumption is that a node can receive any broadcast message sent by others in the same subnet. However, this may not be true for nodes in a wireless mobile network. The bandwidth in this kind of network is usually limited. Thus, this network model introduces great challenges for routing protocols.

Many MANET routing protocols have been proposed. Past work focused on designing new protocols, comparing existing protocols, or improving protocols before standard MANET routing protocols are defined. Most research in this field is based on simulation studies of the routing protocols of interest in arbitrary networks with certain traffic

profiles. However, the simulation results from different researchers are not consistent. This is because of the lack of consistency in MANET routing protocol models and application environments including networking and user traffic profiles. Thus, published results and conclusions cannot be generalized and it is difficulty for one to choose a proper routing protocol for his or her MANET application.

The investigation and comparison of MANET routing protocols can be aided by a framework that characterizes MANET routing protocols. However, there is lack of research on this kind of framework. We developed such a framework for MANET routing protocols based on the concept of a relay node set (RNS). The framework provides a new view of MANET routing protocols and highlights relations among properties of the protocols. It facilitates the comparison, design, and improvement of MANET routing protocols. It also identifies factors that affect the operation and, thus, performance of routing protocols and can be used to guide the design of controllable MANET simulation experiments.

MANETs usually have bandwidth and, often, power constraints. Therefore, overhead is an important issue for routing protocols in MANETs. Recent research has focused on simulation studies to investigate and improve routing protocols with respect to control overhead. To develop a systematic method for the design and improvement of routing protocols, we used the RNS framework to draw an analytical model for control overhead of MANET routing protocols. Aided by this analytical model, we designed a new routing protocol using the concept of a connected dominating set (CDS). This new routing protocol has low control overhead and maintains full topology. It illustrates the utility of the RNS framework. A prototype Linux implementation of this protocol has been developed and a simulation model for ns2 is also available.

According to the analytical model drawn from the framework, node mobility is one of the most important characteristics of a MANET. We defined a relative node mobility model to study the relation between the node mobility and link durations in MANETs.

We proposed an adaptive interval scheme based on a neighbor stability criterion. We also introduced a two-state connectivity model for simulation of MANET routing protocols. This model can provide relatively simple control of network environment parameters that affect the performance of routing protocols. The research on the link duration, the adaptive interval scheme, and the new mobility model is in the early stage and can, potentially, lead to promising future research.

We improved a widely-used network simulator ns2, by fixing bugs and incorporating the same assumptions into the models of the four MANET routing protocols that we studied. Simulation results based on fair comparisons suggest that our new protocol has low control overhead, maintains full topology information, and supports shortest path routing, compared to OLSR, AODV, and TBRPF. Our simulation results show that different routing protocols are suitable for different MANET environments. OLSR and AODV are both suitable for networks without power and bandwidth limitations because OLSR may have high control overhead and the control overhead for AODV is sensitive to the traffic load in terms of the number of traffic flows. AODV has small end-to-end delays, so, it is more suitable for real-time applications than other MANET routing protocols. However, the performance of AODV, in terms of control overhead and throughput, is more sensitive to network and application profiles than the other protocols we studied. Thus, AODV is not suggested for applications that require a routing protocol whose performance is robust over wide variety of conditions. OLSR always produces the highest throughput, but with larger control overhead compared to OSPF-MCDS. However, OLSR has longer average hop counts than other protocols. This is because that the delay to propagate the link state information in OLSR is longer than those of others. In other words, some user data packets are sent by OLSR to a next hop that was the correct next hop, but not now, due to network topology changes. TBRPF has the lowest control overhead. However, it does not provide full topology information. For applications, such as the policy-based quality of service application mentioned in Section 7.2, which requires full topology information, OSPF-MCDS and OLSR are better choices. Further-

more, OSPF-MCDS always has the shortest hop count compared to the other three protocols. Therefore, OSPF-MCDS has the best performance in situations where user data transmissions are the dominating factor in network power consumption. OSPF-MCDS also has the smallest end-to-end delay when the number of traffic loads is large and the link density is high. The reason is partially because MCDS nodes that are responsible to propagate control messages are not necessary the gateways for user traffic flows. Therefore, OSPF-MCDS can be a good candidate for real-time MANET applications with large number of traffic flows and high link density. Simulation experiments also show that proactive routing protocol cannot work properly when the link connectivity change rate is close to the frequency of periodic broadcast used by proactive protocols. Reactive routing protocols fail to operate properly when the time for the initiator to receive the route reply is close to the average link up lifetime.

8.2 Contributions

Part of our work, including the RNS framework and the MCDS algorithms, have received considerable interest from the participants in Internet Engineering Task Force (IETF) MANET standardization group. Two of our papers are cited in one of the recent Internet Drafts [79]. Also, our emulation test bed software has been downloaded over 320 times.

We believe that future research based on our work can also be valuable for the MANET research community. The OSPF-MCDS routing protocol guided by the RNS framework indicates that it is possible to provides full topology and maintains shortest path routing for MANETs with low control overhead. The analytical model on control overhead and the mobility studies presented in this dissertation can provide guidance for future MANET research. Improved MANET simulation models and fair comparison assumptions can be used in future MANET simulation experiments. Our comparison study of four MANET routing protocols can provide guidance for one to choose proper

routing protocols for particular MANETs. The next section summarizes promising future research directions.

8.3 Future Directions

As a special type of network, MANETs have received increasing research attention in recent years. There are many active research projects concerned with MANETs. This section focuses on promising future research directions based on our current research.

As pointed out in Section 3.6, there are some parameters defined in the analytical model drawn from the RNS framework that may not be directly measured. This is a limitation of using our RNS framework. One suggested approach is to use simulation or real-time measurements to estimate those values. This suggests a potential research topic on MANET routing in which estimates of parameters, including network and traffic profiles, can be used to adaptively choose different routing protocols or different modules for one protocol. This also suggests a model for an adaptive MANET routing protocol in which the protocol can choose the combination of different modules that is the most efficient for the current environment or in which MANET nodes can choose different protocols according to the environmental parameters.

Since the RNS framework can characterize MANET routing protocols using four modules, it is possible to describe different MANET protocols in a similar four-module finite-state-machine (FSM) model. In other words, we can extend the RNS framework to construct a FSM framework to describe all those protocols. This FSM framework can support formal verification of MANET routing protocols or efficient implementation of these routing protocols.

It may also be interesting to extend the RNS framework to consider protocols for multicast routing in MANETs. Similarly, there are multicast protocols proposed for MANETs. A framework that characterizes these protocols can aid the design, compari-

son, and improvement of these protocols.

Further study of node mobility is also a promising research direction. Such a study might aid in the design of simulation mobility models, improve estimates of link and path lifetimes, and improve the performance of MANET routing protocols.

More extensive simulation and emulation studies can be used to compare different protocols. Analysis and conclusions can guide users when they choose routing protocols for their MANET applications and aid designers in improving protocols.

Cross-layer design [37, 38] is another promising research direction. Since OSPF-MCDS provides full topology information, it is possible to use this information to tune the parameters of lower and/or higher layer protocols so that the overall network performance, especially, seen by the user, can be improved.

Bibliography

[1] T. Lin, S. F. Midkiff, and J. S. Park, "A Framework for Wireless Ad Hoc Routing Protocols," in *Proc. of IEEE Wireless Communications and Networking Conference (WCNC)*, vol. 2, pp. 1162-1167, 2003.

[2] Internet Engineering Task Force (IETF) MANET working group, "Mobile Ad-hoc Networks (manet)," December, 2001. Available at http://www.ietf.org/html.charters/manet-charter.html.

[3] Virginia Tech, "LMDS at Virginia Tech," September 13, 2001. Available at http://www.lmds.vt.edu.

[4] INT Media Group, "MMDS," 2002. Available at http://webopedia.internet.com/TERM/M/MMDS.html.

[5] IEEE LAN MAN Standards Committee, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std. 802.11a-1999*, The Institute of Electrical and Electronics Engineers, New York, New York, Dec. 1999.

[6] IEEE LAN MAN Standards Committee, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std. 802.11-1997*, The Institute of Electrical and Electronics Engineers, New York, New York, 1997.

[7] T. S. Rappaport, *Wireless Communications Principles & Practice*, Upper Saddle River, New Jersey: Prentice-Hall, 1996.

[8] Bluetooth SIG, "The Official Bluetooth Wireless Info Site", 2003. Avaliable at www.bluetooth.com.

[9] Wireless Embedded Systems, Computer Science Department, University of California at Berkeley, "Self-organized wireless sensor network," August, 2001. Available at http://webs.cs.berkeley.edu/800demo.

[10] J. Wells, "A Network Mobility Survey and Comparison with a Mobile IP Multiple Home Address Extension," M.S. thesis, The Bradley Department of Electrical and Computer Engineering, School of Engineering, Virginia Tech, USA, 2003. Available at http://scholar.lib.vt.edu/theses/available/etd-12272003-225555.

[11] C. Perkins, Editor, "IP Routing for Wireless/Mobile Hosts (mobileip)," Internet Engineering Task Force RFC 2002, October, 1996. Available at http://www.ietf.org/rfc/rfc2002.txt.

[12] J. Moy, "OSPF Version 2," Internet Engineering Task Force RFC 2328, April 1998. Available at http://www.ietf.org/rfc/rfc2328.txt.

[13] P. Jacquet and L. Viennot, "Overhead in Mobile Ad-hoc Network Protocols," Research Report-3965, INRIA, France, June 2000. Available at http://ftp.inria.fr/INRIA/publication/publi-ps-pz/RR/RR-3965.pz.gz.

[14] R. V. Boppana and S. P. Konduru. "An adaptive distance vector routing algorithm for mobile, ad hoc networks," in *Proc. of the 2001 IEEE INFOCOM and Joint Conference of the Computer and Communications Societies*, 2001, vol. 3 , pp. 1753-1762.

[15] T. Goff, N. B. Abu-Ghazaleh, D. S. Phatak, and R. Kahvecioglu, "Preemptive routing in ad hoc networks," in *Proc. of the Seventh International Conference on Mobile Computing and Networking*, 2001, pp. 43-52.

[16] N. J. Zhou and A. A. Abouzeid, "Information-theoretic lower bounds on the routing overhead in mobile ad-hoc networks," in *Proc. of IEEE International Symposium on Information Theory*, 2003, pp. 455-455.

[17] J. Yoo, S. Choi, and C. Kim, "Control overhead reduction for neighbour knowledge acquisition in mobile ad hoc networks," in *Electronics Letters*, vol. 39, issue 9, pp. 740-741, 2003.

[18] D. Kim, C. K. Toh J.-C. Cano, and P. Manzoni, "A bounding algorithm for the broadcast storm problem in mobile ad hoc networks," in *Proc. of Wireless Communications and Networking Conference (WCNC)*, vol. 2, pp. 1131-1136, 2003.

[19] C. E. Jones, K. M. Sivalingam, P. Agrawal, and J. C. Chen, "A survey of energy efficient network protocols for wireless networks," in *Wireless Networks*, vol. 7, no. 4, pp. 343-358, 2001.

[20] J. H. Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," in *Proc. of 19th INFOCOM*, pp. 22-31, 2000.

[21] A. Helmy, "Architectural framework for large-scale multicast in mobile ad hoc networks," in *Proc. of IEEE International Conference on Communications (ICC) 2002*, vol. 4, pp. 2036-2042, 2002.

[22] D. D. Perkins, H. D. Hughes, and C. B. Owen, "Factors affecting the performance of ad hoc networks," in *Proc. of IEEE International Conference on Communicatioins (ICC) 2002*, vol. 4, pp. 2048-2052, 2002.

[23] T. W. Chen, "Efficient Routing and Quality of Service Support for Ad Hoc Wireless Networks." Ph.D. dissertation, University of California, Los Angeles, Department of Computer Science, March 1998. Available at http://citeseer.nj.nec.com/52768.html.

[24] J. C. Liu, Q. Zhang, W. W. Zhu, J. Zhang, and B. Li, "A novel framework for QoS-aware resource discovery in mobile ad hoc networks," in *Proc. of IEEE International Conference on Communications (ICC) 2002*, vol. 2, pp. 1011-1016, 2002.

[25] O. Hussein and T. Saadawi, "Ant routing algorithm for mobile ad-hoc networks (ARAMA)", in *Proc. of International Conference on Performance, Computing, and Communications Conference (IPCCC)*, pp. 281-290, 2003.

[26] C. H. Yeh and T. T. You, "A QoS MAC protocol for differentiated service in mobile ad hoc networks," in *Proc. of International Conference on Parallel Processing*, pp. 349-356, 2003.

[27] A. Zhou and H. Hassanein, "Load-balanced wireless ad hoc routing,"in *Proc. of Canadian Conference on Electrical and Computer Engineering,* vol. 2 , 2001, pp. 1157-1161.

[28] J. H. Song, V. Wong, and V. C. M. Leung, "Load-aware on-demand routing (laor) protocol for mobile ad hoc networks," in *Proc. of Vehicular Technology Conference (VTC) Spring 2003*, vol. 3, pp. 1753-1757, 2003.

[29] I. D. Aron and S. K. S. Gupta, "On the scalability of on-demand routing protocols for mobile ad hoc networks: an analytical study," in *Journal of Interconnection Networks*, vol. 2, no. 1, pp. 5-29, 2001.

[30] C. A. Santivanez, B. McDonald, I. Stavrakakis, and R. Ramanathan, "On the scalability of ad hoc routing protocols," in *Proc. of INFOCOM 2002*, vol. 3, pp. 1688-1697, 2002.

[31] J. Lundberg, "Routing Security in Ad Hoc Networks," Helsinki University of Technology, 2000. Available at http://citeseer.nj.nec.com/400961.html.

[32] S. Capkun, J. P. Hubaux, and L. Buttyan, "Mobility helps security in ad hoc networks," in *Proc. of the 4th ACM international symposium on Mobile ad hoc networking and computing*, pp. 46-56, 2003.

[33] Y. G. Zhang, W. K. Lee, and Y. A. Huang, "Intrusion detection techniques for mobile wireless networks," in *Wireless Networks*, vol. 9, issue 5, pp. 545-556, 2003.

[34] L. Qin, "Pro-Active Route Maintenance In DSR," M.S. thesis, Ottawa-Carleton Institute of Computer Science, School of Computer Science, Carleton University, Canada, 2001. Available at http://citeseer.nj.nec.com/qin01proactive.html.

[35] A. Boukerche and S. Rogers, "GPS query optimization in mobile and wireless networks," in *Proc. of the 6th IEEE Computers and Communications Conference*, pp. 198-203, 2001.

[36] INFOSAT Telecommunications, "Iridium System Specifications," INFOSAT telecommunications, January 18, 2002. Available at http://iridium.infosat.com/iridium_specifications.htm.

[37] A. Safwati, H. Hassanein, and H. Mouftah, "Optimal cross-layer designs for energy-efficient wireless ad hoc and sensor networks," in *Proc. of International Conference on Performance, Computing, and Communications Conference (IPCCC)*, pp. 123-128, 2003.

[38] W. H. Yuen, H. Lee, and T. D. Andersen, "A simple and effective cross layer networking system for mobile ad hoc networks," in *Proc. of the 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 4, pp. 1952-1956, 2002.

[39] Z. Q. Ye, S. V. Krishnamurthy, and S. K. Tripathi, "A framework for reliable routing in mobile ad hoc networks," in *Proc. of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 1, pp. 270-280, 2003.

[40] A. M. Abbas and B. N. Jain, "An analytical framework for path reliabilities in mobile ad hoc networks," in *Proc. of the 8th IEEE International Symposium on Computers and Communication (ISCC)*, pp. 63-68, 2003.

[41] D. B. West, *Introduction to Graph Theory, 2nd ed.*. Upper Saddle River, NJ: Prentice Hall. 2001.

[42] A. Boukerche, "Performance comparison and analysis of ad hoc routing algorithms," in *Proc. of IEEE International Conference on Performance, Computing, and Communications*, pp. 171-178, 2001.

[43] J. Moy, "Link-State Rouing", *Routing in Communications Networks*, Martha Steenstrup, editor, Upper Saddle River, New Jersey: Prentice-Hall, 1995, pp. 135-157.

[44] G. S. Malkin and M. E. Steenstrup, "Distance-Vector Routing," *Routing in Communications Networks*, Martha Steenstrup, editor, Upper Saddle River, New Jersey: Prentice-Hall, 1995, pp. 83-98.

[45] B. Baccala, editor, "Link State Routing Protocols," *Connected: An Internet Encyclopedia*, April, 1997. Available at http://www.freesoft.org/CIE/index.htm.

[46] J. J. Garcia-Luna-Aceves and M. Spohn, "Efficient routing in packet-radio networks using link-state information," in *IEEE Wireless Communications and Networking Conference (WCNC)*, vol. 3, pp. 1308-1312, 1999.

[47] R. G. Ogier, "Efficient Routing Protocols for Packet-Radio Networks Based on Tree Sharing." in *Proc. of the 6th IEEE Intl. Workshop on Mobile Multimedia Communications*, pp. 104-113, 1999.

[48] R. G. Ogier, F. L. Templin, B. Bellur, and M. G. Lewis, "Topology Broadcast Based on Reverse-Path Forwarding (TBRPF)," Internet Engineering Task Force (IETF) draft, November 2002. Available at http://www.ietf.org/internet-drafts/draft-ietf-manet-tbrpf-06.txt.

[49] D. B. Johnson, D. A. Maltz, Y. C. Hu, and J. G. Jetcheva, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)," Internet Engineering Task Force

(IETF) draft, Febuary 2002. Available at http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-07.txt.

[50] C. E. Perkins, E. M. Belding-Royer, and S. R. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," Internet Engineering Task Force (IETF) draft, November 2002. Available at http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-12.txt.

[51] V. Park and S. Corson, "Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification," Internet Engineering Task Force (IETF) draft, July 2001.

[52] E. Gafni and D. Bertsekas, "Distributed algorithms for generating loop-free routes in networks with frequently changing topology," in *IEEE Trans. on Communication*, vol. 29, no. 1, pp. 11-18, January 1981.

[53] T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L. Viennot, "Optimized Link State Routing Protocol," Internet Engineering Task Force (IETF) draft, March, 2002. Available at http://www.ietf.org/internet-drafts/draft-ietf-manet-olsr-06.txt.

[54] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint relaying: An efficient technique for flooding in mobile wireless networks," Research Report-3898, INRIA, France, 2000.

[55] M. Gerla, X. Y. Hong, L. Ma, and G. Y. Pei, "Landmark Routing Protocol (LANMAR) for Large Scale Ad Hoc Networks," Internet Engineering Task Force (IETF) draft, November 2002. Available at http://www.ietf.org/internet-drafts/draft-ietf-manet-lanmar-05.txt.

[56] M. Gerla, X. Y. Hong, and G. Y. Pei, "Fisheye State Routing Protocol (FSR) for Ad Hoc Networks," Internet Engineering Task Force (IETF) draft, June 2001. Available at http://www.ietf.org/internet-drafts/draft-ietf-manet-fsr-03.txt.

[57] Z. J. Haas, M. R. Pearlman, and P. Samar, "The Intrazone Routing Protocol (IARP) for Ad Hoc Networks," Internet Engineering Task Force (IETF) draft, July 2002. Available at http://www.ietf.org/internet-drafts/draft-ietf-manet-zone-iarp-02.txt.

[58] Z. J. Haas, M. R. Pearlman, and P. Samar, "The Interzone Routing Protocol (IERP) for Ad Hoc Networks," Internet Engineering Task Force (IETF) draft, July 2002. Available at http://www.ietf.org/internet-drafts/draft-ietf-manet-zone-ierp-02.txt.

[59] Meeting Report, "Mobile Ad-hoc Networks (manet) Charter," Internet Engineering Task Force (IETF), March 18, 2002. Available at http://www.ietf.org/proceedings/02mar/179.htm.

[60] C. Ying, Q. Lv, Y. Liu, and M. Shi, "Routing protocols overview and design issues for self-organized network," in *Proc. of International Conference on Communication Technology*, vol. 2, pp. 1298-1303, 2000.

[61] E. M. Royer and C. K. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," in *IEEE Personal Communications*, vol. 6, no. 2, pp. 46-55, 1999.

[62] S. R. Das, C. E. Perkins, and E. M. Royer, "Performance comparison of two on-demand routing protocols for ad hoc networks," in *Proc. 19th INFOCOM*, vol. 1, pp. 3-12, 2000.

[63] C. E. Perkins, E. M. Royer, S. R. Das, and M. K. Marina, "Performance comparison of two on-demand routing protocols for ad hoc networks," in *IEEE Personal Communications*, vol. 8, no. 1, pp. 16-28, 2001.

[64] A. Boukerche, "A simulation based study of on-demand routing protocols for ad hoc wireless networks," in *Proc. 34th Annual Simulation Symposium*, pp. 85-92, 2001.

[65] T. Ogawa, E. Kudoh, and H. Suda, "Multi-routing schemes for ad-hoc wireless networks," in *Proc. IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 866-869, 2001.

[66] J. Raju, and J. J. Garcia-Luna-Aceves, "A comparison of on-demand and table driven routing for ad-hoc wireless networks," in *IEEE International Conference on Communications*, vol. 3, pp. 1702-1706, 2000.

[67] M. E. Steenstrup, "Routing under uncertainty: a comparative study," in *Proc. IEEE Wireless Communications and Networking Conference*, vol. 1, pp. 112-116, 2000.

[68] I. D. Aron and S. K. S. Gupta, "On the scalability of on-demand routing protocols for mobile ad hoc Networks: an analytical study," in *Journal of Interconnection Networks*, vol. 2, no. 2, pp. 5-29, 2001.

[69] V. D. Park and M. S. Corson, "A performance comparison of the temporally-ordered routing algorithm and ideal link-state routing," in *Proc. 3rd IEEE Symposium on Computers and Communications*, pp. 592-598, 1998.

[70] V. Ramasubramanian, Z. J. Haas, and E. G. Sirer, "SHARP: A Hybrid Adaptive Routing Protocol for Mobile Ad Hoc Networks," in *Proc. The Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pp. 303-314, 2003

[71] J. J. Garcia-Luna-Aceves and M. Spohn, "Scalable link-state internet routing," in *Proc. IEEE International Conference on Network Protocols*, pp. 52-61, 1998.

[72] T. Lin, S. F. Midkiff, and J. S. Park, "Approximation algorithms for minimal connected dominating sets and application with routing protocol in wireless ad hoc network," in *Proc. of IEEE International Performance Computing and Communications Conference (IPCCC)*, pp. 157-164, 2003.

[73] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets," in *Proc. of European Symposium on Algorithms*, pp. 179-193, 1996.

[74] B. Das and V. Bhargavan, "Routing in ad-hoc networks using minimum connected dominating sets," in *Proc. of IEEE International Conference on Communications*, vol. 1, pp. 376-380, 1997.

[75] J. Wu and H. Li, "A dominating-set-based routing scheme in ad hoc wireless networks," in *Wireless Networks in the Telecommunication Systems Journal*, vol. 3, pp. 63-84, 2001.

[76] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks," in *IEEE Trans. on Parallel and Distributed Systems*, vol. 13, No. 1, pp. 14-25, 2002.

[77] U. C. Kozat, G. Kondylis, B. Ryu, and M. K. Marina, "Virtual dynamic backbone for mobile ad hoc networks," in *Proc. of IEEE International Conference on Communications (ICC)*, vol. 1, pp. 250-255, 2001.

[78] Robert Sedgewick, *Algorithms in C++*. Addison-Wesley Publishing Company, Reading, MA., pp. 427, 1992.

[79] R. Ogier, SRI. Internet Draft. October 2003. "Alternative Designs for OSPF Extensions for Mobile Ad Hoc Networks." Available at http://www.ietf.org/internet-drafts/draft-ogier-manet-ospf-extension-00.txt.

[80] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," in *Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, no. 5, pp. 483-502, 2002.

[81] T. Lin and S. F. Midkiff, "Mobility versus link stability in simulation of mobile ad hoc networks," in *Proc. of Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, pp. 3-8, 2003.

[82] T. Lin, S. F. Midkiff, J. S. Park, and Y. Lin, "An analytical study of connectivity changes in mobile ad-hoc network simulations," in *Proc. of Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, 2004.

[83] M. Liberatore, January 2002. "Background (Mobility Models and Data for Wireless Networks)" Available at http://signl.cs.umass.edu/ liberato/wireless_background.html.

[84] G. Pei, M. Gerla, and T. W. Chen, "Fisheye state routing: a routing scheme for ad hoc wireless networks," in *Proc. of IEEE International Conference on Communications*, pp. 70-74, 2000.

[85] G. Pei, M. Gerla and X. Hong, "LANMAR: landmark routing for large scale wireless ad hoc networks with group mobility," in *Proc. of IEEE/ACM MobiHOC*, pp. 11-18, 2000.

[86] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *Proc. of ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 85-97, 1998.

[87] C. Chiang and M. Gerla, "On-demand multicast in mobile wireless networks," in *Proc. of IEEE International Conference on Network Protocols*, pp. 262-270, 1998.

[88] J. J. Garcia-Luna-Aceves and M. Spohn, "Source-tree routing in wireless networks," in *Proc. of IEEE International Conference on Network Protocols*, pp. 273-282, 1999.

[89] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. "Scenario-based performance analysis of routing protocols for mobile ad-hoc networks," in *Proc. of ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 195-206, 1999.

[90] D. Johnson and D. Maltz, "Dynamic source routing in ad hoc wireless networks," in T. Imelinsky and H. Korth, editors, *Mobile Computing*, pp.153-181. Kluwer Academic Publishers, 1996.

[91] E. M. Royer and C. E. Perkins, "Multicast operation of the ad-hoc on-demand distance vector routing protocol," in *Proc. of ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 207-218, 1999.

[92] C. Bettstetter, "Mobility modeling in wireless networks: categorization, smooth movement, and border effects," in *ACM Mobile Computing and Communications Review*, vol. 5, no. 3, pp. 55-67, 2001.

[93] C. Bettstetter and C. Wagner, "The spatial node distribution of the random waypoint mobility model," in *Proc. of the 1st German Workshop on Mobile Ad-Hoc Networks*, Ulm, Germany, GI Lecture Notes in Informatics, no. P-11, pp. 41-58, 2002.

[94] C. Bettstetter, H. Hartenstein, and X. Pérez-Costa, "Stochastic properties of the random waypoint mobility model: epoch length, direction distribution, and cell change rate," in *Proc. of ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems*, pp. 7-14, 2002.

[95] C. de Waal and M. Gerharz, "BonnMotion mobility scenario generator and analysis tool," 2003. Available at http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/BonnMotion.

[96] J. K. Yoon, M. Y. Liu, and B. Noble, "Random waypoint considered harmful," in *Proc. of IEEE INFOCOM*, vol. 2, pp. 1312-1321, 2003.

[97] C. de Waal and T. Lin, Personal Communication, January 4 - Febuary 10, 2003.

[98] K. Fall and K. Varadhan, editor, "The ns Manual (formerly ns Notes and Documentation)," Project report by UC Berkeley, LBL, USC/ISI, and Xerox PARC, January 2002. Available at http://www.isi.edu/nsnam/ns/doc/index.html.

[99] Scalable Network Technologies Inc., 2004. "Scalable Network Technologies - Products - QualNet" Available at http://www.qualnet.com/products/QualNet/index.html.

[100] OPNET Technologies Inc., 2004. "OPENT Technologies Inc." Available at http://www.opent.com.

[101] X. Hong, M. Gerla, G. Pei, and C. C. Chiang, "A Group Mobility Model for Ad Hoc Wireless Networks," in *Proc. of ACM/IEEE MSWiM*, pp. 53-60, 1999.

[102] G. Pei, M. Gerla, X. Hong, and C. Chiang, "A Wireless Hierarchical Routing Protocol with Group Mobility," in *Proc. of the IEEE Wireless Communication and Networking Conference (WCNC)*, vol. 3, pp. 1538-1542, 1999.

[103] S. R. Das, C. E. Perkins, E. M. Royer, and M. K. Marina, "Performance comparison of two on-demand routing protocols for ad hoc networks," in *IEEE Personal Communications Magazine special issue on Ad hoc Networking*, pp. 16-28, 2001.

[104] F. Bai, N. Sadagopan and A. Helmy, "IMPORTANT: a framework to systematically analysis the impact of mobility on performance of routing protocols for adhoc network," in *Proc. of IEEE INFOCOM*, vol. 2, pp. 825-835, 2003.

[105] K. H. Wang and B. C. Li, "Group mobility and partition prediction in wireless adhoc networks," in *Proc. of IEEE International Conference on Communications (ICC)*, vol. 2, pp. 1017-1021, 2002.

[106] N. Sadagopan, F. Bai, B. Krishnamachari, and A. Helmy, "PATHS: analysis of PATH duration statistics and their impact on reactive MANET routing protocols," in *Proc. of the 4th ACM international symposium on Mobile ad hoc networking and computing*, pp. 245-256, June, 2003.

[107] E. M. Belding-Royer et al., Mobility Management and Networking (MOMENT) Laboratory, UCSB. AODV implementations. Available at http://moment.cs.ucsb.edu/AODV.

[108] T. Clausen and P. Jacquet, HIPERCOM : High Performance Communication, INRIA, France. OLSR ns2 implementations. Available at http://menetou.inria.fr/olsr.

[109] M. Minematsu, M. Saito, H. Aida, Y. Tobe, and H. Tokuda, "HoWL: An efficient route discovery scheme using routing history in ad hoc networks," in *Proc. of Local Computer Networks (LCN)*, pp. 20-29, 2002.

[110] R. Jain, *The Art of Computer Systems Performance Analysis*, John Wiley & Sons, pp. 441 455, 1991.

[111] T. Lin, S. F. Midkiff, and J. S. Park, "A Dynamic Topology Switch for the Emulation of Wireless Ad Hoc Networks Using a Wired Network," in *Proc. of IEEE Wireless Local Network Workshop*, pp. 791-798, 2002.

[112] K. Fall and K. Varadhan, editor, "Visualization with Nam - The Network Animator" Project report by UC Berkeley, LBL, USC/ISI, and Xerox PARC, January 2002. Available at http://www.isi.edu/nsnam/ns/doc/node491.html.

[113] T. D. Dyer and R. V. Boppana, "Analysis of TCP and UDP traffic in MANETs," in *Proc. of the 3rd IEEE workshop on Wireless LANs* 2001.

[114] IEEE LAN MAN Standards Committee, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std. 802.11g*, The Institute of Electrical and Electronics Engineers, New York, New York, June 2002. Available at http://grouper.ieee.org/groups/802/11/Reports/tgg_update.htm.

[115] E. N. Gilbert, "Capacity of a burst-noise channel," in *Bell Systems Technical Journal*, Vol. 39, pp. 1253-1265, September 1960.

[116] L. L. Peterson and B. S. Davie, *Computer Networks*, 2nd edition, Morgan Kaufmann Publishers, San Francisco, CA, 2000.

[117] G. T. Nguyen, R. H. Katz, B. D. Noble, and M. Satyanarayanan, "A trace-based approach for modeling wireless channel behavior," in *Proc. Winter Simulation Conf.*, pp. 597-604, 1996.

[118] B. D. Noble, M. Satyanarayanan, G. T. Nguyen, and R. H. Katz, "Trace-based mobile network emulation," in emphProc. ACM SIGCOMM, pp. 51-61, 1997.

[119] D. B. Johnson, "Validation of wireless and mobile network models and simulation," in *Proc. DARPA/NIST Network Simulation Validation Workshop*, 1999. Available at http://www.monarch.cs.cmu.edu/monarch-papers/darpa99.ps.

[120] K. Fall, "Network Emulation in the VINT/NS Simulator," in *Proc. IEEE International Symp. on Computers and Communications*, pp. 244-250, 1999.

[121] K. Xu, M. Takai, J. Martin, and R. Bagrodia, "Looking ahead of real time in hybrid component networks," in *Proc. 15th Workshop on Parallel and Distributed Simulation*, pp. 14-21, 2001.

[122] W. Chao, J. P. Macker, and J. W. Weston, "A flexible, low-cost tabletop mobile network emulator and laboratory testing toolkit," Project Report by Protocol Engineering and Advanced Networking (Protean), Research Group, Communication Systems Branch, Information Technology Division, the Office of Naval Research, USA, November 2002. Available at http://downloads.pf.itd.nrl.navy.mil/proteantools/Emulator_Report_v3.doc.

[123] M. Fjällström and J. Nielsen, "Ad Hoc Protocol Evaluation (APE) test bed," developed by Department of Computer Science, Uppsala University, Sweden, January 2002. Availabe at apetestbed.sourceforge.net.

[124] J. T. Moy, *OSPF Complete Implementation*, Addison-Wesley, 2001.

[125] J. T. Moy, "OSPFD Routing Software Resources." Available at http://www.ospf.org.

[126] MandrakSoft Inc., "Mandrake Linux Home," 2002. Available at http://www.linux-mandrake.com/en.

[127] INRIA, "Optimized Link State Routing," December 2001. Available at http://menetou.inria.fr/olsr.

[128] ZNYX Networks, Inc., "ZNYX ZX370 PCI 4-channel 10/100 Mbps card," 2004. Available at http://www.znyx.com/products/hardware/zx370.htm.

[129] L. A. DaSilva, S. F. Midkiff, J. S. Park, G. Hadjichristofi, K. Phanse, T. Lin, and N. J. Davis, "Network Mobility and Protocol Interoperability in Ad Hoc Networks," submitted to *IEEE Communications Magazine*.

[130] The Open Group Base Specifications Issue 6 IEEE Std 1003.1, 2003 Ed. "The Base Specifications Issue 6," 2003. Available at http://www.opengroup.org/onlinepubs/007904975/functions/xsh_chap02_09.html.

[131] Libpcap, 2004 "The libpcap project," 2004. Available at http://sourceforge.net/projects/libpcap.

[132] P. Wang, "Libnet," February 2003. Available at http://libnet.sourceforge.net.

[133] T. Al-Herbish and J. W. Temples, "Raw IP Networking FAQ," November 1999. Available at http://www.whitefang.com/rin/rawfaq.html#20.

[134] M. Michal, "FreeBSD and pthreads," February 2004. Avaliable at http://www.mail-archive.com/ntop-dev@unipi.it/msg03167.html.

[135] P. O'Hanlon and K. Hasler, Network Research Group at the Lawrence Berkeley National Laboratory in collaboration with the University of California, Berkeley. "Video Conferencing Tool." Available at http://www-mice.cs.ucl.ac.uk/multimedia/software/vic.

Appendix: Specification of OSPF-MCDS

Tao Lin
Scott F. Midkiff
Jahng S. Park
Bradley Department of Electrical
and computer engineering
Virginia Tech
USA
19 March 2004

Open Shortest Path First with Minimal
Connected Dominating Set (OSPF-MCDS)

draft-ospfmcds-00.txt

Status of this Memo

This document is a submitted by the Laboratory for Advanced networking (LAN) in the Bradley Department of Electrical and computer engineering at Virginia Polytechnic Institute and State University. Comments can be sent to midkiff@vt.edu or taolin@vt.edu.

Abstract

This document describes the Open Shortest Path First with Minimal Connected Dominating Set (OSPF-MCDS) protocol for

routings in mobile ad hoc networks (MANETs). The protocol is an optimization of the pure link state algorithm. The key concept used in this protocol is that of minimal connected dominating set (MCDS)[1]. We call this protocol OSPF-MCDS.

Link state records are broadcast in the MCDS. This technique substantially reduces the number of retransmissions in a MANET as compared to a pure flooding mechanism, where every node retransmits each first-seen control message from neighbors. The algorithm to select MCDS in this protocol is an approximation algorithm that could form a minimal MCDS with low control overhead. The number of broadcast relay nodes is small and, thus, the overall control overhead is low according to the framework proposed in [2]. In OSPF-MCDS, information broadcast "through" the selected MCDS nodes is link state information. MCDS nodes act as the designated routers in OSPF. Thus, this protocol can maintain full topology information and provide optimum routing as OSPF in MANETs.

1. Introduction

The Open Shortest Path First with Minimal Connected Dominating Set (OSPF-MCDS) is developed for mobile ad hoc networks (MANETs). It is a table driven and proactive protocol, and exchanges topology information with other nodes regularly. The nodes that are selected as relay

nodes form a minimal connected dominating set (MCDS). Only nodes in the MCDS will re-broadcast the first-seen control message broadcast by neighbors. Thus, the number of rebroadcasts is reduced. There are schemes to handle the topology changes so that CDS will not always be re-built with extra control overhead. Link state information will be propagated to all nodes via MCDS so that all nodes keep identical copies of network topology. In route calculation, Dijkstra algorithm can be used to build shortest paths to all destinations.

MCDS are selected based on approximation algorithms presented [1]. Nodes in MCDS are connected and cover all other nodes. Therefore, all nodes are guaranteed to receive the link state information broadcast in the network. This protocol uses a MCDS as the relay node set to replace the designated router concept used in OSPF [4].

OSPF-MCDS is developed to work independently from other protocols. It inherits the concepts from OSPF (STANDARD RFC 2328 [4]) but using different message formats. The protocol is developed in the NAVCIITI project supported by the Office of Naval Research (ONR), USA.

1.1. Changes

N/A

1.2. Draft Organization

TBD.

1.3. OSPF-MCDS Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119 [5]. The OSPF-MCDS protocol uses the following terminology:

close neighbor set of node i

A set of neighbor nodes of node i, in which any node's neighbor is either node i or still a member of this node set. Note that not all nodes can have close neighbor sets. If a node does have at least one such neighbor set, it is possible to generate this close neighbor set by starting from any node in that close neighbor set and adding its neighbor into the node set.

common end node

A node that connects two nodes is called a common end node for these two nodes. There could be one

interface or two interfaces at this node to connect to those two nodes.

common neighbor

A common neighbor for a pair of neighbor nodes is a node that connects to both of these two neighbor nodes.

connected nodes

Nodes that could form a connected graph are called connected nodes.

connected dominating set (CDS)

A connected dominating set is formed by a set of connected nodes, which covers all nodes in the network. Nodes in the CDS re-broadcast first-seen broadcast messages, while other nodes keep silent.

cost of a link

There is a metric value assigned to each interface. It could be the power consumption, traffic load, or any other values associate with this interface or this node. The cost of a link equals to the sum of the metrics of the two interfaces that form this

link.

cost of a path

The cost of a path is the sum of cost of all links along this path.

cover

A node is covered by all its neighbors. A node covers all its neighbors. A set of nodes M covers a set of nodes N when any node in N is covered by at least a node in M.

declare

A node may broadcast a control message to its neighbors. This action is called declare.

end MCDS node

An end point node of a link which is in the MCDS.

exit node

Exit nodes are nodes in an open neighbor set of node i. They have neighbors that are not adjacent to node i.

interface

A network device participating in the routing. For example, wireless network card. A node may have more than one interface. Each interface should have its unique IP address.

larger MCDS neighbor

MCDS neighbors that with larger router ids.

link

A link is a connection between a pair of interfaces from two different nodes. A node is said to have a link to another node when they can communicate with each other via that pair of interfaces.

MCDS neighbor

Neighbors that are in the MCDS.

MCDS node

A node that is in the MCDS.

minimal connected dominating set (MCDS)

A connected dominating set of minimal size is called minimal connected dominating set (MCDS).

neighbor interface

It is the interface that used by a node to communicate with a neighbor via a link to that neighbor.

neighbor node

A node X is a neighbor node of node Y if there is a link between X and Y.

neighbors of a link

Neighbors of a link are the pair of nodes that form this link.

node

A router that runs the OSPF-MCDS routing protocol.

non-adjacent neighbors of node i

A set of neighbors of node i in which any two nodes are disconnected

non-MCDS neighbor

Neighbors that are not in the MCDS.

non-MCDS node

A node that is not in the MCDS.

open neighbor set of node i

An open neighbor set of node i is a set of neighbor nodes of i that does not include any close neighbor set.

path

A sequence of links that connects the sender interfaces and the receiver interfaces. Adjacent links must have a common end node.

receiver interface

The receiver interface of a control message is the interface that receives that control message.

router id

Generally, the router id for a node is the IP address of its interface. When a node has multiple interfaces, there is a copy of OSPF-MCDS running on each interfaces. The route id for a node in a copy of OSPF-MCDS is the IP address of the corresponding interface that runs this copy of OSPF-MCDS.

sender interface

The sender interface of a control message is the interface that transmits the message.

sub tree of node i

It is a tree formed by all edges terminating at node i.

symmetric link

The pair of interfaces that associate with a link can both receive messages from the other, then this link is called symmetric link.

root of a sub tree

It is the common node of all edges in the sub tree.

1.4. Applicability Section

OSPF-MCDS is a proactive routing protocol for mobile ad hoc networks. It provides good performance, in terms of control overhead and optimum routings, for large and dense mobile networks with heavy traffic load in terms of number of traffic flows because of the idea of broadcast using MCDS. The larger and more dense a network, the more optimization can be achieved as compared to the classic link state algorithm with blind broadcasts. [1]

OSPF-MCDS is well suited for networks where the traffic load is heavy, in terms of number of traffic flows, and the network is large, compared to a reactive protocol[2]. OSPF-MCDS uses the minimal size of relay nodes to broadcast and small computation time to maintain the relay node set. Therefore, it has low control overhead according to the overhead model presented in [2, 3]. Since it is maintains full topology, it can provide optimum routings for MANETs. Note that the term of optimum routing could be smallest-hop-routing, minimum-power-consumption-routing, or balance-load-routing, depending on the definition of cost for each link.

1.5. Protocol Overview

OSPF-MCDS is a distributed proactive routing protocol for MANETs. The protocol inherits many good features for pro-active protocols, such as optimum routings. OSPF-MCDS is

optimized over the pure link state protocol for MANET routing.

OSPF-MCDS uses hello protocol to detect link up or down events. In stead of blind broadcasts, OSPF-MCDS reduces the number of rebroadcasts using the minimal connected dominating set (MCDS). MCDS selection is done by an approximation algorithm. It relies on the local copy of the topology in each node. Broadcast in MCDS can reduce control overhead, compared to blind broadcasts used by some other protocols. Link state information will be propagated to all nodes via MCDS. Thus, each node can build its own routing table for all destinations by Dijkstra algorithm.

A MCDS is maintained locally when the topology changes. OSPF-MCDS benefits from traffic patterns where a large group of nodes are communicating with another large group of nodes, and where the pairs of source and destination nodes are changing frequently. OSPF-MCDS is scalable for large and dense networks [2]. OSPF-MCDS can also work properly even if the underlying MAC layer protocol does not guarantee delivery of control messages between nodes. Proof is shown in [3].

1.6. Connected Dominating Set

The idea of connected dominating set is to reduce the

number of rebroadcast in the network and, thus, to reduce the control overhead. A node rebroadcasts a first-seen broadcast message only if it is a member of the MCDS. The selection of MCDS is a distributed procedure. Nodes decide whether to join the MCDS by themsevles. The MCDS set is connected and cover all nodes in the network. Therefore, all nodes can receive those broadcasting messages. The protocol is optimized when the size of MCDS is the minimum. An approximation algorithm concluded from observations is used by OSPF-MCDS [1]. The details are presented in section 2.2.1.

2. Protocol Functioning

This section presents the packet formats and procedures to process and broadcast messages. 2.1 describes the format of control messages used in OSPF-MCDS. Packet processing and broadcast are presented in 2.2. A technique to reduce collision of control messages is introduced in 2.4.

2.1. Packet Formats

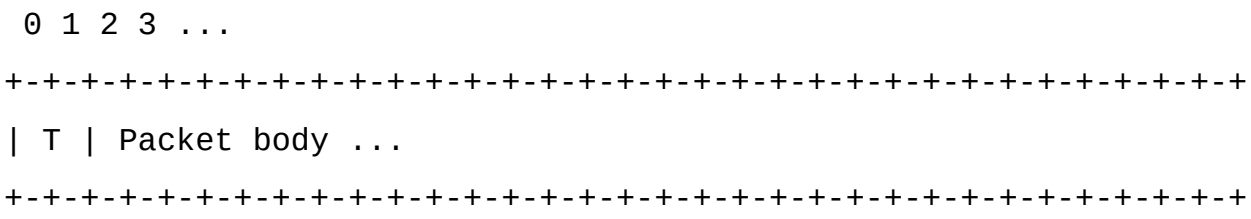
OSPF-MCDS has three sub protocols, hello protocol, GCDS protocol, and link state synchronize protocol. All messages are using IP data packet with one unique protocol types. We use one un-assigned numbers, 168, in our draft and implementations.

Four packet types are defined in OSPF-MCDS, HELLO packet (H), Link Database Description (LDD), Link State Description (LSD) and Interface and/or Prefix Declaration (IPD). The following sections present details.

In this version, we assume that these control messages are sent in SEND_TIME_INTERVAL seconds. Thus, control packets in one buffer can share the same IP and MAC headers.

2.1.1. General Header

The basic layout of any packet formats in OSPF-MCDS is as shown follows (omitting the IP headers):



T: packet type

T	Type
0x00	H
0x01	LDD

```
        +-------+------+
        |  0x10 |  LSD |
        +-------+------+
        |  0x11 |  IPD |
        +-------+------+
```

2.1.2. HELLO Message (H)

A HELLO message includes the list of newly detected neighbors. The general packet format is shown as follows.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 0|     N     |  Neighbor ID List ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

N: No. of New Neighbors. The maximum value is 63. If the number of new neighbor is larger than the maximum value, another HELLO message will be used.

The neighbor List contains IDs of those new neighboring nodes:

```
 0   1                   2                   3
 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Neighbor ID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
:                                                               :
:                                                               :
```

2.1.3. Link Database Description (LDD)

Link database description messages are used to exchange link state information between nodes.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 1|    PAD    |# of Link Lists|      Link Lists ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

PAD: PAD is reserved. It can be used for checksums in the future.

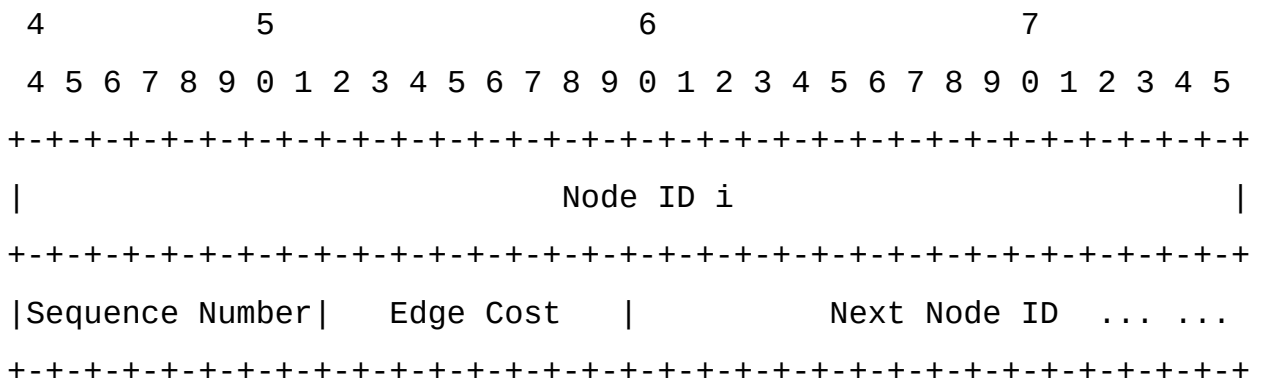
# of Link Lists: The number of link list in this message.

The format of one Link List is shown as follows:

```
 1       2                   3                   4
 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  No. of edges |                       Sub Tree Root ID
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                |                          Edge List ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

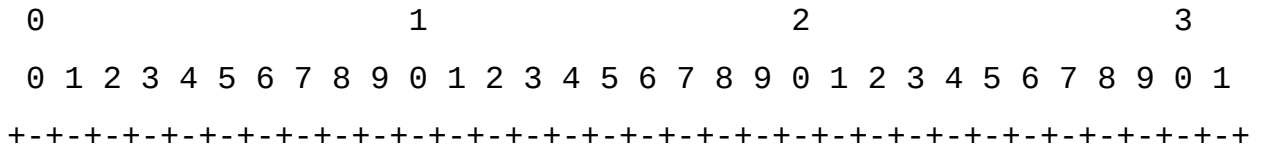
No. of edges: the size of the Edge List in this Link List. Edges in one Link List share one common end node.
Sub Tree Root ID: The router ID of the root node, which is the common end node of those edges in this list.

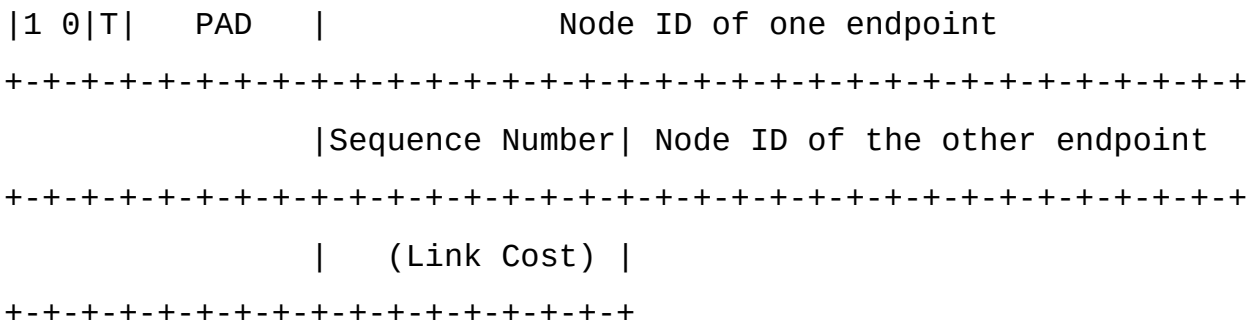
Note that the maximum value of the number of edges is 255. If it is not enough for a link list, another link list will be used. An Edge List is formed by edge entries, which consists node id of the other end point of one edge, sequence number of this edge and the cost for this edge. The details are shown below:



2.1.4. Link State Description (LSD)

Link state descriptions are used to propagate topology changes in the network right after any topology changes.





T: the sub type for the link state.

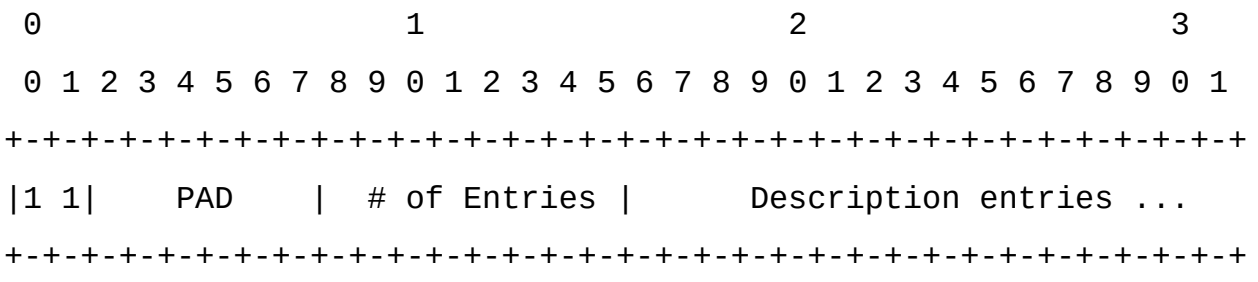
T	Type
0x0	Link UP
0x1	Link DOWN

PAD: PAD is reserved. It can be used for checksums in the future.

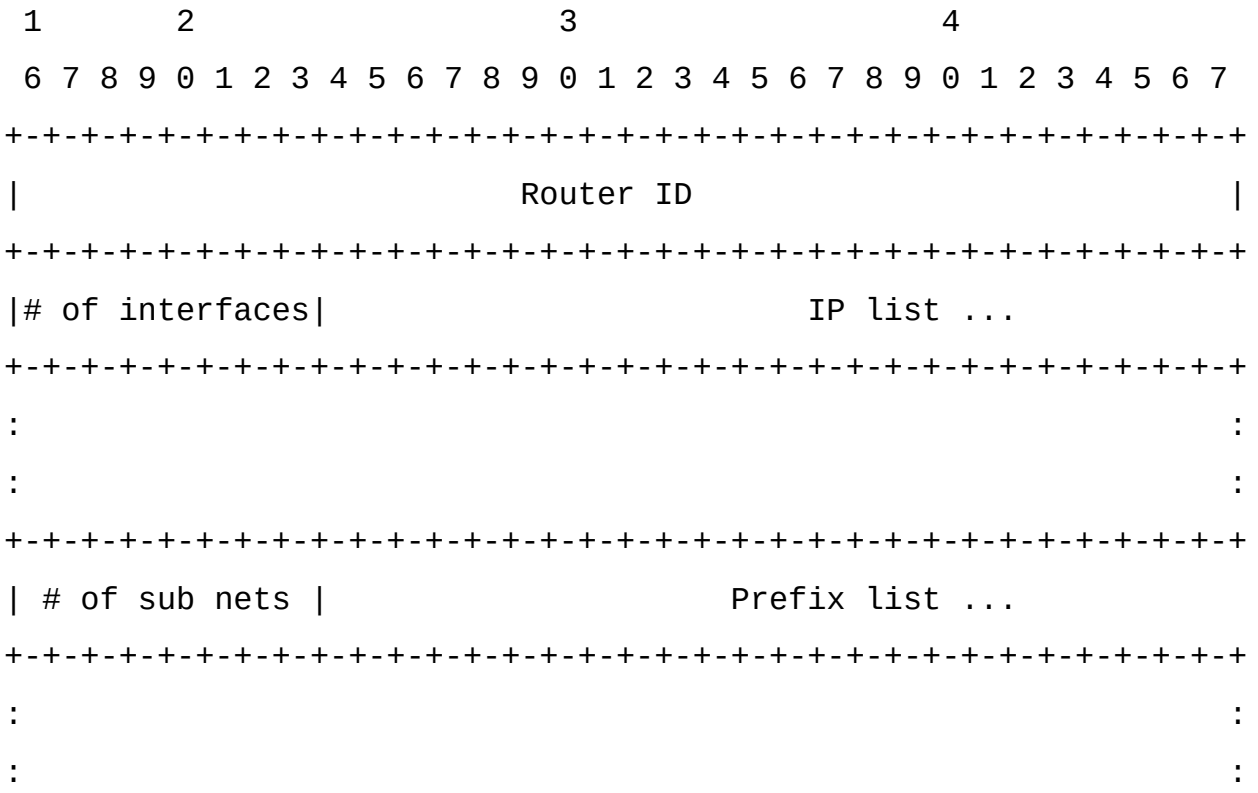
Other parameters are defined by their names. Note that the Link Cost field will be used only for link-up events.

2.1.5 Interface and/or Prefix Declaration (IPD)

Interface and/or Prefix Declaration (IPD) is used to broadcast the information of interfaces and/or subnets attached to a node.

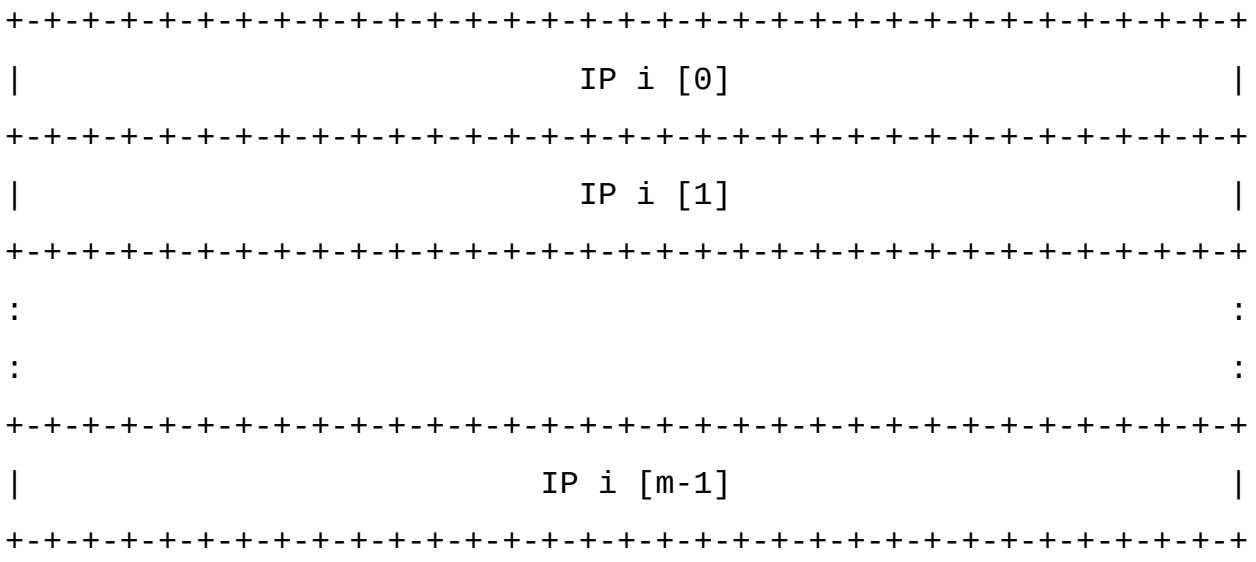


One node description is used to describe the interface(s) or/and subnets attached to one node:

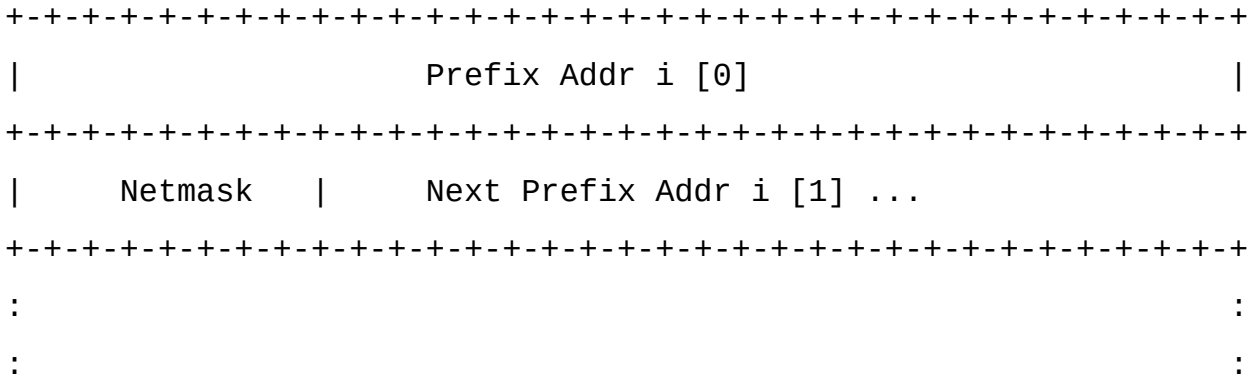


Here, the IP list represents all the interfaces that are broadcasted by node i and the prefix list shows all the sub nets that are broadcasted by node i. The format of these two lists are shown as follows.

The IP list is:



The prefix list is:



Note that the Netmask field is an integer. For example, netmask 32 represents 255.255.255.255 and netmask 24 represents 255.255.255.0.

2.2. Packet Processing and Broadcasting

This protocol provides nodes with full copy of the network topology. Nodes run an algorithm to decide whether it is in the MCDS. MCSD nodes can re-broadcast first seen control messages. Details are presented in the following sections.

2.2.1. Minimal Connected Dominating Set (MCDS)

Approximation algorithms for minimal connected dominating set (MCDS) are presented in this section. The global algorithm can be used when the channel error rate is low and the local algorithm is recommended when the channel error rate is high.

2.2.1.1 Global CDS Algorithm

The global CDS algorithm assumes that all nodes maintain local copies of the entire topology. If the channel error rate is high, the distributed topologies may require long converge time. Thus, this algorithm is recommended in an environment with low channel communication error rate.

Every node runs the same algorithm in two stages. All nodes will be examined in the first stage. We categorize MCDS nodes into two types. POTENTIAL MCDS node implies that a node has a high possibility of being a MCDS node, while PRIMARY MCDS node means that this node has to be in the MCDS. PRIMARY MCDS nodes include all the nodes that have

at least one close neighbor set and at least one neighbor that is not in that neighbor set. A node i becomes a POTENTIAL MCDS node if all of its neighbors form a close neighbor set, and node i has the largest degree comparing with other neighbors. In case of a tie, the node with larger ID wins. This rule also breaks ties in the following cases. (Note that if the network is fully connected, no one will be chosen as a MCDS node.) If an open neighbor set of node i is detected, and one exit node that covers all neighbors has larger degree than node i, node i will maintain its type. Otherwise, it is a POTENTIAL MCDS. For all the other cases, nodes become non-MCDS nodes. Note that a node with degree of 1 is always a non-MCDS node. [1]

This algorithm assigns MCDS or non-MCDS nodes with proper types in the first stage. In the second stage, it will check POTENTIAL MCDS nodes by the ascending order of node ids. (We tried using the order of node degrees, but it does not improve the performance in term of the average size of MCDS.) For each POTENTIAL MCDS node, if its MCDS neighbors cover all of its neighbors, and they are connected in the entire potential MCDS, this node becomes a non-MCDS node. [1]

2.2.1.2 Local CDS Algorithm

The local CDS algorithm constructs the MCDS only based

on the node set including nodes that are in the three-hop range. Therefore, if the channel error rate is high, the distributed MCDS results can converge as long as nodes have correct three-hop range information, which does not require a long time. Therefore, this algorithm is recommended in an environment with high packet error rate.

Every node runs the same algorithm in two stages. Only nodes within two-hop range will be examined in the first stage. The algorithm in this stage is the exactly same as the one in the first stage of the global CDS algorithm presented in 2.2.1.1.

The local CDS algorithm assigns MCDS or non-MCDS nodes with proper types in the first stage. In the second stage, if the center node is a POTENTIAL MCDS node, its MCDS neighbors will be checked. If its MCDS neighbors with larger node IDs and PRIMARY MCDS neighbors with smaller IDs cover all of its neighbor nodes, and these MCDS neighbors are connected, the center node becomes a non-MCDS node.

2.2.2. Neighbor Sensing

If the link layer can provide the neighbor detection, the function described in this section can be ignore (Refer to Section 3).

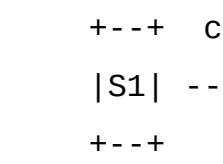
A new link is considered as stable iff the number of control package (a package may contain one or more control messages) heard recently (within Dead Interval time, which is defined in the following paragraph) reaches the number_of_messages_ heard_before_reply (default value is 2). The neighbor sensing starts to work only after the new link is stable.

OSPF-MCDS uses a simple hello protocol to detect neighboring nodes. A router can have multiple interfaces that are used for OSPF-MCDS routing. Each interface periodically broadcasts a HELLO message. The periodic time for HELLO messages is a user-defined value, named Hello Interval. If a new HELLO message is received (after the corresponding link is stable), the sender IP will be added to the receiver's next HELLO message as the router ID for that new neighbor. If a node receives a HELLO message containing its ID, a two-way connection is up. When the sender has smaller node ID, the receiver sends a link LSD Link UP if there is no LDD message to be sent soon (Refer to Section 2.2.3.). Otherwise, the new neighbor's IP address will be included in the next Hello.

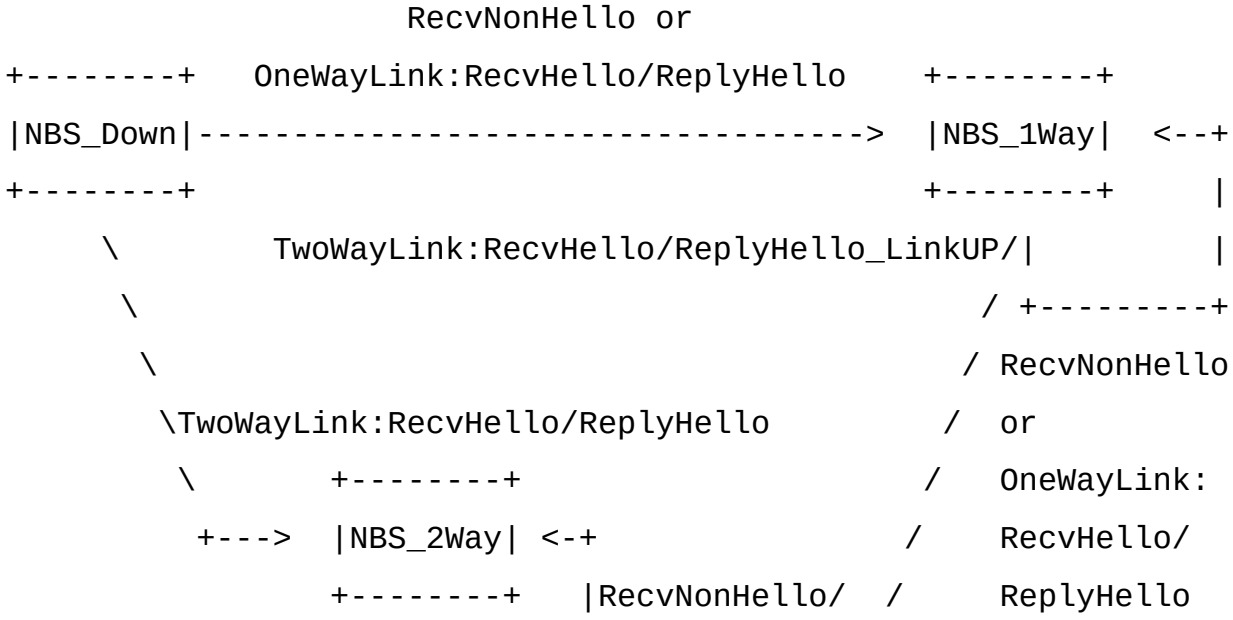
If an expected HELLO message is missing in a period of Dead Interval time, the generator of that HELLO message is considered as lost. In other words, a link is down. In the implementation, a counter is set for each neighbor, recording the number of missing HELLO messages. If a HELLO message is received from this neighbor, this counter is reset to

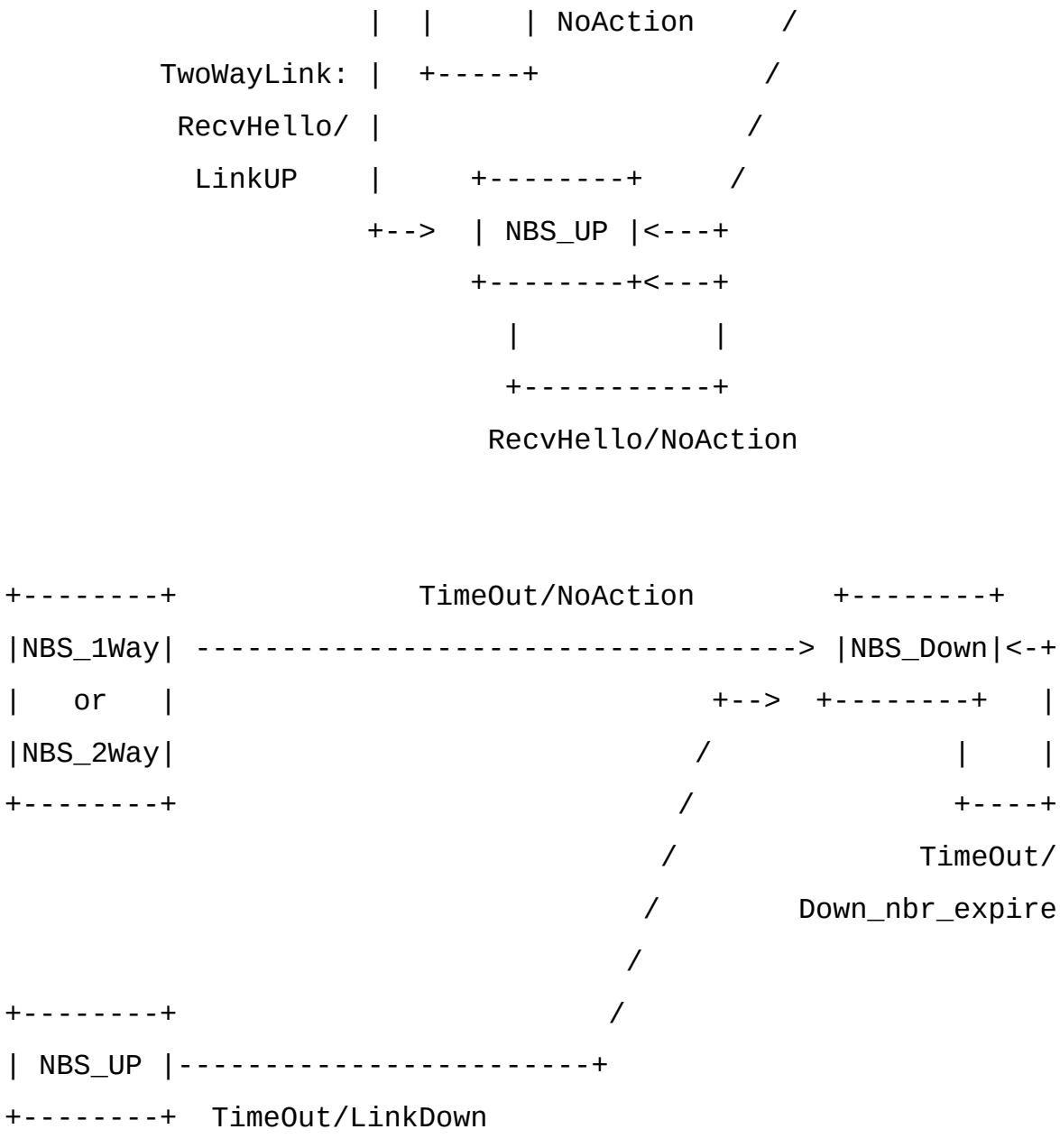
zero. Each time when the center node generates a HELLO message, this counter is increased by one. The neighbor expires once the counter reaches the maximum allowed value. In this draft, we named the maximum number of allowed missed HELLO messages the Dead Interval.

Assume that where S1 is the previous state, S2 is the next state, c (if not NULL) is the condition, e is the event, and a is the corresponding action. Therefore, an FSM entry S1: c:e/a S2 can be represented by following example:



The finite state machine for neighbors is demenstrated below.





2.2.3. Topology Update

A topology update is triggered when a link is up or down. Link State Description (LSD)is used for this procedure.

When a new link is up, the neighbor node with larger ID set the sequence number and send an LSD message to declare the link-up event. When an old link is down, both end nodes broadcast the Link DOWN event using an LSD message. Note that in case of a link-up, the neighbor node does not send an LSD message if there is only two nodes in its link state database. In other words, all known nodes in the network have the knowledge of this new link and there is no need to broadcast it. Similarly, in case of a link down event, a neighboring node of that link does not send an LSD message if there is no neighbor node in its neighbor list.

If a LDD periodic message is going be to sent soon (Refer to Section 2.2.4 for the definition), an LSD Link UP message is not forwarded. Otherwise, when the GCDS algorithm is used, an MCDS node rebroadcast any first-seen LSD messages. If the local CDS algorithm is used, an MCDS node rebroadcast a LSD Link UP message if this new link is within the three-hop range, i.e., the minimum hop counts from this MCDS node to any end node of this new link is equal or less than 3. This is used to guarantee the identical calculation results of MCDS nodes at distributed nodes. If the LSD describes a link down event, it is forwarded by MCDS nodes even if the local CDS algorithm is used. This is used to guarantee that no one uses this link in its shortest path tree. In case of a link change, the shortest paths to all destinations are recalculated.

When an edge is inserted into the local database via a LSD message, it is possible that the next synchronization message for this link might not come in tie since the LSD is transmitted immediately while the link synchronization is propagated via MCDS nodes which may has a longer delay. Therefore, the first synchronization is expected to arrive longer than ordinary next synchronization. To handle this problem, the link expiration time for this link is set to be twice as the normal expiration time (Refer to Section 2.2.4 for the setting of expiration for edges).

Proof in [3] shows that by the schemes described in this section and Section 2.2.4, OSPF-MCDS can work properly without the guaranteed delivery support from the underlying MAC layer protocols.

2.2.4. Topology Synchronization

There is a counter associated with the HELLO protocol. When a HELLO message is sent, the counter is increased by one. When the counter reaches a pre-defined maximum value, say 3, the counter is reset and the node also sends a periodic broadcast of its local link state database. When the counter is smaller than the pre-defined maximum value by one, we say that this node is going to broadcast link state database soon. In this case, there

is no LSD Link UP message to be forwarded or sent by this node before the counter reaches the maximum value (refer to Section 2.2.3).

Non-MCDS nodes only broadcast part of its neighbor list in which nodes are non-MCDS and have smaller node IDs. The senders increase the sequence numbers of these edges by one before the control message is generated.

MCDS nodes broadcast its neighbor list and full copy of their local link state databases using LDD messages. MCDS nodes also increase the sequence numbers of neighboring edges that connect to neighbors with small IDs.

Note that a neighbor node is not added into the LDD message if the counter for that neighbor is one less than the maximum allowed missing HELLO messages. Similarly, if the counter for an edge is one less than the maximum allowed missing synchronization, this edge is excluded from the corresponding LDD message.

When a node receives an LDD message, the sequence number of each link state description in this message is checked with the local copy. If the received one has a later sequence number (refer to Section 8), the local link state is updated. If there is no local copy for one link entry, this link is considered to be a new link and

processed as if a LSD Link UP message is received.

Similar to the counter used for neighbors, there is also a counter for each edge. If an edge is refreshed by a LDD message, i.e., the received link state entry is later than the local copy, the counter is reset to zero. Each time when a node generates a periodic broadcast message, the counters for all edges are increased by one. If the counter for a link reaches the maximum allowed value, this edge expires in the local link state database. This maximum allowed value is called as Synchronization Time. Expired edges are stored in a temporary database together with their sequence numbers. This is because that these edges may not expire at a neighbor node. If this neighbor node broadcasts these edges, the receiver can detect that these are expired edges based on the temporary database.

If any link expires before the generating of the next broadcast message, the MCDS is re-selected and the shortest paths to all possible destinations are re-calculated.

2.2.5. Sending Control Messages

Messages are buffered if there is no HELLO message waiting to be sent. In other words, only HELLO message can trigger the sending of control messages. This is

used to combine multiple control messages into one packet and these messages can share the same IP and MAC headers which usually have large sizes compared to our control messages.

2.2.6. MCDS Maintaining When Connectivity Changes

Before a LDD is sent, the local link state database is visited. If there is a link state changes, the MCDS is re-calculated.

2.2.7. Topology Discovery

OSPF-MCDS uses the Dijstra shortest path routing algorithm to find routes to all possible destinations. Here the cost of each link is defined by the sum of the costs of two end nodes (more specifically, it is the sum of costs of two sender interfaces). Based on previous sections, a full copy of the network topology is maintained in every nodes. Therefore, optimum routing is possible.

2.2.7.1. Shortest Path Routing

OSPF-MCDS uses Dijstra algorithm. Detail is omitted.

2.2.7.2. Associated Networks and Hosts

Nodes use Interface and/or Prefix Declaration (IPD) packets

to declare all associated interfaces and network prefixes. By this control message, OSPF-MCDS can work with traditional wired network. User can use a configuration file to indicate which associated interfaces and/or sub nets take part in the OSPF-MCDS routing. OSPF-MCDS broadcasts the IPD messages to all nodes in the network via the MCDS.

2.3. Multiple Interfaces

We assume that any node has at most one interface in one subnet. One copy of OSPF-MCDS runs in one subnet. If a node connects to more than one subnet, this node is called a gateway node. It converts the link state information in one subnet into net information and forward them to other subnets in the IPD format.

Assume two gateway nodes, say i and j, both connect to two clusters, say C1 and C2. If node i receives a IPD originated from j in C1 which contains some net or nice addresses in C2, node i simply skips these destinations when it converts its link state database in C1 for C2. This is used to avoid potential deadlock between C1 and C2.

2.4. Collision Avoidance

Nodes use a random Jitter to adjust the time to send control messages. The random adjust time is uniformly distributed between [-Jitter, +Jitter]. Users can define their own

Jitter values. This will help nodes to avoid synchronized replies to a certain control message.

3. Clustering

TBD.

4. Optional Link Layer Notification

If link layer provides notifications for new link-up or old link-down events, OSPF-MCDS can work without the hello protocol. In other words, OSPF-MCDS does not need the hello protocol. This can save bandwidth. If a packet forwarded to a neighbor failed due to link down, the shortest path tree can be recalculated and the packet can be forwarded to another proper neighbor following a new shortest path. This can improve throughput and end-to-end delay metrics.

5. Security Considerations

TBD.

6. IPv6 Considerations

TBD.

7. Proposed Values for the Constants

Hello Interval (interval time between HELLO packets):	1.0 sec
Dead Interval (number of HELLO):	3
Timeout (number of HELLO):	3
Synchronization Time (number of HELLO):	3
Jitter:	0.2 sec
number_of_messages_heard_before_reply:	2

The default values here are experiment values, not recommended values.

8. Sequence Numbers

The sequence number from 0x0, and is increased by one after it is used for updating new link states. Note that length of the binary sequence number is 8 bits. So when the sequence number reaches the maximum value, it will change back to 0x0.

Assume that we try to compare two sequence number, say s1 and s2. If (s1 > s2 and s1 - s2 < Max_Sequence_Number/2) or (s1 < s2 and s2 - s1 > Max_Sequence_Number /2), we say s1 is larger than s2.

9. Acknowledgments

The authors wish to thank Luiz DaSilva, Nat Davis, Michael Christman, Kaustubh Phanse, and John Wells of Virginia Tech

for their contributions to this research. This research is supported in part by the Office of Naval Research through the Navy Collaborative Integrated Information Technology Initiative (NAVCIITI).

10. Authors' Addresses

Tao Lin
Bradley Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061
USA

Email: taolin@vt.edu

Scott F. Midkiff
2040 Torgersen Hall (0281)
Bradley Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061
USA

Email: midkiff@vt.edu

Jahng S. Park
Bradley Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061

USA

Email: jspark@vt.edu

References:

1. T. Lin, S. F. Midkiff, and Jahng S. Park. "Approximation Algorithms for Minimal Connected Dominating Sets and Application with Routing protocol in Wireless Ad Hoc Network," in proceeding of IEEE IPCCC, pp. 157-164, 2003.
2. T. Lin, S. F. Midkiff, and Jahng S. Park. "A Framework for Wireless Ad Hoc Routing Protocols," in proceeding of IEEE WCNC, vol. 2, pp. 1162-1167, 2003.
3. T. Lin, Ph.D. Dissertation. "Mobile ad-hoc network routing protocols: Methodologies and Applications." ECE Department, Virginia Tech, March 2004.
4. J. Moy. (April, 1998). "OSPF version 2." Internet Engineering Task Force (IETF). [Online]. Available: http://www.ietf.org/rfc/rfc2328.txt.
5. S. Bradner. (March 1997) "Key words for use in RFCs to Indicate Requirement Levels," Internet Engineering Task Force (IETF). [Online]. Available: http://www.ietf.org/rfc/rfc2119.txt.

Curriculum Vitae

Tao Lin earned his B.S. degree in Automation Department from Tsinghua University, P. R. China in July 1998. He spent one and half year on the beautiful Hawaiian island of Oahu and got his M.S. degree in Electrical Engineering from University of Hawaii at Manoa. He then joined the Ph.D. program in the Bradley Department of Electrical and Computer Engineering at Virginia Tech. He is a student member of the IEEE and Sigma XI. His career plan is to continue his research in the wireless networking area.

Publications

1. T. Lin, S. F. Midkiff, J. S. Park, and Y. Lin, "An Analytical Study of Connectivity Changes in Mobile Ad-Hoc Network Simulations," in *Proceedings of Communication Networks and Distributed Systems Modelling and Simulation Conference (CNDS)*, San Diego, CA, USA, January 2004.

2. T. Lin, S. F. Midkiff, and J. S. Park, "Minimal Connected Dominating Set Algorithms and Application for a MANET Routing Protocol," in *Proceedings of the IEEE International Performance Computing and Communications Conference (IPCCC)*, pp. 157-164, Phoenix, AZ, April 2003.

3. T. Lin, S. F. Midkiff, and J. S. Park, "A Framework for Wireless Ad Hoc Routing Protocols," in *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, vol. 2, pp. 1162-1167, New Orleans, LA, March 2003.

4. T. Lin and S. F. Midkiff, "Mobility versus Link Stability in the Simulation of Mobile Ad Hoc Networks," in *Proceedings of the Communication Networks and Distributed Systems Modelling and Simulation Conference (CNDS)*, pp. 3-8, Orlando, FL, January 2003 (invited paper).

5. T. Lin, S. F. Midkiff, and J. S. Park, "A Dynamic Topology Switch for the Emulation of Wireless Mobile Ad Hoc Networks," in *Proceedings of the IEEE Local Computer*

Network (LCN), Wireless Local Network Workshop, pp. 791-798, Tampa, FL, November 2002.

6. T. Lin and G. H. Sasaki, "Nonblocking WDM Networks with Fixed-Tuned Transmitters and Tunable Receivers," in *Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing*, pp. 400-401, Monticello, IL, September 1999.

7. G. H. Sasaki and T. Lin, "Minimal Cost WDM Network for Incremental Traffic," in *Proceedings of the SPIE All-Optical Networking Conference: Architecture, Control, and Management Issues*, vol. 3843, pp. 173-180, Boston, MA, September 1999.

8. G. Sasaki and T. Lin, "A Minimal Cost WDM Network for Incremental Traffic," in *Proceedings of the IEEE Information Theory and Communication Workshop (ITW)*, pp. 5-7, Kruger National Park, South Africa, June 1999.

9. L. A. DaSilva, S. F. Midkiff, J. S. Park, G. Hadjichristofi, K. Phanse, T. Lin, and N. J. Davis, "Network Mobility and Protocol Interoperability in Ad Hoc Networks," journal paper submitted, 2004.