1-12-2006

Topology Control, Routing Protocols and Performance Evaluation for Mobile Wireless Ad Hoc Networks

Hui Liu

TOPOLOGY CONTROL, ROUTING PROTOCOLS AND PERFORMANCE EVALUATION FOR MOBILE WIRELESS AD HOC NETWORKS

by

HUI LIU

Under the Direction of Yi Pan

ABSTRACT

A mobile ad-hoc network (MANET) is a collection of wireless mobile nodes forming a temporary network without the support of any established infrastructure or centralized administration. There are many potential applications based the techniques of MANETs, such as disaster rescue, personal area networking, wireless conference, military applications, etc. MANETs face a number of challenges for designing a scalable routing protocol due to their natural characteristics. Guaranteeing delivery and the capability to handle dynamic connectivity are the most important issues for routing protocols in MANETs. In this dissertation, we will propose four algorithms that address different aspects of routing problems in MANETs. Firstly, in position based routing protocols to design a scalable location management scheme is inherently difficult. Enhanced Scalable Location management Service (EnSLS) is proposed to improve the scalability of existing location management services, and a mathematical model is proposed to compare the performance of the classical location service, GLS, and our protocol, EnSLS. The analytical model shows that EnSLS has better scalability compared with that of GLS. Secondly, virtual backbone routing can reduce communication overhead and speedup the routing process compared with many existing on-demand routing protocols for routing detection. In many studies, Minimum Connected Dominating Set (MCDS) is used to approximate virtual backbones in a unit-disk graph. However finding a MCDS is an NP-hard problem. In the dissertation, we develop two new pure localized protocols for calculating the CDS. One emphasizes forming a small size initial near-optimal CDS via marking process, and the other uses an iterative synchronized method to avoid illegal simultaneously removal of dominating nodes. Our new protocols largely reduce the number of nodes in CDS compared with existing methods. We show the efficiency of our approach through both theoretical analysis and simulation experiments. Finally, using multiple redundant paths for routing is a promising solution. However, selecting an optimal path set is an NP hard problem. We propose the Genetic Fuzzy Multi-path Routing Protocol (GFMRP), which is a multi-path routing protocol based on fuzzy set theory and evolutionary computing.

INDEX WORDS: Wireless Mobile Ad-hoc networks, Routing Protocol, Position-based Routing, Connected Dominating Set, Multipath Routing, Performance

TOPOLOGY CONTROL, ROUTING PROTOCOLS AND PERFORMANCE EVALUATION FOR MOBILE WIRELESS AD HOC NETWORKS

by

HUI LIU

A Dissertation Submitted in Partial Fulfillment of Requirements for the Degree of

Doctor of Philosophy

In the College of Arts and Sciences

Georgia Stage University

2005

TOPOLOGY CONTROL, ROUTING PROTOCOLS AND PERFORMANCE EVALUATION FOR MOBILE WIRELESS AD HOC NETWORKS

by

HUI LIU

ACKNOWLEDGMENTS

The dissertation would not have been possible without the help of so many people. I would like to take this opportunity to express my deep appreciation to all those who helped me in this hard but extremely rewarding process.

First and foremost, I would like to thank my advisor, Professor Yi Pan, for all his help, advice, support, guidance, and patience. During four years study, sometimes I may be tired and disappointed but I always believe that Dr. Pan was there to consult, to clear my thoughts, and to seek strength and direction from. He always found time whenever I needed an intelligent discussion. He always encouraged me whenever I was frustrated. He always raised the bar inspiring me to strive for the best, and was always among the first ones to applaud any accomplishment. I have been very lucky to work with him.

I am grateful to Dr. Michael Weeks, Dr. Anu Bourgeois, and Dr. Yichuan Zhao for serving on my Ph.D committee, and for their time and co-operation in reviewing this work. I come a long way as far as technical writing is concerned, and Dr. Bourgeois and Dr. Pan help me a lot for improving my writing skills. I would like to thank Dr. Zhao specially for providing a jump-start to the analytical modeling aspects related to my research. Special thanks also go to Dr. Weeks for his proof-reading of the draft and thanks to him this final version is more sound and readable.

I have been really fortunate to study with a group of really smart and fun people. To the current and past members of the Yamacraw projects in Computer Science

Department of our university my special thanks. I also thank Dr. Raj Sunderraman for his great patience and support.

Outside of work, my stay in campus has been truly memorable. I got introduced to the wonderful sand volleyball game, and played badminton very often. The year with service on Chinese Student Union is very unforgettable. Yuanchen He, Yuchun Tang, Hai Deng, Hongli Geng, Bill Zhong, Wei Zhong, Feng Tan, Shasha Luo, and others, thank for your excellent company and unselfish help. I never feel lonely and arduous because of you although I am far away from my homeland and family.

I am very lucky to have a great brother, Dr. Yi Liu, and my sister-in-law, Qiang Le. I want to tell you, my dear brother and sister, I was so happy to see you when I was on the land of Atlanta first time. Special thanks also go to Jie Huang for his encouragement.

The more important thanks are reserved for the last. Many thanks to my parents, Feng Liu and Yuhua Peng, for their constant support, concern, and motivation. Dear Daddy and Mom, the years around you are the most wonderful and happiest time during my life. I miss you very much.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

Personal Digital Assistants	PDAs
Wireless Mobile Ad Hoc Network	MANET
Global System for Mobile Communication	GSM
Code Division Multiple Access	CDMA
Universal Mobile Telecommunications System	UMTS
Wireless Local Loop	WLL
Wireless Local Area Network	WLAN
Packet Radio Network	PRNet
Survivable Radio Networks	SURAN
Global Mobile	GloMo
Quality of Service	QoS
The ACM Symposium on Mobile Ad Hoc Networking & Computing	MobiHoc
Internet Engineering Task Force MANET Working Group	IETF MANET WG
Enhanced Scalable Location management Service	EnSLS
Minimum Connected Dominating Set	MCDS
Connected Dominating Set	CDS
Genetic Fuzzy Multipath Routing Protocol	GFMSR
Global Positioning System	GPS
Wireless Routing Protocol	WRP
Least Resistance Routing	LRR

Global State Routing	GSR
Fisheye State Routing	FSR
Adaptive Link-state Protocol	ALP
Source Tree Adaptive Routing	STAR
Optimal Link State Routing	OLSR
Landmark Ad hoc Routing	LANMAR
Associativity-based Routing	ABR
Lightweight Mobile Routing	LMR
Signal Stability-based Adaptive Routing	SSA
Routing On-demand Acyclic Multipath Algorithm	ROAM
Multipath Dynamic Source Routing	MDSR
Relative Distance Micro-discovery Ad-hoc Routing	RD-MAR
Efficient Secure Dynamic Source Routing	ESDSR
Intrazone Routing Protocol	IARP
Interzone Routing Protocol	IERP
Zone Routing Protocol	ZRP
Geographical Routing Algorithm	GRA
Geographical Distance Routing Protocol	Gedir
Ultra High Frequency	UHF
Very High Frequency	VHF
Distributed Location Management	DLM
Distance Routing Effect Algorithm for Mobility	DREAM
Greedy Perimeter Stateless Routing Protocol	GPSR

Grid Location Service	GLS
Most Forward within Transmission Range	MFR
Nearest with Forward Progress	NFP
Relative Neighborhood Graph	RNG
Gabriel Graph	GG
Media Access Control	MAC
Unit Disk Graph	UDG
Maximal Independent Set	MIS
Multipoint Relaying	MPR
Marking Process by Classification of Neighbors	MPCN
Iterative Localized Algorithm for CDS	ILA_CDS
Open Shortest Path First	OSPF
Split Multipath Routing Protocol	SMR
Stability-based Multipath Routing Algorithm	SBMR
Route Discovery Packet	RREQ
Route Reply Packet	RREP
Evolutionary Ad-hoc On-demand Fuzzy Routing	E-AOFR
Fuzzy Logic Wireless Local Aware Multipath Routing	FLWLAMR
Fuzzy Inference System	FLS
Constant Bit Rate	CBR
User Datagram Protocol	UDP
Distributed Coordination Function	DCF
Ad Hoc On Demand Distance Vector routing	AODV

Dynamic Source Routing	DSR
Temporally Ordered Routing Algorithm	TORA
Zone Routing Protocol	ZRP
Location-Aided Routing	LAR
Personal Area Network	PAN
Routing Internet Protocol	RIP
Genetic Algorithm	GA
Membership Function	MF

CHAPTER 1

INTRODUCTION

1.1 Research Motivation and Contributions

In recent years, pervasive computing has enjoyed a tremendous rise in popularity. New small mobile computing devices such as Personal Digital Assistants (PDAs), cell phones, handhelds, and wearable computers enhance information processing and accessing capabilities. Moreover, we will see smaller and more powerful computers embedded in every traditional appliance, e.g. digital camera, cooking oven, washing machine, and thermostats, with computing and communication powers attached. With this in view, just like the Internet empowered a whole new world of applications for desktop computers, the development of robust pervasive computing will enable exciting new possibilities for the computing devices of the future.

A fundamental building block of pervasive computing likely includes wireless mobile networking. Wireless mobile networks have traditionally been based on the cellular concept and relied on good infrastructure support, in which mobile devices communicate with access points like base stations connected to the fixed network infrastructure. Typical examples of this kind of wireless mobile networks are GSM, CDMA, UMTS, WLL, WLAN, etc. However, this kind of infrastructured wireless mobile network is not suited in many contexts, where there does not exist any fixed infrastructure or to deploy one is not cost effective. For example, battlefield conditions, deserts, and earthquake rescue etc. In these contexts, it is required that a wireless network can be set up on-demand, automatically, and instantly. Thus, a wireless mobile ad hoc network (MANET) has attracted more and more interest as an infrastructureless approach to support truly pervasive computing. A MANET is a collection of wireless nodes that can dynamically form a temporary network to exchange information without any pre-existing fixed network infrastructure or centralized administration.

The term "MANET" is relatively new, but the concept of mobile packet radio networks, where every node in the network is mobile and where wireless multihop (store-and-forward) routing is utilized, dates back to the seventies. In 1972 DARPA initiated a research effort to develop and demonstrate a packet radio network (PRNet)[46], which was to provide an efficient means of sharing a broadcast radio channel as well as coping with changing and incomplete connectivity. The initial PRNet protocols adopted a centralized control station, but the core PRNet concept quickly evolved into a distributed architecture consisting of a network of broadcast radios with minimal central control, using multihop store-and-forward routing techniques to create complete end-user connectivity from incomplete radio connectivity. The original ideas about MANET came out during the research of PRNet. In order to enhance the robustness and scalability, DARPA initiated the Survivable Radio Networks (SURAN) program in 1983. With the widespread implementation of Internet and Web technology, both commercial and defense sectors are motivated to extend the global information infrastructure into the mobile wireless environment, DARPA funded the Global Mobile (GloMo) Information Systems program in 1994 [57] that has just recently concluded. The goal of the GloMo program [88] was "to make the mobile, wireless environment a first-class citizen in the defense information infrastructure by providing user friendly connectivity and access to services for mobile users". Interest in the improvements based packet radio networks was not limited to the United States. Many countries have proposed their own solutions. MANET is a new network architecture that absorbed the advantages of the previous research results.

While the original motivation for MANET was military needs, its non-military applications have grown substantially since the mid-1980s. Police, fire and rescue, disaster relief, robotics, space, distributed sensors, and impromptu team communications are a few possible applications of MANET technology because in these application scenarios, information exchange between mobile hosts cannot rely on any fixed network infrastructure, but on rapid configuration of wireless connections on-the-fly.

MANET is a self-configuring network of wireless links connecting mobile hosts, which may be routers and terminals at the same time. It has a lot of special features such as distributed administration, multihop routing, self-organized architecture, and limited resources etc. Regardless of the attractive applications, the features of MANET introduce a host of unresolved research issues related to the realization of a global MANET. These research issues include routing, security and reliability, QoS, power management, and etc, which involve every network layer. Research in the area of MANET is receiving much attention from academia, industry, and government. So far, many international conferences and workshops on MANET have been held by e.g. IEEE and ACM. For instance, MobiHoc (The ACM Symposium on Mobile Ad Hoc Networking & Computing) has been one of the most important conferences of ACM SIGMOBILE (Special Interest Group on Mobility of Systems, Users, Data and Computing). Among the many research issues, it has been widely recognized that routing strategy is the most important. To determine viable routing paths and deliver messages in a decentralized manner where network topology changes dynamically is far less certain than a well-defined problem. It is required to not only design new models that describe the features of the target MANET very well, but also propose new algorithms that are able to safely and efficiently route information to mobile destinations, in order that MANET can support different types of multimedia applications. Factors such as unpredictable wireless link quality, propagation path loss, fading, multiuser interference, power expense, and topological changes become relevant issues that add more difficulties and complexities to the routing protocol design. Therefore, the Internet Engineering Task Force MANET Working Group (IETF MANET WG) has been formed since 1996, aiming to investigate and develop candidate standard Internet routing support for mobile, wireless IP autonomous segments and develop a framework for running IP based protocols in ad hoc networks. There are nearly a dozen candidate routing protocols [87] currently being discussed within the MANET WG for achieving this goal [66]. They serve the purpose of demonstrating the functionality and performance of ad hoc routing with comparatively simple protocols, whereas very few of them can be regarded to really fulfill the requirements of a real application scenario.

In this research we would like to propose three interesting algorithms that address different aspects of routing problems in MANETs. Firstly, in position based routing protocols, to design a scalable location management scheme is inherently difficult, because it would represent a functional deadlock problem; it is impossible to obtain the position information of a specific node without a position server, on the other hand, without position information, how to reach the position server. Enhanced Scalable

Location management Service (EnSLS) is proposed to improve the scalability of existing location management services. We derive an analytical model to compare the control overhead of our location services with another existing location service, and the result shows our EnSLS is more scalable than others. Secondly, virtual backbone routing can reduce communication overhead and speedup the routing process compared with many existing on-demand routing protocols for routing detection. In many studies, Minimum Connected Dominating Set (MCDS) is used to approximate virtual backbones in a unit-disk graph. However finding a MCDS is a NP-hard problem. In this research, we propose two protocols for calculating the CDS. Our new protocols largely reduce the number of nodes in CDS compared with existing methods. We show the efficiency of our approach through both theoretical analysis and simulation experiments. Finally, using multiple redundant paths for routing is a promising solution. However, selecting an optimal path set is a NP hard problem. We propose the Genetic Fuzzy Multi-path Routing Protocol (GFMRP), which is a multi-path routing protocol based on fuzzy set theory and evolutionary computing.

1.2 Dissertation Organization

The primary focus of this dissertation is routing and topology control protocols in MANETs. Specially, the scalability, resource management, performance evaluation and application of MANET routing and topology control protocols are considered. The remainder of this dissertation is organized into chapters as following.

Chapter 2 provides the required background information by briefly discussing about MANETs and introduces routing problems in an MANET, and gives an overview of routing protocols in MANETs. Chapter 3 overviews the location-based routing protocols in MANETs, and points out the principles needed to afford scalable routing in MANETs. It considers location services and packet forwarding strategy altogether. Further, in this chapter we propose a novel location management service for mobile ad hoc networks, called Enhanced Scalable Location management Service (EnSLS). We propose a model to analyze the scalabilities of two location services, GLS and our new protocol EnSLS in Chapter 4. Theoretical analysis shows that EnSLS significantly reduces the control message overhead needed to manage the location information.

Chapter 5 presents the concept of virtual backbone routing in MANETs. It gives a simple survey about the approximation of virtual backbones in networks via connected dominating set. We compare the performances of different types of algorithms for constructing connected dominating sets. Chapter 6 proposes a new algorithm for constructing the virtual backbone for routing in MANETs via classification of neighbors. It prefers to select vertices, which can cover as many as possible as other vertices, to an initial dominating set during the marking process. At the same time, correctness and performance analysis of the new algorithm are presented. A five-phase distributed algorithm is illustrated in Chapter 7. The communication and computation complexities of this algorithm remain the same polynomial complexity as existing algorithms shown in theoretical analysis. Simulation results prove the protocol largely reduce the number of nodes in dominating sets, because our goal is to generate a small dominating set in order to speedup the routing process and decrease the communicating overhead.

Chapter 8 addresses the correlation of multiple routing objectives. It shows that multiple routing objectives can be met altogether if we consider correlated different route selection metrics at the same time. This chapter proposes a simple and effective protocol called Genetic Fuzzy Multi-path Routing Protocol (GFMRP), which considers the multiple correlated selection parameters, based on fuzzy set theory and evolutionary computing. GFMRP employs a simple load-balancing method to distribute the traffic load, by spreading data packets over all the paths of the reliable multiple paths set in a round-robin fashion. The performance of GFMRP is evaluated in terms of packet delivery ratio, average end-to-end delay, and the frequency of route rediscovery in *ns2* context. Chapter 9 describes the procedure of optimization of the fuzzy logic system taken in GFMRP.

Finally Chapter 10 summarizes the accomplishments of this study and relates them to the wider field. Some improvements are suggested as future directions of research.

CHAPTER 2

BACKGROUND

2.1 Concepts and Characteristics for Mobile Ad Hoc Networks

A mobile ad-hoc network is a particular type of wireless network in which an association of mobile nodes forms a temporary network, without any support of fixed infrastructure or central administration. Mobile nodes can control connections and disconnections by the distances between them and the willingness to collaborate during the formation of short-lived networks. That means a connection is achieved either through a single-hop radio transmission if two nodes are located within wireless transmission range of each other, or through relays by intermediate nodes that are willing to forward packets for them. Figure 2.1 describes an example of MANET including four mobile nodes. In this example, nodes S, B, and C are within the transmission ranges of each other. Thus, nodes S, B, and C are named neighbors, and they can communicate directly with each other. However, node D does not reside in the transmission range of node S. If node S wants to send data to node D, the data must be routed through the intermediate node, such as node C, which acts as the router between node S and node D. Routes between a source and destination may potentially contain multiple hops.

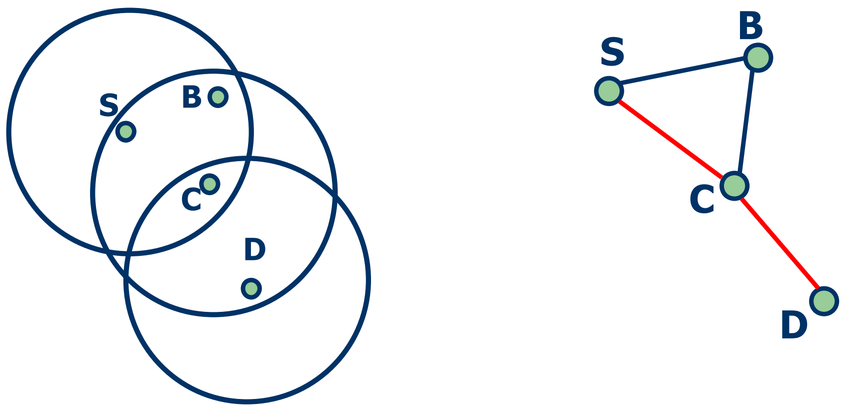


Figure 2.1. Example for mobile ad hoc networks

Although an autonomous MANET is useful in many scenarios, the integration of MANET with the Internet is desirable due to the growing interest in the Internet and associated technologies. A MANET may be connected at the edges to the fixed, wired Internet. Thus, MANETs will expand the present Internet, and wireless access to the same. A Mobile Internet Router is one of the main requirements for an ad-hoc network to gain access to the Internet. Under this scenario, the mobile terminals in ad-hoc networks might dynamically obtain and lose Internet connectivity through interfaces not participating in the MANET routing. Among mobile terminals, some of them can directly connect to the Internet and serve as Access Points for the rest of the mobile terminals in the Internet mobile ad-hoc network. Therefore, an Access Point will provide a gateway for the Internet, and is assumed to have access to any information. A typical scenario would be a laptop that might be connected to the Internet through an Ethernet link for a limited time while participating in a MANET through a wireless interface.

MANETs are distinguished from other communication networks by the following features:

1) Limited Resources

Battery is the only source of power for nodes in many ad hoc network environments, and the need to keep these nodes compact, light and even wearable, imposes limitation on their storage and processing capabilities. This is again very different from conventional wired networks, wherein the network nodes seldom depend on batteries as their sole source of energy, and typically have significant storage and processing capacity.

2) Mobile Nodes Play Multiple Roles

In most wired networks, network nodes play distinct roles, such as sources, destinations or routers. Also, nodes are typically dedicated to specific network operations and their characteristics well suited to the role they play. For example, machines are specifically designed to operate as servers, and dedicated high-end routers and switches are used to handle network traffic. On the other hand, in ad hoc networks, most nodes are expected to route packets for other nodes in the network, while they themselves may also be a source or destination for one or more application flows. A management framework must account for the multiple roles played by network nodes.

3) Dynamic Topology

The topology of a MANET can change very dynamically for various reasons. In MANETs, the topology changes as nodes move out of range of one or more nodes with which they were connected, and move closer and connect to other nodes. In addition, even in fixed wireless ad hoc networks (e.g., wireless sensor fields), due to limited survivability of wireless links (subject to fading or jamming) or of the nodes themselves (e.g., damaged due to hostile conditions, or discharged battery), the logical topology of the network may change.

Keeping current knowledge of the network topology is an important requirement in any network management system. In fixed wired networks, this is a relatively simple task since the changes in topology (mainly due to node or link failure, or addition/removal of a node) are infrequent. In a wireless ad hoc environment, it is crucial that the management system keep up with the frequent topology changes. However, the frequent exchange of topology information may lead to considerable signaling overhead, congesting low bandwidth wireless links, and possibly depleting the limited battery life of the nodes involved. Hence, the choice of mechanism used to collect or manage topology information is critical.

4) Low Bandwidth, Variable Capacity Links

Wireless links are typically more bandwidth-constrained than their wired counterparts. Fading, interference, jamming etc., may cause intermittent link failures or considerable variation in the channel error rate. In addition, the diverse nature of nodes (e.g., with different transmission power levels), and communication technologies (e.g., IEEE 802.11, direct line of sight UHF/VHF links, satellite links, etc.) being used may lead to links of varying capacity in multi-hop wireless networks.

5) Heterogeneity

Heterogeneity is inherent to most ad hoc networks due to the diverse nature of communication technologies (IEEE 802.11, line of sight UHF/VHF, etc.) that may be used and the different types of nodes – ranging from sensors, palmtops, laptops to mobile networks hosted on a ship, a tank or an airplane – that may form the network. The heterogeneity of nodes can be a criterion to assign roles (e.g., management server versus client) to the various nodes. An ad hoc network may also be a result of a multi-organization consortium and present additional interoperability challenges for a management system.

6) Limited Survivability

One of the major challenges in using ad hoc networks is their limited survivability and vulnerability to security attacks [109]. The use of a wireless medium for communication opens another venue for initiating link level attacks ranging from passive eavesdropping to message replay and message distortion. In addition, deployment of wireless ad hoc networks in diverse and often hostile environments (rapidly deployed military battle-site network, sensor fields used to collect sensitive data in remote, unmanned locations) makes these networks even more prone to network security attacks leading to failure of network elements.

2.2 Application of Ad Hoc Networks

In this section, we look at some of the potential applications for MANETs that might provide the basis for commercially successful products.

1) Military application

MANETs perfectly satisfy military needs like battlefield survivability. Wireless electronic devices carried in soldiers, tanks, airplane and other military equipment can form MANETs to support communication among them in order to collaboratively achieve military goals since there is not any pre-placed infrastructure and connectivity in battlefield environments. The research of MANETs originated from military application. In the future providing services for military field is still a hot topic.

2) Mobile conferencing

Perhaps the prototypical application requiring the establishment of a MANET is mobile conferencing. When mobile computer users gather outside their normal office environment, the business network infrastructure is often missing. But the need for collaborative computing might be even more important here than in the everyday office environment. As it turns out, the establishment of a MANET for collaborative mobile computer users is needed even when there may be Internet infrastructure support available. This results from the likely overhead required when utilizing infrastructure links, which may entail drastically suboptimal routing back and forth between widely separated office environments.

3) Emergency services

We are all familiar with situations in which loss of local power causes loss of electricity, and each year natural disasters destroy people's lives around the world. As the Internet grows in importance, the loss of network connectivity during such natural disasters will become an ever more noticeable consequence of the calamity. Furthermore, network applications will become increasingly important for emergency services, and thus it will be important to find ways to enable the operations of networks even when infrastructure elements have been disabled as part of the effects of a disaster.

4) Personal area networks

The idea of a personal area network (PAN) is to create a very localized network populated by some network nodes that are closely associated with a single person. These nodes may be attached to the person's belt or carried in a purse. More exotic visions of the future include virtual reality devices attached around the head and other devices more oriented toward the sense of touch. These devices may or may not need to have an attachment to the Internet, but they will almost certainly need to communicate with each other while they are associated with their users' activities. In this scenario, mobility is not the overriding consideration.

Actually Bluetooth is a commercial application of MANETs that is designed to eliminate wires between personal devices. Currently Bluetooth only provides short distance communication. However, when interactions between several PANs are needed, mobility becomes suddenly much more important. In other words, when people meet in real life, their PANs are likely to become aware of each other also. Methods for establishing communications between nodes on separate PANs could benefit from the technologies of MANETs.

5) Embedded computing applications

Some researchers predict a world of ubiquitous computing, in which computers will be around us, constantly performing mundane tasks to make our lives a little easier. These ubiquitous computers will often react to the changing environment in which they are situated and will themselves cause changes to the environment in ways that are predictable and planned. Ubiquitous intelligent internetworking devices that detect their environments, interact with each other, and respond to changing environmental conditions will create a future that is as challenging to imagine as a science fiction scenario. These capabilities can be provided with the use of MANETs.

2.3 Routing Problem of MANETs

Routing protocols are used to determine paths through the network so a data packet can get from its source, hop by hop, to its destinations. In general, one goal of routing is to choose a suitably "efficient" path, where efficiency can be measured in terms of end-to-end delay, packet delivery ratio, power expended, amount of self-interference, etc.

Routes between a source and a destination may potentially contain an ordered series of intermediate nodes that act as the routers. The multiple hops communication paradigm has three main performance advantages compared with single hop communication solution:

1) Adaptability. By deploying a multiple hop data forwarding network, packets can be routed around obstructions or areas captured by enemy units, which is very crucial for the battlefield scenario.
2) Spatial reuse [50]. Packet forwarding over multiple hops via small radii transmissions will exploit spatial reuse, by allowing multiple concurrent packet transmissions in different regions of the network, and maximize throughput.
3) Energy consumption efficiency. Packet forwarding via multiple small radii transmissions as opposed to a single large radius transmission will improve the throughput per unit energy [41].

In traditional hop-by-hop solutions to the routing problem, each node in the network maintains a routing table containing each destination with a corresponding next-hop node and link cost. Packets are forwarded by consulting the routing table for the next-hop node leading to the shortest path to the destination. In MANETs, the routing table at each node can be thought of as a view into part of a distributed data structure that, when taken together, describes the topology of the network. The goal of the routing protocol is to ensure that the overall data structure contains a consistent and correct view of the actual network topology.

One challenge in creating a routing protocol for ad hoc networks is to design a single protocol that can adapt to the wide variety of conditions that can be present in any ad hoc network over time. For example, the bandwidth available between two nodes in the network may vary from more than 10 Mbps to 10 Kbps or less. The highest speeds are achieved when using high-speed network interfaces with little interference, and the extremely low speeds may arise when using low-speed network interfaces or when there is significant interference from outside sources or other nodes' transmitters. Similar to the potential variability in bandwidth, nodes in an ad hoc network may alternate between periods during which they are stationary with respect to each other and periods during which they change topology rapidly. Conditions across a single network may also vary, so while some nodes are slowly moving, others change location rapidly.

Another challenge for routing is that mobility causes the next-hop node to be disconnected as nodes move in and out of transmission range. The result is that routes are frequently broken causing extra network traffic to reconstruct the routing table. If there is a high frequency of broken links, the overhead cost of routing can dominate the traffic load causing congestion and consuming precious energy in an attempt to discover unstable pathways. Thus routing protocols in MANET should have the capability to handle dynamic connectivity.

2.4 Overview of Routing Protocols in MANET

Many routing protocols have been proposed for MANETs. Those routing protocols are classified into two main categories: topology-based and position-based. Topology-based routing protocols are based on the information concerning links [62, 32]. In position-based routing protocols, mobile nodes know physical position information by geolocation techniques such as GPS [13, 47, 51, 108, 43, 60, 95]. In this section, a brief survey of various routing mechanisms is given.

Topology based routing protocols can be generally grouped by the routing strategy into one of three categories. The first type of topology based routing is a proactive routing protocol, which maintains network topology through the periodic exchange of control information. A proactive protocol might be appropriate in a network in which non-local communications are normal and route maintenance must be rapid. Some of proactive routing protocols are distance vector typed. Pure distance vector algorithms do not perform very well in MANETs due to the count-to-infinity problem [98], where two nodes both believe their route back to the reference node is through each other upon loss of a link. Thus, newly proposed protocols modify and enhance the distance vector algorithms, e.g., Distributed Bellman Ford [14, 27], Routing Internet Protocol (RIP) [67], etc. Proactive routing protocols of this type include the Destination-Sequenced Distance Vector (DSDV) Routing [80], Wireless Routing Protocol (WRP) [74], Least Resistance Routing (LRR) [82], and the protocol proposed by Lin and Liu [62]. Others of proactive routing protocols are based on link state [71, 72] algorithms. Link state based algorithms consist of Global State Routing (GSR) [18], Fisheye State Routing (FSR) [79], Adaptive Link-State Protocol (ALP) [29], Source Tree Adaptive

Routing (STAR) [30], Optimized Link State Routing (OLSR) protocol [42], and Landmark Ad Hoc Routing (LANMAR) [32]. In general, proactive protocols are not efficient enough due to much communication overhead when nodes are mobile.

The second type of topology based routing is a reactive routing protocol, which does not require maintaining a route to each destination of the network on a continual basis. Instead, routes are established on demand by the source. When a route is needed by the source, it floods a route request packet to construct a route. Upon receiving route requests, the destination selects the best route based on route selection metrics. In reactive routing protocols, control communication overhead is greatly reduced compared with proactive routing protocols since no effect is made to maintain the total network topology. Numerous protocols of this type have been proposed, such as Lightweight Mobile Routing (LMR) [19], Dynamic Source Routing (DSR) [45], Ad-Hoc On Demand Distance Vector (AODV) routing [81], Associativity-Based Routing (ABR) [99], Signal Stability-Based Adaptive (SSA) routing [25], Routing On-demand Acyclic Multipath (ROAM) algorithm [84], Multipath Dynamic Source Routing (MDSR) [75], Relative Distance Micro-discovery Ad Hoc Routing (RD-MAR) protocol [3], Efficient Secure Dynamic Source Routing (ESDSR) [6], etc.

The third type of topology based routing is hybrid in nature, which combines proactive and reactive techniques. As an example of a hybrid MANET routing protocol, ZPR [78] defines a zone around each node where the local topology is proactively maintained via the Intrazone Routing Protocol (IARP) [38]. When routes are required outside the local zone, a reactive route discovery mechanism is used via the Interzone Routing Protocol (IERP) [37]. This type routing protocols include the Zone Routing

Protocol (ZPR), the Bordercast Resolution Protocol, the Temporally Ordered Routing Algorithm (TORA) [77], and the Landmark Routing Protocol (LAN-MAR) [32].

The results in [70] show that the use of location information in MANETs significantly improves routing performance. With the knowledge of a node's geographical location, position based routing can be more effective at the cost of overhead required for exchanging location information. In position based routing, nodes maintain a location table that records the location of some other nodes and the time at which that location information was received. A sender node then uses this information to improve the efficiency in the transmission of data packets. Examples of position based routing protocols include the Location-Aided Routing algorithm (LAR) [51], the Distance Routing Effect Algorithm for Mobility (DREAM) [13], the Greedy Perimeter Stateless Routing algorithm (GPSR) [47], the Geographical Routing Algorithm (GRA) [43], the Geographic Distance routing protocol (Gedir) [60], and the GRID protocol [95].

CHAPTER 3

POSITION-BASED ROUTING SCHEMES

3.1 Background

Location-based routing approaches eliminate some of the limitations of topology-based routing through obtaining the node's geographic locations via the GPS [2] or some other types of positioning service. Recent research [47, 58, 108] shows that location-based routing can significantly reduce communication overhead compared with topology-based methods. Location-based routing protocols are likely to be more scalable, since each node only needs to know the location of the destination and its neighbors' locations to make a forwarding decision.

There are two important issues in geographic ad hoc routing protocols. One is the existence of scalable location services, which manage the position information of any mobile node at any time throughout the entire network. The current position of a specific node can be learned in a deterministic manner with the help of a position service such as GPS. Each mobile node registers its current position with a location service. Another is the packet-forwarding strategies, via which a node makes the forwarding decision based on the position of a packet's destination and the positions of the node's immediate one-hop neighbors. When a node does not have the position of its communication partner, it requires such information from a location service, and then embeds the position of destination into the header of data packets. However, the positions of the neighbors are typically learned through one-hop broadcasts. These beacons are broadcasted periodically by all mobile nodes and contain the position of the sending node. Then, a node forwards data packets to the destination node via packet-forwarding strategies. Examples of location-based routing protocols include DREAM [13], LAR [51], GPSR [47], GLS [58] and DLM [108]. In fact, to design a scalable location management scheme is inherently difficult because it would represent a functional deadlock problem; it is impossible to obtain the position information of a specific node without a location server, on the other hand, without position information, how would a node reach the location server?

We can distinguish three main packet-forwarding strategies for location-based routing, which are greedy forwarding, restricted directional flooding, and hierarchical approaches. In greedy forwarding, a node forwards a given packet to one of its neighbors that is located closer to the destination than the forwarding node itself. Generally, there are different optimization criteria a node can use to decide to which neighbor a given packet should be forwarded, such as most forward within transmission range (MFR) [97], nearest with forward progress (NFP) [40], and compass routing [53] etc. In restricted directional flooding, a node forwards a given packet to some not only one one-hop neighbors. Unfortunately, both forwarding strategies may fail if there is no one-hop neighbor that is closer to the destination than the forwarding node itself. In order to cope with this kind of failure, recovery strategies have been proposed that the packet should be forwarded to the node with the least backward (negative) progress [97] if no nodes can be found in the forward directions. Examples of recovery strategies are the face-2 algorithm [16] and the Greedy Perimeter Stateless Routing Protocol (GPSR) [47]. Hierarchical approaches are promising methods to enhance the scalability of traditional networks by establishing some form of hierarchy. Hierarchical forwarding strategies use greedy forwarding for wide area routing and non-position-based approaches for local area routing.

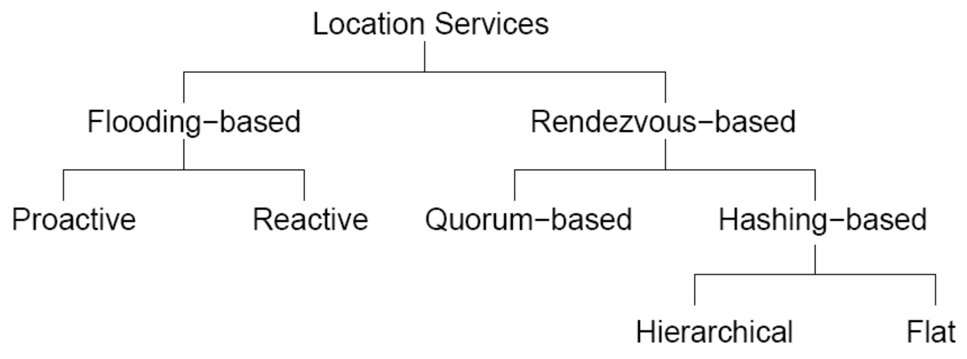


Figure 3.1 A taxonomy of location services

Furthermore, Figure 3.1 gives the taxonomy of the numerous location services proposed so far [23]. At the top level, location services can be divided into flooding-based and rendezvous-based approaches. Flooding-based protocols can be further divided into proactive and reactive approaches. In the proactive flooding-based approach, each (destination) node periodically floods its location to other nodes in the network each of which maintains a location table recording the most recent locations of other nodes. The interval and range of such flooding can be optimized according to the node's mobility and the "distance effect". For example, flooding should be more frequent for nodes with higher mobility, and flooding to faraway nodes can be less frequent than to nearby nodes. DREAM [13] serves as a good example of proactive flooding-based location services. In reactive flooding-based approaches [51], if a node cannot find a recent location of a destination to which it is trying to send data packets, it floods a scoped query in the network in search of the destination.

In rendezvous-based protocols, all nodes (potential senders or receivers) in the network agree upon a mapping that maps each node's unique identifier to one or more other nodes in the network. The mapped-to nodes are the location servers for that node. They will be the rendezvous nodes where periodical location updates will be stored and location queries will be looked up. Two different approaches of performing the mapping, quorum-based and hashing-based, have been proposed. In the quorum-based approach [39], each location update of a node is sent to a subset (update quorum) of available nodes, and a location query for that node is sent to a potentially different subset (query quorum). The two subsets are designed such that their intersection is non-empty, and thus the query will be satisfied by some node in the update quorum. Several methods on how to generate quorum systems have been discussed in [39].

In hashing-based protocols, location servers are chosen via a hashing function, either in the node identifier space or in the location space. Hashing-based protocols can be further divided into hierarchical or flat, depending on whether a hierarchy of recursively defined subareas is used. In hierarchical hashing based protocols (for example, [58, 108]), the area in which nodes reside is recursively divided into a hierarchy of smaller and smaller grids. For each node, one or more nodes in each grid at each level of the hierarchy are chosen as its location servers. Location updates and queries traverse up and down the hierarchy. A major benefit of maintaining a hierarchy is that when the source and destination nodes are nearby, the query traversal is limited to the lower levels of the hierarchy. Since the height of the hierarchy is $O(logN)$, effectively each node's location is disseminated to $O(logN)$ location servers. The best example of hierarchical rendezvous-based location service is GLS [58], which will be discussed in the following sections.

In flat hashing-based protocols (for example, [94, 103]), a well-known hash function is used to map each node's identifier to a home region consisting of one or more nodes within a fixed location in the network area. All nodes in the home region maintain the location information for the node and can reply to queries for that node; they serve as its location servers. Typically, the number of location servers in the home region is independent of the total number of nodes in the network, and thus effectively each node's location is disseminated to $O(1)$ location servers.

Li *et al.* proposed the Grid Location Service (GLS) [58]. In GLS, each mobile node maintains its location information in a number of location servers in each sibling region with IDs "closest" to its own ID. The distribution density of a node's location servers is not even. Thus, it has an advantage; the distances traversed by a location query are proportional to data packet path lengths between destination nodes and source nodes. However, when nodes move arbitrarily, GLS becomes less efficient. Because in selection of location servers, a node has to possibly scan the entire region to obtain the one with the "closest" ID as the location server, and in location maintenance, with nodes' entry or departure in a particular region, the scheme needs to check the entire particular region periodically and change the choices of location servers. All of these incur heavy signaling overhead. Yuan *et al.* proposed a Distributed Location Management (DLM) [108]. Within this scheme, a hash function is used to map a node's ID directly to a particular minimum partition, all nodes in this minimum partition are selected as the node's location servers, which reduces the signaling overhead without the process of scanning the entire region compared with GLS. DLM introduces hierarchical addressing models based on logical network partitions. Positions of nodes can have different accuracy levels. Consequently, only a small set of location servers need to be updated when a node moves while different location servers may carry position information of different levels of accuracy. However, location servers distribute evenly throughout the whole network. The distance traversed by a location query is independent of data packet path length. If there exist some void minimum partitions, DLM requires location query forwarding and hash function computation several times. It costs the available bandwidth of MANETs.

In this chapter, we propose a novel location management service for mobile ad hoc networks, called Enhanced Scalable Location management Service (EnSLS) [64], which combines the advantages of GLS and DLM. Firstly, EnSLS maintains the property: long distance queries are proportionally penalized. EnSLS maps a node's unique ID to the central coordination of a specific minimum region via a hash function. It takes the GPSR [47] as the underlying packet forwarding protocol, which allows that all packets destined for an arbitrary location (possibly unoccupied by a node) to be forwarded consistently to the same node in the neighborhood of that location [85]. Then the node receiving the location maintenance packet acts as one of the location servers. In the process of location server's selection and update, it only requires control packet forwarding and hash function computation once. All of these can significantly reduce the end-to-end delays and total signaling traffic needed to manage the location information.

Although each protocol is proposed along with some theoretical and/or simulation analysis, little is known about the relative performance of these protocols, especially in a realistic setting, i.e., a mobile environment. In Chapter 4, we derive a theoretical bound on the performances of GLS and EnSLS via a mathematical model, where GLS is a very classical location service in MANET.

The rest of this chapter illustrates the design and performance of our newly proposed scheme EnSLS. Section 3.2 gives a brief review of related work, including the packet forward algorithm: GPSR, which will be used in EnSLS. Section 3.3 gives the details of EnSLS. Optimization of our location service is illustrated in Section 3.4. We analyze and compare our scheme's routing performance scalability in Chapter 4.

3.2 Related Work

In this section, we introduce geographic forwarding and two related location services under study. For each location service, we describe how it (1) performs a location update in selected location servers, (2) performs a location query, and (3) maintains location information.

3.2.1 Geographic Forwarding: Greedy Perimeter Stateless Routing Algorithm (GPSR)

We use GPSR [47] for geographic packet forwarding. We assume each mobile node is equipped with a type of position service such as GPS to obtain its own position. All nodes send beacons containing the position of the sending node periodically, so each node maintains a table that includes all its neighbors' unique IDs and geographic positions. Under GPSR, when a node wants to send a packet to a destination, the node consults its "neighbor table" and forwards the packet to the neighbor that is geographically closest to the destination node. The neighbor receiving the packet also applies the same solution. Finally, the packet arrives at the destination. This is called "most forward within transmission range" (MFR) [97].

Unfortunately, MFR may fail to find a route between source and destination even though one path exists. Since a node perhaps does not know about any nodes geographically closer than itself to the destination node, it can not forward the packet. We call this situation a dead end. GPSR recovers from the dead end through a perimeter routing strategy, which takes a planar subgraph of the wireless network's connectivity graph to route around dead ends. The graph of mobile ad hoc networks is generally not planar, for planar graphs are graphs with no intersecting edges. At first GPSR planarizes the graph depending on having current position information for a node's current set of neighbors via Relative Neighborhood Graph (RNG) [100] or Gabriel Graph (GG) [28]. Then perimeter routing strategy uses the right hand rule to forward the packet on faces of the planar subgraph that is progressively closer to the destination. GPSR returns a perimeter routing to MFR when the packet reaches a node closer to the destination than at which the packet entered the dead end. Karp and Kung [47] simulate GPSR on mobile ad hoc networks and show that it is a scalable scheme for packet forwarding and delivers more packets successfully with lower routing overhead.

The important characteristic is that GPSR allows all packets destined for an arbitrary location (perhaps unoccupied by a node) to be routed consistently to the same node in the vicinity of that location. Our proposed scheme takes this strength to route location information update and location information query to the same node which acts as a location server, as long as a hash function maps IDs into the same location.

3.2.2 Grid Location Service (GLS)

The Grid Location Service (GLS), proposed in [58], is part of the Grid project. GLS relies on a grid-based geographic hierarchy overlaying the ad-hoc network. In this hierarchy, a large square is divided into exactly four smaller square areas recursively. The smallest square is referred to as an order-1 square, which can be covered by a node's transmission range. The four order-(n-1) squares sharing the same order-n square as the parent square are sibling squares. For each node, one node from each sibling square of the node's square at each hierarchy level is selected as its location server. Each node periodically broadcasts a list of all neighbors it can reach in one hop using a beacon. This ensures that each node knows the location of all nodes in its order-2 square. Figure 3.2 shows an example of a hierarchy. GLS assigns a unique, random ID to a node by applying a strong hash function to the node's unique name, which could be a host name, IP address, or MAC address.

Let us consider an arbitrary node υ. The following rules are used to determine where to store and update υ's position information:

1) The set of nodes selected as location servers of υ are based on the relation of their nodes' IDs to υ's ID and their position in the grid hierarchy. Location servers of node υ have the least ID greater than υ's ID in a particular grid square.

2) The density of location servers for node υ is high in squares near υ and low in squares far from node υ.

3) Node υ updates its nearby location servers more frequently than location servers that are far away from υ.

We illustrate the mechanism of GLS by a simple example (see Figure 3.2). For node 20, the selected location servers in the three surrounding first-order squares are nodes 21, 25, and 30 since all the three nodes are in the same first-order square with node 20 itself, and have the nearest ID in each square. In the surrounding three second-order squares, again the nodes with the nearest ID are chosen to host the node's position; in the example these are nodes 23, 33, and 59. This process of selecting location servers is repeated until the area of the ad-hoc network has been covered. We can see that the density of location servers for a given node, such as node 20, decreases logarithmically with the distance from that node.

A node sends out location updates at a rate proportional to its speed and those updates are sent to distant servers less often than to local servers. A node updates its location servers in order-*i* sibling squares after each movement of $2^{i-1}d$, where d is a particular threshold distance. Since there are three sibling squares and thus three location servers at each level of the hierarchy, and the hierarchy height is $O(\log_4 N)$, the total number of location servers is $O(3\log_4 N)$.

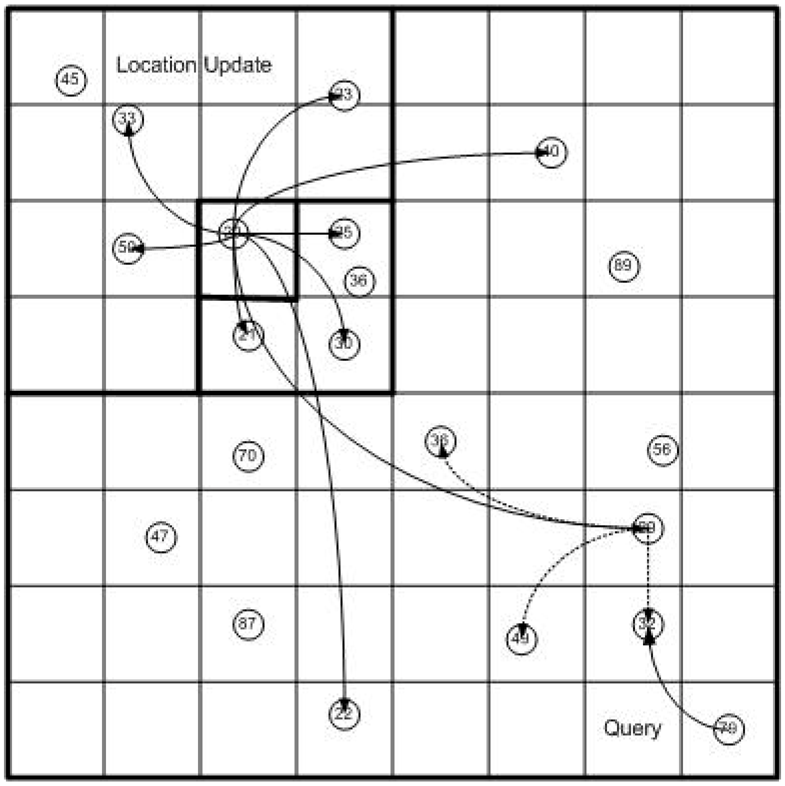


Figure 3.2 An example of GLS

Now we also use the example in Figure 3.2 to explain how a node queries the location information of an arbitrary node from that node's ID through GLS. Suppose node 79 wants to obtain the position of node 20. It should therefore locate a 'nearby' node that hosts the position of node 20. In the example this is node 29. However, node 79 does not know that node 29 holds the required position; it must discover this information. There exists a trail of descending node IDs from each of the squares of all orders to the correct position server. Position queries for a node can now be directed to the node with the nearest ID the querying node knows of. In our example, this would be node 32. This guarantees that no nodes in that square have IDs between 20 and 32. The node with the nearest ID does not necessarily know the node sought, but it will know a node with a nearer node ID. Node 32 also tries to find a nearest ID to destination node 20 that it knows. This is node 29. The process continues until a node that has the position information is reached. Note that a query or a query reply stays inside the smallest square containing the source and the destination.

However, since the hashing is performed individually for each sibling square at each level of the hierarchy, every time a node crosses a grid boundary, it has to update some of its location servers accordingly. In the worst case, when a node crosses a boundary between two highest level grids, it has to select and update all of its O ($logN$) location servers. Furthermore, in order to alleviate the location query failures due to the mobility of location servers, a node has to broadcast "forwarding pointers", which indicate the destination order-1 square node will move, to all nodes in its previous order-1 square [58]. When a node enters a new order-1 square, it collects the forwarding pointers by piggy-backing them on beacon packets. As a result, the size of the beacon packets tends to grow.

Li *et al.* showed that GLS had significantly better scaling properties than previous routing protocols for mobile ad hoc networks in [58]. However we may notice that when nodes move arbitrarily, GLS becomes less efficient. Two main disadvantages cause the heavy signaling overhead. Firstly, a node has to possibly scan the entire region to obtain the one with the “closest” ID in order to select it as a location server for holding position information of that node. Then with nodes’ entries or departures in a particular square, the scheme needs to check the entire particular square periodically, and perhaps in some situation it needs to change nodes for holding position information of some other nodes.

3.2.3 Distributed Location Management (DLM)

Similar to GLS, in Distributed Location Management (DLM) [108], which is proposed by Yuan *et al.*, the entire network deployment is partitioned into a hierarchical grid. As shown in Figure 3.3, the whole network is referred to 0-order region. Each k-order region includes $a \times b$ (k+1)-order regions until the smallest square can be covered by the transmission area of a node. Each node in DLM has a changeable address consisting of a list of strings, (i, j), which represents the relative position of this k-order region with respect to this (k+1)-order region with the upper left corner as the origin point, and i identifies the row; j identifies the column.

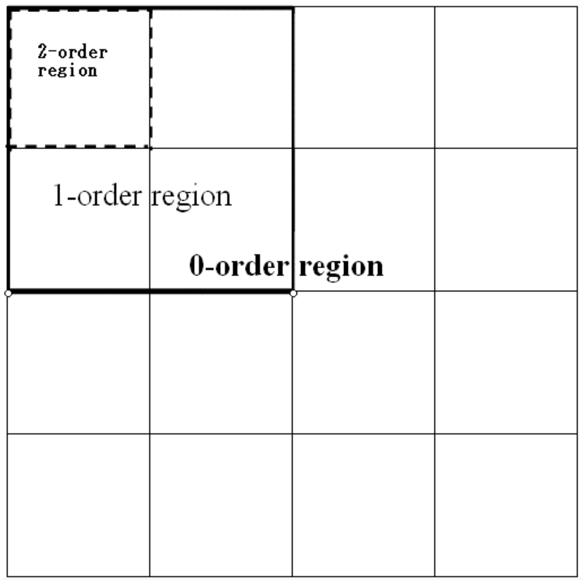


Figure 3.3. Network Partition Representation

DLM assumes a uniform distribution of the location servers. The server density is a parameter that may be adapted to better suit the characteristics of the network. The density of location servers for a specific node is one server in each of the *m*-order regions, where *m* is the parameter to adjust the density. To the contrary of GLS, location servers in DLM are not directly nodes, but regions in the grid. Nodes that happen to be located in these regions offer the location service. This solution increases DLM's robustness to mobility. The selection mechanism for the predetermined regions is carried out through a hash function that maps node identifiers to region addresses.

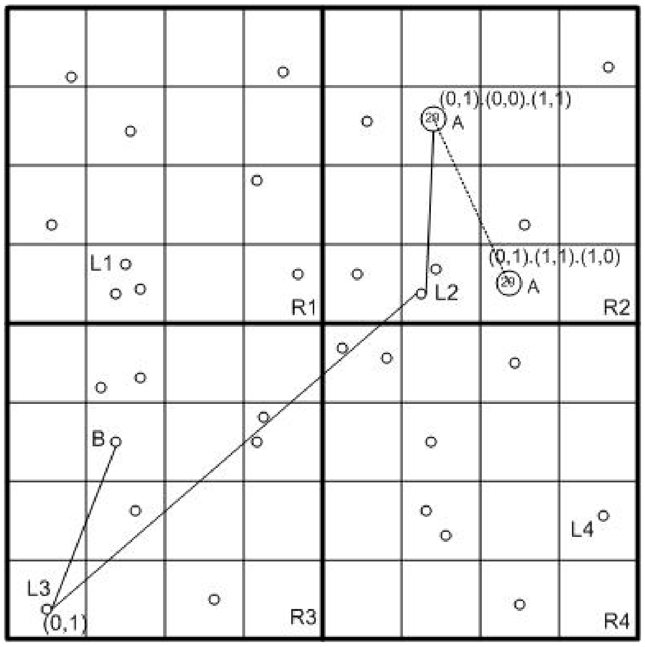


Figure 3.4. An example of DLM.

We illustrate the selection mechanism for the predetermined regions via a concrete example in Figure 3.4. In this figure, the 0-order region is partitioned into four 1-order regions, referred to as R_1, R_2, R_3 and R_4, respectively. If we set the density of location servers as 1, $m = 1$, which means node A has one server in each of the 1-order regions, where $a = b = 4$, and ID(A) = 29. DLM uses a hash function to map A's ID to a particular minimum partition (i,j). The hash function is represented as i = [ID(A) mod (($a \times b$)] / a and j = [ID(A) % (($a \times b$)] mod a. If there are not any nodes in this particular minimum partition, it checks another backup minimum partition according to the following rules: (1) if $j > 0$, the backup partition's relative address is (i, j-1); (2) if $j = 0$ and $i > 0$, its relative address is (i-1, b-1); (1) if $j = 0$ and $i = 0$, its relative address is (a-1, b-1). Since 29 mod ($a \times b$) = 13, the relative address of the region where L_1, L_2 reside with respect to its 1-order region R_1, R_2 respectively, is (3, 1), where 3 = 13 / 4 and 1 = 13 mod 3. L_2 is located in the region whose relative address is (3, 0), since the region whose relative address is (3, 1) with respect to R_3 is a "void" region. Similarly, L_3 is located in the region (2, 3), since region (3,1) and (3,0) are both "void".

DLM introduces a hierarchical addressing model and distinguishes between a full length address policy and a partial length address policy. Under the full length address policy, location servers store the precise position of nodes. When nodes change regions due to mobility, it is necessary to update all location servers. Under the partial length address policy, the accuracy of node location information stored at the location servers increases along with the proximity of the location servers to the nodes. To the contrary of the full length address policy, several queries are necessary to locate a node. Nevertheless, the partial addressing scheme offers overall increased performance, since it reduces the scope of frequency of location server updates due to node mobility. Indeed, only the location servers affected by the distance traveled by the nodes need to be updated.

Figure 3.4 depicts how to query a node's address from the location services. The smallest common region of two nodes, υ and ω, is the smallest region that is able to contain both nodes. For example, in Figure 3.4, the smallest region of nodes, A and L_2, is the 1-order region R_2, and the smallest common region of A and L_1 is the entire 0-order region. For node A, and one of its location servers, L_i, we assume their smallest common region is an n-order region. If $n = m$, then L_i stores A's full address; Otherwise, if $n < m$, L_i only stores A's $(n + 1)$-level partial node address to identify what $(n + 1)$-order region node A, lies in. So in this example, L_2 holds A's full address, (0, 1).(0, 0).(1, 1), because L_2 and A have 1-order smallest common region. L_1, L_3 and L_4 store A's partial address (0, 1). When node A, moves into a new position (0, 1).(1, 1).(1, 0), A needs to update its new position information to location server L_2, and does not need to update other three location servers. Suppose node *B* wants to obtain the position of node *A*. It should therefore locate a 'nearby' location server L_3, that hosts the position of node *A*, in this example. However node *B* does not know that node L_3 is *A*'s location server, it must discover this information. Thus, node *B* uses parameter *m* and hash function to find the correct location server. Node *B* knows that *A* resides in 1-order region (0,1), then *B* maps *A*'s ID into the location server L_2 in (0, 1) 1-order region. L_2 stores the full address of *A*, now *B* has the position address of node *A*.

To allow query resolution despite mobility of location servers themselves, DLM provides two solutions for location maintenance. (1) A location server is responsible to find new location servers for all nodes on which it hosts the position information, and handoff the position information of these nodes to new servers when it moves away from the previous region. (2) A location server discards the position information it hosts when it moves away from the region. Each time when a node updates its location, it may check whether the original location server is still in the region. If it is, this node will update the server with its new location. Otherwise, a node will choose a new server using the selection mechanism.

However, there exist some disadvantages in the DLM scheme. Firstly, location servers distribute evenly throughout the whole network; thus, the distance traversed by a location query is independent of data packet path length. At the same time, if there exist some void minimum partitions, DLM requires several instances of location query forwarding and hash function computation. It costs limited network bandwidth.

3.3 Description of EnSLS

We introduce an Enhanced Scalable Location management Service (EnSLS) [64] that is designed to address the problems of GLS and DLM. EnSLS is based on the idea that a node maintains its current location in a number of location servers distributed throughout the network. These location servers are not specially designated; each node acts as a location server on behalf of some other nodes. EnSLS maintains the property: long distance queries are proportionally penalized. The location servers for a node are relatively dense near the node but sparse from node. EnSLS maps a node's unique ID to the central location of a specific minimum region via a consistent hashing. It takes the GPSR [47] as the underlying packet forwarding protocol, which allows that all packets destined for an arbitrary location (possibly unoccupied by a node) to be forwarded consistently to the same node in the neighborhood of that location [85]. Thus, EnSLS ensures that the node whose location is currently closest to the hashed location as the node's location server.

We model a mobile ad-hoc network as a set of mobile nodes deployed in a predetermined rectangular region of 2D dimension. Each node has a unique ID such as an IP address, MAC address or Internet host name. In our model we assume that mobile nodes do not move out of the deployed area. This is guaranteed in some realistic situations. For example, entire combat field forms the geographic extent of the ad-hoc network. However, there exist situations where the ad-hoc network may move into new geographic areas such that the area boundaries need to be redefined and updates need to inform all the nodes. We expect such management function to be performed by a management layer on top of the routing layer. We assume that every node has a table containing all its neighbors' unique IDs and geographic positions, which can be collected by periodic beacon messages.

3.3.1 Hierarchical Network Model

As DLM, our scheme divides the deployed network into a hierarchy of grids with squares of increasing size as shown in Figure 3.5. The whole ad-hoc network, which is also referred to as the order-0 region, is partitioned into 2×2 order-1 squares, and each order-1 region is also partitioned into 2×2 order-2 squares, and so on. Thus, every order-(n-1) square includes four order-n squares. When the size of smallest region is fully covered by a node's transmission range, we stop this partitioning. We denote the smallest region as a unit partition, and the order number of unit partition as N_{max}. In Figure 3.5, 3-order region R_5 is a unit partition, and N_{max} equals 3.

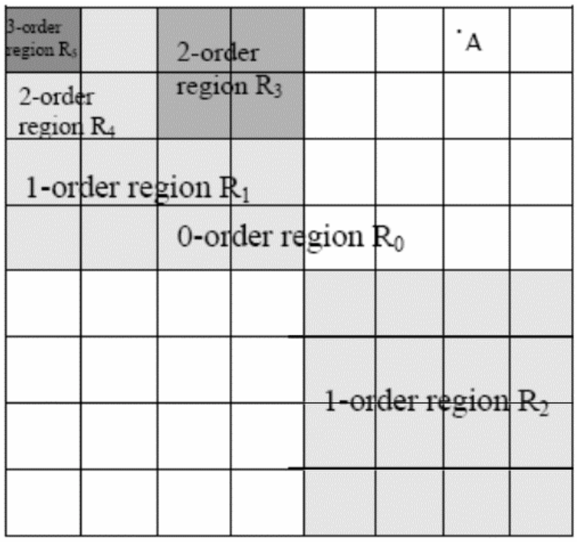


Figure 3.5. Global partitioning of network.

Two nodes residing in the same minimum partition are one-hop neighbor, and they know each other's information through beacons. We denote the relative address of a minimum partition with respect to N_{max}-1, …, 0-order region *R* as (i, j), which *i* identifiers the row, and *j* identifies the column with the upper left corner of *R* as the origin point. As shown in Figure 3.5, the relative address R_5 with respect to 1-order region R_1 is (0, 0). The relative address of the minimum partition *A* resides in is (0, 6) with respect to 0-order region R_0. When the deployment area and partitioning scheme are determined by upper network layer protocols, a mobile node easily converts its coordinate location such as longitude and latitude provided by GPS to its node address.

3.3.2 Selection of Location Servers

In our proposed scheme, a specific node *A* needs to select some nodes as its location servers and update them with *A*'s changing locations, using its ID and the predetermined grid hierarchy. Thus, if any other node *B* wants to contact *A*, *B* needs to know where *A*'s location servers are from *A*'s ID, the only information. There is no static relation between *A*'s ID and *A*'s location because of *A*'s mobility. *A*'s location servers have little knowledge beyond *A*'s ID, so it is important how we determine which nodes are the location servers of node *A*. In GLS, *A*'s strategy is to recruit nodes with IDs the least "closest" to its own ID to serve as its location servers. This appears to assume that a node can scan the entire square (of arbitrary size) and choose the appropriate nodes as its servers. While in our proposed scheme, we use a consistent hash function to map *A*'s ID to central coordinates of several minimum partition regions, then choose the nodes geographically closest to the particular locations as location servers. Since the underlying geographic routing protocol is GPSR, we ensure that location updates and location queries meet at the same nodes, location servers, in the vicinity of those locations.

We look at the whole grid decomposition as a tree, as in GLS, and a node selects location servers in each sibling of a square that contains the node. A node chooses three location servers for each level of the grid hierarchy. Here a hash function is used to map *A*'s ID to the relative address of a minimum partition with respect to N_{max}-1,…1–order region *R*. There is a relationship between node ID and a specific minimum partition.

f(Node ID) → relative address of a minimum partition with respect to region R

f is a many-to-one mapping that is static and known to all nodes of the ad hoc network. We represent function *f* in Table 3.1.

For a specific large region *R*, node *A* sends update packets with its current position information to the central coordinates of minimum partition region with the relative address (i, j). Location servers in the specific region *R* are those nodes geographically nearest the destination coordinates. We denote these kinds of coordinates as home coordinates. If a node lies in the exact home coordinate, then it is declared as a location server of node *A*. Otherwise the update packet is addressed to a specific location, home coordinate, so it is treated by GPSR as a packet bound for a disconnected destination; no receiver ever sees the packet addressed to its own identifier. Under GPSR packets forwarding, the packet enters perimeter mode at the node which is selected a location server, as no neighbor of the location server can be closer to the destination. The packet then traverses the entire perimeter that encloses the destination, before returning to the location server. We name this perimeter the home perimeter. All nodes in this perimeter store the replicated location information of node *A* beside the specific location server. The selection algorithm for coordinates of location servers is formally presented in Table 3.1.

```
SelectLoactionServer (ID(A))
{For each partition R having the ($N_{max}$ - 1) smallest common region with node A
    {
        R is minimum partition itself.
        Let (x, y) be the coordinate of the center of R
        //We define (x, y) as the home coordinate of node A.
            Any node L geographically nearest to the home coordinate (x, y) is one of location servers A
    }
  //C——the order number of smallest common region with node A
  For C = ($N_{max}$ - 2) downto 0 do
    {k = $N_{max}$ – C-1
     a = $2^k$;
    For each partition R having the C-order smallest common region with node A
        {i = [ID(A) % $a^2$] / a
         j = [ID(A) % $a^2$ ] % a
         consider the minimum partition P whose relative address with respect to R is (i, j)
         Let (x, y) be the coordinate of the center of P
         //We define (x, y) as the home coordinate of node A.
         B sends update packet to (x, y) with A's current location with GPSR
         If L lies in coordinate (x, y) then
```

Table 3.1. Selection algorithm of location servers of a specific node A.

Figure 3.6 illustrates the selection algorithm by an example. Firstly we consider three regions R_1, R_2 and R_3 that have the 2-order smallest common region with A. We can see that R_1, R_2 and R_3 are minimum partition regions themselves. Minimum partition regions can be covered by the transmission area of a mobile node. Let L_1, L_2 and L_3 be the centers of these three regions respectively, so node A sends update packets with its current position information to L_1, L_2 and L_3. S_1, S_2 and S_3 are selected as location servers of node A and store A's position, since they are geographically nearest to L_1, L_2 and L_3 respectively. Then we look at regions R_4, R_5 and R_6 which have the 1-order smallest common region with A. Since $k = N_{max} - C - 1 = 1$, $a = 2^k = 2$, and ID (A) = 27, i = [ID(A) mod a^2] / a = 1, j = [ID(A) mod a^2] mod a =1, we choose the centers of minimum partition regions with relative addresses (1, 1) with respect to R_4, R_5 and R_6 as home coordinates (L_4, L_5 and L_6), and S_4, S_5 and S_6 as location servers of node A in 2-order regions. As far as 1-order regions, R_7, R_8 and R_9, they have the 0-order smallest common region with node A. According to our algorithm, the home coordinates are the center of minimum partition regions having relative addresses (2, 3) with respect to R_7, R_8 and R_9, so S_7, S_8 and S_9 are selected as location servers of node A. Notice that S_8 is closer to L_8 home coordinate than node X and Y to L_8. So S_8 acts as location server of A's, X and Y do not.

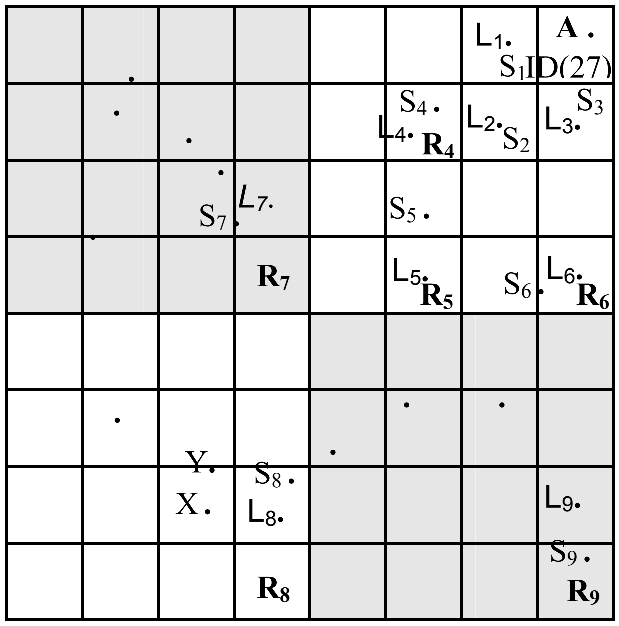


Figure 3.6. Location server distribution of node A

3.3.3 Maintenance of Location

In ad hoc mobile networks, mobility increases the maintenance overhead of location information of every node in two ways. First, each of the mobile nodes needs to update its location servers with its new location when it moves around. Second, we need to consider the mobility of location servers themselves. If *S* is one of location servers of node *A*. *S* is no longer a location server of node *A* when *S* moves away from original position and is no longer the nearest one from the home coordinate. In the first case, we require that a node sends out updates information at a rate proportional to its speed. Thus nodes avoid generating excessive amounts of update traffic by linking their location update rates to their distance traveled. A node updates its *n*-order location servers every time it moves out of its (*n*+1) regions. For example if we estimate every "*t*" times a node moves out of its minimum partition region, the node updates its N_{max}-order location servers every "*t*" intervals; the node updates its (N_{max}-1)-order location servers every 2*t* intervals. In general, a node updates its (N_{max}-k)-order location servers every $2^k t$ intervals. Therefore updates are sent to distance servers less often than to local servers, even when stationary nodes also send location updates at a low rate. Each location update packet includes a timeout threshold corresponding to the specific periodic update interval, and this allows that the location servers can replace the position entries shortly after a new entry is expected. The location update packet also maintains the time at which the location update packet is generated so that the freshness of location information obtained from different nodes for the same destination can be compared. Every node receiving these kinds of update packets stores the ID of destination node and its corresponding geographic position in its cache.

On the other hand, even if a node remains stationary, as location servers of the node move or fail or power off, we need to choose new location servers to store the location information of the node. Due to node mobility, there might also be a situation as follows, during location server selection, *S* is the nearest one to a home coordinate and selected as a location server of node *A*, and then a node has moved to the position that is closer to this home coordinate than *S*. In this case, if location query for *A* occurs, perhaps this query cannot obtain the correct response of *A*'s location. Let us think about how to solve the mobility of location servers. Recall that GPSR routes all update packets on a tour of the perimeter that encloses a destination location. Our scheme stores a copy of *A*'s location at each node on the home perimeter. We call these nodes in the perimeter replica nodes. A node becomes location server when the update packet arrives after completing its tour of the perimeter. Location servers originate refresh packets addressed to the home coordinates, generated by the hash function periodically. When a fresh packet arrives at a node, two scenarios occur. If the receiver is closer to the home coordinate than the originator, then the receiver consumes the refresh packet and initiates its own; if the receiver is not closer than the originator, then it forwards the refresh packet in perimeter mode. When a node returns to its originator, and that node perhaps was not previously the location server for node *A*, it consumes the refresh packet and becomes the location server for node *A*. That is, the new location server of *A* sets its own refresh timer and subsequently originates refreshes for *A*. Thus it ensures that the node closest to a node's hash location will become the location server for that node and store that node's location after topological changes. In some situations, the location server for node *A* fails, and its replica nodes will note the absence of refreshes for node *A* from its location server and step forward to initiate refreshes. A replica node may or may not be the new location server itself; the GPSR routing procedure causes the refresh to reach the new location server.

Figures 3.7 to 3.9 show an example of the operation of the location update. Here, node *S*'s ID hashes to the home coordinate *L*. After *S* sends update packet with its current location to *L*, node *A* becomes location server of node *S*, and periodically sends a refresh to *L* containing *S*'s location. Figure 3.7 shows the perimeter enclosing *L* after this refresh has returned to *A*. Suppose node *A* fails. After some time elapses during which node *F* receives no refreshes from node *A*, node *F* sends a refresh to *L* containing *S*'s location, as shown in Figure 3.8. This refresh is delivered to node *D*, which becomes the new location server for node *S*. Figure 3.9 shows the network after *D* has sent a refresh that has returned to it, and replicas it has recruited along the new perimeter of *L*.

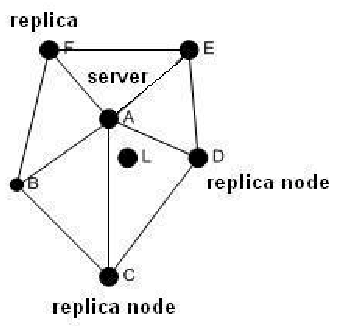


Figure 3.7: L is hash addressed home coordinates. A is location server, and B, C, D, E and F are replica nodes

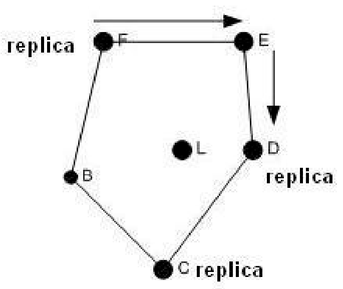


Figure 3.8: A fails, F initiates a refresh packet to L

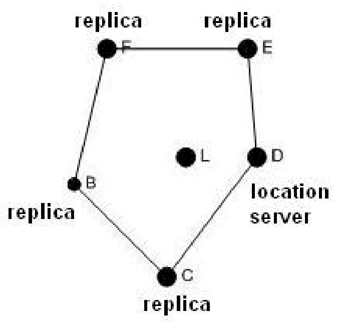


Figure 3.9: D becomes the new location server. B, C E and F are replica nodes

3.3.4 Query of Location

When a node *S* wants to send a data packet for destination *D*, as we said before that each node has a table for its neighboring nodes' position information and a cache for storing some nodes' ID and their corresponding position information, *S* first checks its location cache and table to find *D*'s geographic position. If it finds an entry for *D*'s location, it sends the data packet to *D*'s recorded location using GPSR geographic forwarding. Otherwise, *S* has to initiate a location query packet for *D*'s position. *S* first computes the relative addresses of location servers for node *D* in every level of the predetermined network hierarchy. We represent these relative addresses of minimum partitions as (i_1, j_1), (i_2, j_2), …, and (i_{Nmax}, j_{Nmax}). Then *S* multicasts location query packets to the home coordinates of these N_{max} specific minimum partitions, so S sends at most N_{max} messages. At least one of location servers receives this location query and sends back a reply message with *D*'s location to *S*. Thus, *S* has the location information of node *D* and delivers a data packet to *D*.

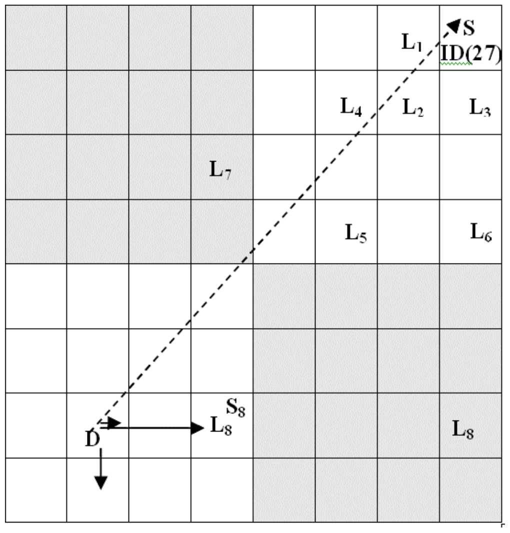


Figure 3.10. Location query example.

Figure 3.10 gives the distribution about hashed locations of the node *D*'s location servers, i.e. L_1, L_2, … , L_9. The nodes that are geographically closest to the hashed locations host the position information of *D*. If *S* wants to contact *D*, and *S* only knows the ID of node *D* is 27, node *S* firstly computes the relative address of node *D*'s home coordinates of location servers for each level of hierarchy. Then node *S* multicasts location query packets to these home coordinates via GPSR. In the example, one of the query packets aim at the home coordinate L_8, and S_8 is the closest node to L_8, thus, it stores the location information of node *D*. Node *S* sends a query packet to node *D* after having its location information. Node *D* reply node *S* with its current location.

3.4 Optimization

In addition to the basic protocol described in the previous sections, there are several optimization techniques to improve the performance of the basic protocol.

1. Since the location updates are sent to distant location servers less often than to local servers, the location information on local servers are more accurate than

that on distant servers. When a node obtains the destination node's location from distant servers, it also send the last location query to the sibling (N_{max}-1)-order squares instead of only exact square which distant servers indicate the destination node resides in. Servers in the sibling (N_{max}-1)-order squares may have the accurate destination node's location.

2. Location replies are issued by intermediate nodes if the requested entry in the location cache is recent enough. This is a tradeoff between faster initial location discovery time and accurate location information.

CHAPTER 4

PERFORMANCE ANALYSIS OF POSITION-BASED ROUTING SCHEMES

Scalability of a location service is best characterized by the control message overhead required to route data packets between nodes. Woo and Singh [103] were the first to analyze the scalability of a location management scheme theoretically. Das, Puch and Hu [23] presented a quantitative model for measuring location service overhead of three representative rendezvous-based location services. In this chapter, we employ a theoretical model to analyze the scalabilities and make a comparison between GLS scheme and our EnSLS scheme. We show our EnSLS scheme has better performance than GLS. For any location servers, message overhead is associated with these three important operations: the update of location information, maintenance of location information and location information query. Handling mobility requires an obvious tradeoff between the costs involved in querying and in updating the location information - reducing the cost of one of them may lead to an increase in the other.

This chapter is organized as follows. In Section 4.1 we provide a glossary of notation used in the remainder of this chapter. Section 4.2 presents our analysis and comparison for the location update costs of GLS and EnSLS. Section 4.3 illustrates our analysis for location maintenance costs. Location query costs of two location services are described in Section 4.4. Section 4.5 puts all the pieces together and provides the total costs of GLS and EnSLS.

4.1 Notation and Assumption

It is important to mention that our analysis ignores the savings obtained by the use of caches in the system. Thus, our analysis model actually provides an upperbound for the total cost. Recall from our description about the two location services, GLS and EnSLS, the whole network area is partitioned into a hierarchy of squares with different sizes. To aid our analysis, we adopt the same partition methods in the two location services. The minimum partition is 1-order square, which can be covered by the transmission range of a mobile node. Four 1-order square forms a 2-order square. We denote the height of network hierarchy as h, thus, the whole network area is a h-order square. If the side length of 1-order square is R, the side of i-order square will be $2^{i-1}R$ and the side length of the whole area is $2^{h-1}R$. We use the following notation in this section:

N	number of nodes,
r_t	transmission radius,
R	the side length of 1-order square
h	the hierarchy of the whole network area
ρ	number of square crossings/s/node
v	speed of node (m/s)
A	area of the whole network
λ	the density of nodes in the network

As the number of nodes in the network grows, we expect the cost of routing to also increase since there are more requirements for location update, location maintenance and location query. Therefore, we should model increasing network sizes properly in order that we can compare the performance of different location services fairly regardless the number of nodes in the network. In our model, the network size must also increase with an increase in the number of nodes N so that the average density of nodes λ is kept constant. This model better reflects our notion of scale and provides a more meaningful measure of a protocol's performance. Thus, we can derive the relationship between the hierarchy of the network h and the number of nodes N as follows, since the h-order square is the whole network, which can be covered by the transmission areas of all the nodes in the network.

$$(2^{h-1})^2 R^2 = A \qquad (1)$$

$$\frac{\pi r_t^2 (N-1)}{\lambda} = A \qquad (2)$$

From formulas (1) and (2), we obtain

$$h = \log \sqrt{\frac{\pi r_t^2 (N-1)}{\lambda R^2}} + 1 \qquad (3)$$

4.2 Location Update Cost

Location update cost covers all the signaling messages nodes send to their location servers whenever they move to a new location. We analyze and compare the location update costs of GLS and EnSLS schemes. The results show that the average location update cost generated by GLS is proportional to the product v and $N^{\frac{1}{2}}$, our EnSLS scheme's average location cost is proportional to the product of v and *logN*. Hence, our EnSLS seems likely to more scalable in practice.

4.2.1. Location Update Cost C_{u_GLS} Per Node Per Second in GLS

In GLS, the hashing is performed individually for each sibling square at each level of the hierarchy. Each node sends three updates to the three location servers in 1-

order sibling squares when it crosses the 1-order square boundary. Three updates are sent to three location servers in 2-order sibling squares when the node crosses the 2-order square boundary, and so on. When crossing the (*h*-1)-order square boundary three updates are sent to the three location servers in (*h*-1)-order squares. Let the lowest-order sibling squares containing the destination and the location server be (*k*+1)-order squares *X* and *Y*, respectively. The cost of performing one update is $u_{k\_GLS} + s_{k\_GLS}$, where u_{k_GLS} is the cost of forwarding the update packet in a straight line towards *Y* while inside *X* and s_{k_GLS} is the cost of sending the update to the location server with the least close ID to the destination via scanning *k*-order square. If the node crosses the 1-order region boundaries at a rate of ρ per second, we assume it crosses *i*-order region boundaries at a rate of $\frac{\rho}{2^{i-1}}$ per second for simplification. The cost of update in GLS can be written as

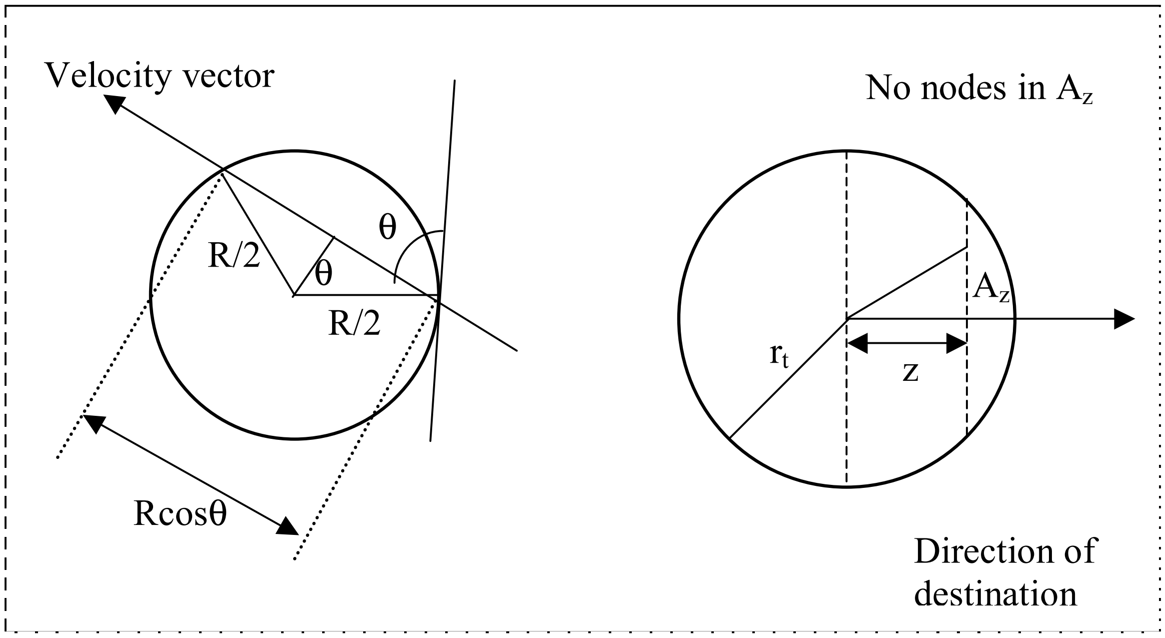
$$c_{u\_GLS} = 3\rho(u_{1\_GLS} + s_{1\_GLS}) + 3\frac{\rho}{2}(u_{2\_GLS} + s_{2\_GLS}) + \ldots + 3\frac{\rho}{2^{h-2}}(u_{h-1\_GLS} + s_{h-1\_GLS}) \qquad (4)$$

We can estimate ρ quite easily by assuming a circular region of radius $\frac{R}{2}$ meters [103]. As illustrated in Figure 4.1(a), once the node enters a region, it travels for some time within the region before exiting. The distance the node travels while in the region can be seen to be *Rcos(θ)* where *θ* is the angle the velocity vector makes with the tangent at the point of entry into the region. Thus, the average distance traveled by a node within a region is

$$\frac{2}{\pi}\int_0^{\pi/2} R\cos(\theta)d\theta = \frac{2R}{\pi}$$

Therefore, we get

$$\rho = \frac{v}{2R/\pi} = \frac{\pi v}{2R} \qquad (5)$$



(a) Distance traveled by a node within a region (b) Distance traveled in one transmission

Figure 4.1 Estimate for ρ and z

We can estimate u_{k_GLS}, the number of hops between the node generating the location update message and the node which is the first node within the *k*-order sibling square to receive the location update message, as

$$u_{k\_GLS} = \frac{d_{k\_GLS}}{z}$$

where d_{k_GLS} is the physical distance between the node generating the location update message and the node which is the first node within the *k*-order sibling square to receive the location update. z denotes the average forward progress made towards the destination in the course of transmission. Recall that GLS uses MFR as the packet forwarding protocol.

Figure 4.1(b) shows the actual physical distance (*z*) covered by one transmission [103]. The circle represents the transmission range of the node located at the center of the circle. The arrow indicates the direction in which the destination is located. In this diagram, the node closest to the destination is distance *z* away from the center and the region A_Z is empty. To determine *z*, let us assume that nodes are placed in the network according to a two dimension uniform distribution. The probability function of *z* is shown as follows.

$$F_z(z) = 1 - \frac{2A_z}{\pi r_t^2}$$

where the excluded region A_Z is

$$A_z = r_t^2 (\cos^{-1}(\frac{z}{r_t}) - \frac{z}{r_t}\sqrt{1-(\frac{z}{r_t})^2})$$

Then, we can then write the pdf (probability density function) for z as

$$f_z(z) = \frac{2}{\pi r_t}\left[\frac{1+\left(\frac{z}{r_t}\right)^2}{\sqrt{1-\left(\frac{z}{r_t}\right)^2}} + \sqrt{1-\left(\frac{z}{r_t}\right)^2}\right]$$

Thus, the average progress made towards the destination in one hop is

$$\bar{z} = \int_0^{r_t} z f_z dz$$

Numerical integration techniques can be used to solve it. $\bar{z}$ is a constant, which is proportional to the transmission radius.

To determine u_{k_GLS} we also need to find the average distance d_{k_GLS}. Since the network area is shaped like a torus, we can simplify d_{k_GLS} as follows.

$$d_{k\_GLS} = 2^{k-1}(2+\sqrt{2})R \qquad (6)$$

Finally, let us estimate s_{k_GLS}, the cost of sending the update to the location server with the least closet ID to the destination via scanning k-order square. The approximate cost of scanning k-order square is the number of transmissions needed to cover the region with the circles of radius r_t. Thus,

$$s_{k\_GLS} = \frac{(2^{k-2}R)^2}{\pi r_t^2} \qquad (7)$$

Theorem 4.1. *The average location update cost per node per second c_{u_GLS} in GLS location service is* $\propto v\sqrt{N}$.

Proof. The average location update cost per node per second is

$$\overline{c_{u\_GLS}} = 3\bar{\rho}(\overline{u_{1\_GLS}} + \overline{s_{1\_GLS}}) + 3\frac{\bar{\rho}}{2}(\overline{u_{2\_GLS}} + \overline{s_{2\_GLS}}) + \ldots + 3\frac{\bar{\rho}}{2^{h-2}}(\overline{u_{h-1\_GLS}} + \overline{s_{h-1\_GLS}})$$

Substituting values for $\bar{\rho}$, $\overline{u_{k\_GLS}}$, and $\overline{s_{k\_GLS}}$, we get

$$\overline{c_{u\_GLS}} = \frac{3(2+\sqrt{2})(h-1)\pi \upsilon}{2z} + \frac{3\upsilon R}{2r_t^2}(2^{h-2} - \frac{1}{2})$$

Substituting h with formula (3), then

$$\overline{c_{u\_GLS}} = \frac{(2+\sqrt{2})\pi\upsilon\log\sqrt{\frac{\pi r_t^2(N-1)}{\lambda R^2}}}{2z} + \frac{\upsilon R}{2r_t^2}\left(\frac{1}{4}\sqrt{\frac{\pi r_t^2(N-1)}{\lambda R^2}} - \frac{1}{2}\right)$$

Simplifying

$$\overline{c_{u\_GLS}} \propto \frac{\pi\upsilon\log\sqrt{\pi(N-1)\lambda}}{z} + \frac{\upsilon}{2r_t}\left(\frac{1}{4}\sqrt{\pi(N-1)\lambda} - \frac{1}{2}\right)$$

where z is a constant which is proportional to the transmission radius, and λ is kept constant despite of the increasing network size. Therefore, eliminating terms we get

$c_{u\_GLS} \propto v\sqrt{N}$. □

4.2.2. Location Update Cost C_{u_Ensls} Per Node Per Second in Ensls

EnSLS maintains the property of GLS, distance location servers are updated less often than local servers. When a node moves into a new 1-order square, it sends update to three 1-order sibling squares. Updates are sent to the three hashed locations in three 2-order sibling squares directly via GPSR when a node moves away from its original 2-order squares, and so on. When crossing the (h-1)-order square boundary three updates are sent to the three location servers in the neighboring (h-1)-order squares. If a node crosses the 1-order region boundaries at a rate of ρ per second, it crosses i-order region boundaries at a rate of $\frac{\rho}{2^{i-1}}$ per second for simplification. The cost of update in EnSLS can be written as

$$c_{u\_EnSLS} = \rho u_{1\_EnSLS} + \frac{\rho}{2}u_{2\_EnSLS} + \frac{\rho}{2^2}u_{3\_EnSLS} + \ldots + \frac{\rho}{2^{h-2}}u_{h-1\_EnSLS} \qquad (8)$$

where u_{k_EnSLS} is the cost of forwarding the update packet in a straight line towards the location servers which are geographically nearest to the hashed locations in the three sibling *k*-order squares.

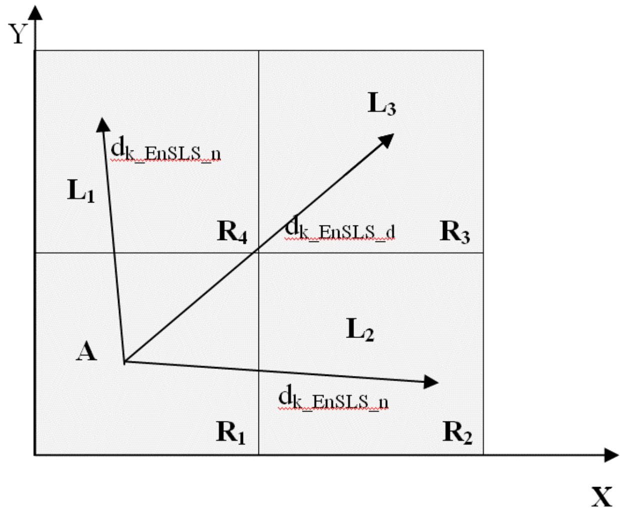


Figure 4.2. Estimate for the cost of updates to three k-order squares.

In order to estimate u_{k_EnSLS}, consider Figure 4.2 that consists of four *k*-order squares, R_1, R_2, R_3, and R_4. Let us assume destination node *A* is located in R_1, and then u_{k_EnSLS} actually includes three components which are the average hops traveled by update messages from destination *A* to the location servers in R_2, R_3, and R_4 respectively. Since the positions of destination node *A* and its location servers are random in the whole network, we can estimate u_{k_EnSLS} as

$$u_{k\_EnSLS} = \frac{2d_{k\_EnSLS\_n} + d_{k\_EnSLS\_d}}{z}$$

where $d_{k\_EnSLS\_n}$ is the average physical distance between one random point within R_1 and one random point within R_2 (this is same with the average distance between one random point within R_1 and one random point within R_3), $d_{k\_EnSLS\_d}$ is the average physical distance between one random point within R_1 and one random point within R_4, and z denotes the average forward progress made towards the destination in the course of transmission.

In order to compute $d_{k\_EnSLS\_n}$ and $d_{k\_EnSLS\_d}$, consider Figure 4.2. The original coordinate lies in left bottom. $d_{k\_EnSLS\_n}$ equals the average distance between two random points, of which one is limited in region R_1 and the other is located in region R_2, such as node A and location server L_2. Assume the side length of the square is l, A's coordinate is (x_1, y_1), and L_2's coordinate is (x_2, y_2). Since the mobile hosts are uniformly distributed in the network, and (x_1, y_1) and (x_2, y_2) are in the restricted square area, we have the probability density functions for (x_1, y_1) and (x_2, y_2) as shown below.

$$f(x_1, y_1) = \frac{1}{l^2}$$

$$f(x_2, y_2) = \frac{1}{l^2}$$

Then, $d_{k\_EnSLS\_n}$ can be expressed as

$$\mathrm{d}_{\mathrm{k\_EnSLS\_n}} = \frac{1}{l^4}\int_0^l\int_l^{2l}\int_0^l\int_0^l \sqrt{(x_1-x_2)^2+(y_1-y_2)^2}\,dx_1dy_1dx_2dy_2$$

And $d_{k\_EnSLS\_d}$ can be expressed as

$$\mathrm{d}_{\mathrm{k\_EnSLS\_d}} = \frac{1}{l^4}\int_0^l\int_l^{2l}\int_l^{2l}\int_0^l \sqrt{(x_1-x_2)^2+(y_1-y_2)^2}\,dx_1dy_1dx_2dy_2$$

We use mathematical software Matlab to obtain the approximate values of $d_{k\_EnSLS\_n}$ and $d_{k\_EnSLS\_d}$.

$$d_{k\_EnSLS\_n} \approx 0.61788l \le \frac{\sqrt{2}}{2}l \qquad d_{k\_EnSLS\_d} \approx 1.15052l \le \frac{\sqrt{5}}{2}l$$

Theorem 4.2. *The average location update cost per node per second c_{u_EnSLS} in EnSLS location service is* $\propto v\log N$.

Proof. The average location update cost per node per second is

$$\overline{c_{u\_EnSLS}} = \overline{\rho}\,\overline{u_{1\_EnSLS}} + \frac{\overline{\rho}}{2}\overline{u_{2\_EnSLS}} + \frac{\overline{\rho}}{2^2}\overline{u_{3\_EnSLS}} + \ldots + \frac{\overline{\rho}}{2^{h-2}}\overline{u_{h-1\_EnSLS}}$$

Substituting values for $\overline{\rho}$ and $\overline{u_{k\_EnSLS}}$, we get

$$\overline{c_{u\_EnSLS}} = \frac{\pi\upsilon(2\sqrt{2}+\sqrt{5})}{4z}(h-1)$$

Substituting *h* with formula (3), then

$$\overline{c_{u\_EnSLS}} = \frac{(2\sqrt{2}+\sqrt{5})\pi\upsilon}{4z}\log\sqrt{\frac{\pi r_t^2(N-1)}{\lambda R^2}}$$

where z is a constant which is proportional to the transmission radius, and λ is kept constant despite of the increasing network size. Therefore, eliminating terms we get $c_{u\_EnSLS} \propto v\log N$. □

Theoretically, EnSLS should outperform GLS in terms of update transmissions as the network is scaled. To estimate the network size and the mobility of hosts beyond which EnSLS outperforms GLS, we plot the formulas for the average number of update transmissions per second for both protocols as a function of the network size *N* and the velocity of hosts *v,* respectively. Assuming the density of hosts λ equals 10, and the average forwarding progress *z* is 150m. Figure 4.3 give a comparison about the average number of update transmissions per second in both location services as a function of the number of hosts in the network when the velocity of hosts is a constant ($v = 15$, $v = 25$). Figure 4.4 plots the scalabilities of both location services, GLS and EnSLS, as a function of the velocities of hosts in the network when the number of hosts is a constant (*N* = 100, *N* = 1000). These plots are generated by the formulas of measuring the average location update costs of two schemes, and show that EnSLS consistently outperforms GLS for all network sizes and the variable mobility.

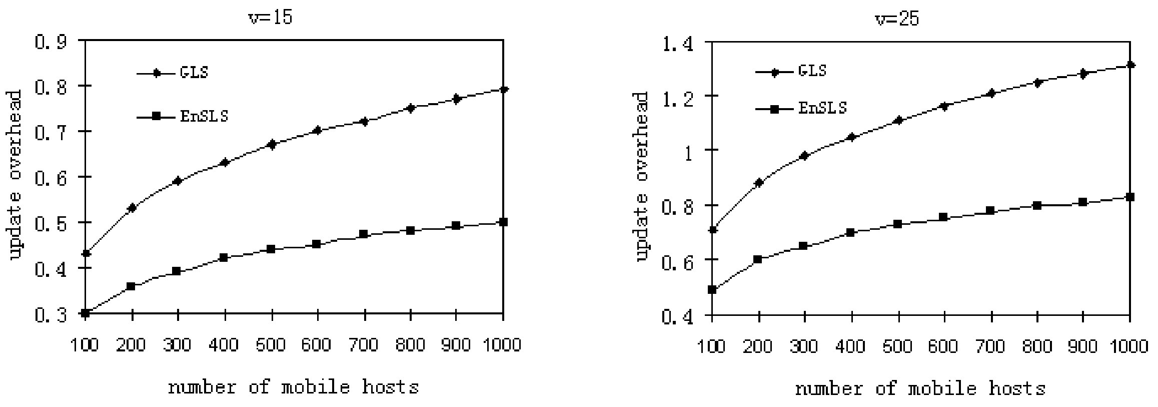


Figure 4.3. The average number of update transmissions as a function of the number of mobile hosts

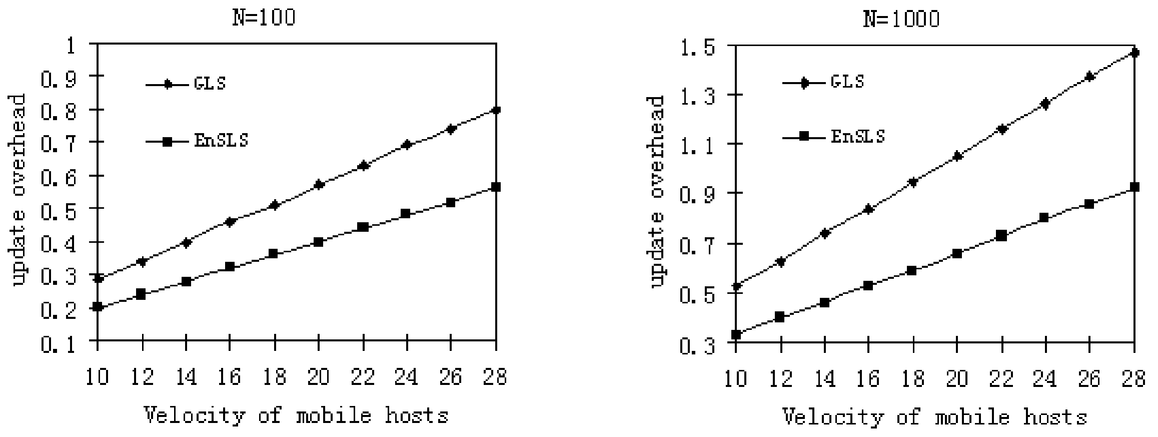


Figure 4.4. The average number of update transmissions as a function of the mobility of mobile hosts

4.3 Location Maintenance Cost

Since in location services, a node may be a location server for some perhaps not only one mobile nodes, location maintenance cost covers all the signaling messages that a node sends out in order to hand off its local location database to other nodes when it moves away from the previous location and to collect the location information it may host for other nodes when it moves into a new region. We analyze and compare the average location maintenance costs of GLS and EnSLS schemes. The results show that the average location maintenance costs generated by GLS and EnSLS are both proportional to the product of v and N. From this aspect, they have almost the same performance.

4.3.1 Location Maintaining Cost C_{m_GLS} Packets Per Second in GLS

In GLS, when a node moves from the 1-order square A to the 1-order square B, it broadcasts a "forwarding pointer" to inform all nodes in A that it has moved to square B and, simultaneously, it needs to inform nodes in B of its arrival and collect the forwarding pointers associated with square B. The message cost of performing this activity is denoted by c_{m_GLS} and in our analysis below we use the units packets/s to measure this cost. We can estimate c_{m_GLS} as follows.

$$\overline{c_{m\_GLS}} = (N\overline{\rho})(2\overline{b} + \delta) \qquad (9)$$

where $\overline{\rho} = \pi v / 2R$ is the number of 1-order boundary crossings per second per node, thus, $N\overline{\rho}$ represents the number of nodes crossing 1-order boundaries per second, $\overline{b}$ is the broadcast cost in a 1-order square, and δ is constant. We can estimate δ as the cost of collecting "forwarding pointers" associated with the new 1-order square. In the worst case, the new node would get as many as λ duplicate responses to its request for this information. In the best case, only one node would respond. Thus,

$$1 \le \delta \le \lambda$$

Using the above formulation we have

Theorem 4.3. *The cost of maintaining location servers in GLS* c_{m_GLS} *is* $\propto \upsilon N$.

Proof. We can simplify the expression for c_{m_GLS} as

$$\overline{c_{m\_GLS}} = \frac{N\pi\upsilon}{2R}\left(\frac{2R^2}{\pi r_t^2} + \delta\right).$$

Thus $c_{m\_GLS} \propto \upsilon N$. □

4.3.2 Location Maintaining Cost C_{m_Ensls} Packets Per Second in Ensls

In EnSLS, a host in its role as a location server also needs to hand off its local location database to neighboring nodes when it moves far away from the hashed location. Since a host can act as location servers for some not only one mobile hosts, it needs to transfer its stored location information for a particular host when it moves some distance away from the hashed location. We assume the local location database of a host holds ∂ copies, where ∂ is a constant, and a host hands off its location information to neighboring nodes when it move across the 1-order square boundary. Thus, the message cost of performing this handing off is denoted by c_{m_EnSLS} and in our analysis below we use the units packets/s to measure this cost. We can estimate c_{m_EnSLS} as follows.

$$\overline{c_{m\_EnSLS}} = \partial(N\overline{\rho})\overline{H}$$

where $\overline{\rho} = \pi\upsilon/2R$ is the number of 1-order boundary crossings per second per node, thus, $N\overline{\rho}$ represents the number of nodes crossing 1-order boundaries per second, $\overline{H}$ is the handing off cost to its neighboring nodes closer to the hashed location, and ∂ is constant. In the worst case, in which no node is closer to the hashed location than the current location server, the maintenance packets travel the whole circle, and then come back to the original location server. The original location server continues to maintain the location information even it moves away from its original 1-order square. Thus, the upper bound $\overline{H}$ can be estimated by the hops of traveling the circle as below.

$$\overline{H} = \frac{\pi R^2}{z}$$

Using the above formulation we have

Theorem 4.4. *The cost of maintaining location servers in EnSLS* c_{m_GLS} *is* $\propto \upsilon N$.

Proof. We can simplify the expression for c_{m_EnSLS} as

$$\overline{c_{m\_EnSLS}} = \frac{\partial N \pi^2 R \upsilon}{2z} .$$

Thus $c_{m\_EnSLS} \propto \upsilon N$. □

4.4 Location Query Cost

Location query cost covers all the signaling messages sent for locating a mobile node in the network. We analyze and compare the average location query costs of GLS and EnSLS schemes. The results show that the average location query costs generated by GLS and EnSLS are both proportional to $N^{\frac{1}{2}}$, but the EnSLS's scheme average location query cost has a less constant than that of GLS.

4.4.1 Cost of Locating A Node C_{q_GLS} Packets in GLS

When a node wants to send a data packet to some destination, it needs to find the current location of the destination before forwarding data to it. In our analysis, we assume that the location information is not cached, and therefore, the source node needs to contact the destination node's location server to find the destination's location. This cost is

$$c_{q\_GLS} = u_{q\_GLS} + u_{r\_GLS} \qquad (10)$$

Where u_{q_GLS} is the unicast cost of sending the location query message to one of the location servers, then forwarding to the destination, and u_{r_GLS} is the unicast cost of query reply from destination node to source node.

We can calculate $u_{q\_GLS} = d_{q\_GLS} / z$ and $u_{r\_GLS} = d_{r\_GLS} / z$. d_{q_GLS} is the average distance the location query message travels from source node to one of location servers, then to the destination. It can be approximated as twice the average distance between two random points in the whole h-order square with an area of $2^{h-1}R \times 2^{h-1}R$, which is about $\frac{2}{3}2^{h-1}R\sqrt{2}$. The factor two comes from the fact that the location server has to forward the query to the destination node. Similarly, query reply d_{r_GLS} takes an average distance of $\frac{1}{3}2^{h-1}R\sqrt{2}$ from the destination to the source of the reply. Therefore,

Theorem 4.5. *The cost of locating a node in GLS c_{q_GLS} is* $\propto N^{\frac{1}{2}}$.

Proof. We can simplify the expression for c_{q_GLS} as

$$\overline{c_{q\_GLS}} = \frac{2^{h-1}R\sqrt{2}}{z}$$

Substituting the value of h, then

$$\overline{c_{q\_GLS}} = \frac{R\sqrt{\frac{2\pi r_t^2(N-1)}{\lambda R^2}}}{z} = \frac{1}{z}\sqrt{\frac{2\pi r_t^2(N-1)}{\lambda}}$$

Thus $c_{q\_GLS} \propto N^{\frac{1}{2}}$. □

4.4.2 Cost of Locating A Node C_{q_Ensls} Packets in Ensls

In EnSLS, when a source node wants to locate some destination node, it needs to multicast location query packets to the home coordinates of (h-1) hashed minimum partitions. One of the location servers receives the location query messages and sends back a reply message with destination's current position. The cost of performing this activity is

$$c_{q\_EnSLS} = u_{q\_EnSLS} + u_{f\_EnSLS} + u_{r\_EnSLS} \qquad (10)$$

Where u_{q_EnSLS} is the multicast cost of sending the location query messages to the location servers, u_{f_EnSLS} is the cost of forwarding the query from a location server to the destination, and u_{r_EnSLS} is the unicast cost of query reply from destination node to source node.

We can calculate $u = d / z$. As we know, the average distance between two random points in an area of $R \times R$ is about $\frac{1}{3}R\sqrt{2}$. d_{f_EnSLS} and d_{r_EnSLS} can be approximated as the average distance between two random points in the whole h-order square. Thus, $d_{f\_EnSLS} = \frac{1}{3}2^{h-1}R\sqrt{2}$ and $d_{r\_EnSLS} = \frac{1}{3}2^{h-1}R\sqrt{2}$.

Finally, we estimate d_{r_EnSLS} as the sum of the average distances between two random points in 1-order square, 2-order square, …, and so on, (h-1)-order square.

$$d_{q\_EnSLS} = \frac{\sqrt{2}R}{3}(1+2+\ldots+2^{k-1}+\ldots+2^{h-2})$$

Simplifying the expression as

$$d_{q\_EnSLS} = \frac{\sqrt{2}R}{3}(2^{h-1}-1)$$

Now we can obtain the following theorem.

Theorem 4.6. *The cost of locating a node in EnSLS c_{q_EnSLS} is* $\propto N^{\frac{1}{2}}$.

Proof. We can simplify the expression for c_{q_EnSLS} as

$$\overline{c_{q\_EnSLS}} = \frac{\overline{d_{q\_EnSLS}} + \overline{d_{f\_EnSLS}} + \overline{d_{r\_EnSLS}}}{z}$$

Substituting the values

$$\overline{c_{q\_EnSLS}} = \frac{\sqrt{2}R}{3z}(2^{h-1}-1+2^{h-1}+2^{h-1}) = \frac{\sqrt{2}R}{z}2^{h-1} - \frac{\sqrt{2}R}{3z}$$

Substituting the value of h, then

$$\overline{c_{q\_EnSLS}} = \frac{R\sqrt{\dfrac{2\pi r_t^2(N-1)}{\lambda R^2}}}{z} - \frac{\sqrt{2}R}{3Z}$$

Thus $c_{q\_EnSLS} \propto N^{\frac{1}{2}}$. □

4.5 Location Management Overhead

The overhead cost of a location management scheme includes the above three parts: location update cost, location maintenance cost and location query cost. Based on the analysis of the costs of three parts, we compute and compare the average location management costs of GLS and EnSLS schemes. The results show that the average location management cost generated by GLS is proportional to the product v and $N^{\frac{3}{2}}$, while the EnSLS's scheme average location management cost is proportional to $(N^{\frac{3}{2}} + vN \log N)$. Therefore, we can expect the EnSLS scheme has better performance than that of GLS in practice.

4.5.1 Cost of C_{gls} Routing Packets Per Second in GLS

Now that we have estimated each of the individual costs involved in maintaining location information and finding the location information, we can estimate the total cost of routing packets in GLS. Let us assume that packets arrive at each node at a rate of ∂ packets per second according to a Poisson process. Then we can write the average cost of routing per second is

$$\overline{c_{GLS}} = N\sigma\overline{c_{q\_GLS}} + \overline{c_{m\_GLS}} + N\overline{c_{u\_GLS}} \qquad (11)$$

We have the following theorem:

Theorem 4.7. *The cost of routing packets per second in GLS is $\propto vN^{\frac{3}{2}}$*

Proof. Simplifying the above equation we obtain

$$\overline{c_{GLS}} \propto Nv\sqrt{N} + vN + N\sigma\sqrt{N} \propto vN^{\frac{3}{2}} \qquad \square$$

4.5.2 Cost of C_{ensls} Routing Packets Per Second in Ensls

Now that we have estimated each of the individual costs involved in maintaining location information and finding the location information, we can estimate the total cost of routing packets in EnSLS. Let us assume that packets arrive at each node at a rate of ∂

packets per second according to a Poisson process. Then we can write the average cost of routing per second is

$$\overline{c_{EnSLS}} = N\sigma\overline{c_{q\_EnSLS}} + \overline{c_{m\_EnSLS}} + N\overline{c_{u\_EnSLS}} \qquad (11)$$

We have the following theorem:

Theorem 4.8. *The cost of routing packets per second in EnSLS is*

$$\propto (N^{\frac{3}{2}} + vN\log N)$$

Proof. Simplifying the above equation we obtain

$$\overline{c_{EnSLS}} \propto vN\log N + vN + N\sigma\sqrt{N} \propto (N^{\frac{3}{2}} + vN\log N) \qquad \square$$

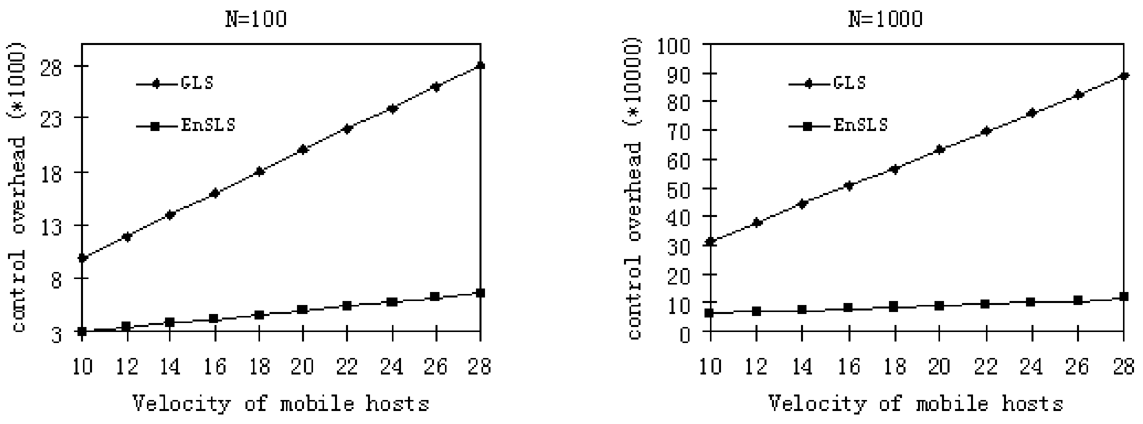


Figure 4.5. The comparison of growth functions of the total control overhead of two location services.

In Figure 4.5, we plot the growth functions of the total control message overhead for both location services, GLS and EnSLS, when the mobility of networks increase, and the number of hosts is a constant ($N = 100$, $N = 1000$). The plots show our proposed protocol has much better performance than that of GLS. In summary, our proposed new location service, EnSLS, should outperform theoretically GLS in terms of the number of control messages in MANET, since GLS grows as $O(vN^{\frac{3}{2}})$ while that of EnSLS grows as $O(N^{\frac{3}{2}} + vN \log N)$. We can conclude EnSLS is more scalable compared with GLS.

CHAPTER 5

USING A VIRTUAL BACKBONE TO IMPROVE ROUTING

5.1 Background

The simplest way of updating routing information in ad hoc network is to send data about the neighborhood of nodes through all available links. This simple technique is called “global flooding”. The main drawback of “global flooding” is the excessive amount of redundant rebroadcasts through the network, thus, degrading its available bandwidth, causing contention and collision easily, the lack of packet delivery successfully guaranteed, etc.

In large and/or dense ad hoc and sensor networks, it is not necessary to use all available nodes for performing data communication tasks. In sensor networks, for example, some nodes are sensing areas, while some other nodes are there to support routing, as basic data communication protocol for data gathering. Some or all sensors can at the same time perform both of the tasks, sensing and forwarding traffic. There are several reasons to reduce the number of nodes needed for monitoring or routing. Face routing [16], for instance, has better performance on a connected dominating set than on a full set [24], since there are fewer nodes, and consequently longer edges, to traverse in the considered planar graph.

In order to avoid the “global flooding” problem, many topology-based routing protocols propose the promising idea of virtual backbones such as cluster-based routing, backbone-based routing and spine-based routing [22, 52, 53, 92] to overcome the "global flooding" problem even a mobile ad hoc network has no fixed backbone infrastructure. The basic idea behind these types of protocols is to divide a mobile ad-hoc network into several small overlapping subnetworks, where each subnetwork is a clique (a complete subgraph). Each subnetwork has one or more virtual backbone hosts to connect to other parts in the network. These virtual backbone hosts form the core infrastructure of the ad-hoc mobile network. The routing process is operated over the core. As a result, any broadcasting of control packets only happens in the core, and communications between core nodes and non-core nodes are all through unicast communications. Therefore, this can substantially reduce the protocol overhead caused by global flooding. The number of hosts forming the virtual backbone must be as small as possible in order to reduce the protocol overhead, to increase the convergence speed, and to simplify the connectivity management. In this case, defining the structure of a suitable backbone is one of the subproblems that must be solved with the objective of providing optimal routing between clients.

The idea of using a virtual backbone for ad hoc systems had been discussed earlier by Ephremides *et al.* [26] and by Gerla and Tsai [33], but only with Das and Bhaarghavan [15] the objective of optimizing the size of the backbone was explicitly studied. Das and Bhaarghavan [15] proposed a distributed algorithm to solve the routing problem, based on a virtual backbone technique and using minimum connected dominating sets (MCDS) in 1997. Virtual backbone routing usually consists of 6 steps [76]:

1. Compute the virtual backbone B, i.e. using an algorithm to approximate MCDS.

2. If the source v, $v \notin B$, v forwards the data packets to one of its neighboring nodes which belong to B. The neighbor acts as a new source to route the packets
3. Broadcast the topology information to all nodes in B.
4. Determine routes in B using some shortest path algorithm
5. Eventually, the packets arrive at a backbone node, which is either the destination node itself or a backbone node in the dominating neighbor set of the destination node.
6. Sending periodic routing maintenance update.

The main objective of virtual backbone algorithms is to reduce the number of nodes for which routing information must be stored. This is achieved by initially computing a virtual backbone B. The nodes in the backbone are then capable of directly collecting information about connectedness for neighbor nodes not in B, and they send this information to other backbone nodes. In this way, information about the complete network is updated, and the number of necessary message exchanges is reduced. The following result can be proved for the algorithm above in [15]:

Theorem 5.1 *To compute the shortest paths between all pairs of nodes in the given network, the Virtual Backbone routing algorithm takes $O(\Delta(n+|C|+\Delta))$ time, using $O(n^2/\Delta+n|C|+m+n\log n)$ messages, where Δ is the maximum number of neighbors for any node, and $|C|$ is the size of connected dominating set* [15].

5.2 Network Model

We assume that a mobile ad-hoc network is deployed in a 2D space, and a mobile node is equipped with an omni-directional antenna that has an equal maximum transmission range. Thus, the topology of such a mobile ad-hoc network can be modeled as a unit-disk graph (UDG). In the case, the radius of the region covered by the signal of each mobile node is scaled to be equal to one. "A graph is a unit graph if and only if its vertices can be put in one to one correspondence with equisized circles in a plane in such a way that two vertices are joined by an edge if and only if the corresponding circles intersect." [68]. A simple graph G (V, E) is used to represent a mobile ad-hoc network, where V represents a set of mobile nodes and E represents a set of edges. An edge (u,v) in E indicates that nodes u and v are neighbors, and that u is within v's range of transmission, while v is within u's range.

Unit disk graphs have many useful geometric properties. In fact, any property of Euclidean graphs is valid for unit disk graphs, since the former is a generalization of the latter. For example, it is known that planar graphs have a structure that allows easier approximation, even for NP-hard approximation problems in general. General approximation algorithms for problems in planar graphs exist for problems such as maximum independent set, partition into triangles, minimum vertex cover, and minimum dominating set. Baker [12] has shown that for these problems there are polynomial algorithms with constant approximation guarantee. In fact, in most cases the approximation algorithms run in linear time.

One of the useful properties of unit graphs is the relationship between sizes of their independent sets and dominating sets. This property is based on the tradeoff between the number of nodes that can be located in a specific region and the number of links that will result from this co-location. A basic lemma proved by Alzoubi *et al.* [8] and used in the analysis of many approximation algorithms for the connected dominating set problem is the following.

Lemma 5.1 *Let I be a maximal independent set on a unit disk graph G, and C an optimal connected dominating set on G. Then,* $|I| \leq 4|C| + 1$

5.3 Approximation of Virtual Backbone by Connected Dominating Set

Currently, Minimum Connected Dominating Set (MCDS) is the main method utilized to approximate the virtual backbone in a mobile ad-hoc network. For future reference, we now formally define dominating sets.

Definition 5.1: *A dominating set (DS) is defined by a subset of vertices of a graph where every vertex that is not in the subset is adjacent to at least one vertex in the subset.*

Definition 5.2: *A connected dominating set (CDS) is a dominating set that induces a connected subgraph.*

Definition 5.3: *The size, S(DS) of a dominating set is the number of nodes in DS.*

The MCDS problem requires the computation of a minimum sized DS is connected, i.e. there is a path between each pair of nodes in CDS. However, finding a MCDS is NP-complete for most graphs.

5.4 Current Research for Determining Connected Dominating Set

Various algorithms that construct a CDS in ad hoc networks have been proposed in recent years. They can be divided into two categories: centralized algorithms that depend on network-wide information or coordination and decentralized that depend on local information only. Centralized algorithms usually yield a smaller CDS than decentralized algorithms, but their application is limited due to the high maintenance cost.

Centralized approximation algorithms for MCDS in mobile ad-hoc networks were first developed by Das *et al.* [15, 22, 92]. These algorithms provided distributed implementations of the two centralized algorithms given by Guha and Khuller [36]. In Das's algorithm, a CDS is found by growing a set *U* starting from a vertex with the maximum node degree. It then iteratively adds to *U* a node that is adjacent to the maximum number of nodes not yet in *U* until *U* forms a dominating set. Finally, it assigns each edge with a weight equal to the number of neighbors not in *U*, and then finds a minimum spanning tree *T* in the resulting weighted graph. All the non-leaf nodes form a CDS. This approach has two main improvements over previous protocols. Firstly, only a few nodes need to keep global information that captures the topological structure changes of the whole network, and as long as network topological changes do not affect these MCDS nodes, there is no need to recapture global information. Thus it reduces the information access overhead and the update overhead. Secondly, each node only needs 2-distance neighborhood information instead of information of the entire network topology. The main shortcoming of this algorithm is that the process of "constructing a spanning tree" is almost sequential, thus it needs a non-constant number of rounds to determine a CDS. Furthermore, the algorithm suffers from high implementation complexities and message complexity. Another algorithm based on a spanning tree was proposed by Wan *et al.* [8]. In this scheme, a maximal independent set (MIS) is elected such that each vertex in the MIS can be connected to the spanning tree via an extra vertex. Since, in unit disk graphs, the size of an independent set is at most four times that of the minimum CDS, this algorithm has approximation ratio of 8. However, this algorithm usually produces a larger CDS than the MCDS algorithm in random unit disk graphs.

Decentralized algorithms can be further divided into cluster-based algorithms and pure localized algorithms. Cluster-based algorithms have a constant approximation ratio in unit disk graphs and relatively slow convergence ($O(N)$ in the worst case). Pure localized algorithms take constant steps to converge, produce a small CDS on average, but have no constant approximation ratio. A cluster-based algorithm usually contains two phases. In the first phase, the network is partitioned into clusters and a clusterhead is elected for each cluster. In the second phase, clusterheads are interconnected to form a CDS.

Stojmenovic *et al.* [96] proposed a cluster-based construction of CDS. Two types of nodes exist in the CDS: the cluster-heads and the border-nodes. The cluster-headers form a maximal independent set (MIS). The framework of constructing an MIS is described as follows:

- Each node uses a record key = (*degree*, x, y) as a unique rank parameter, where *degree* is the number of neighbors of the node and x and y are its two coordinates in the deployed network. The ranks are used to order all nodes.
- Each node with the lowest rank among all neighbors declares itself as a cluster-head by broadcasting.
- Whenever a node receives a message for the first time from a cluster-head, it gives up election as a cluster-head.
- Whenever a node has received the giving-up messages from all of its neighbors with lower ranks, it broadcasts a message to declare itself as a cluster-head.

After a node collects the status of all neighbors, it joins the cluster centered at the neighboring cluster-head with the lowest rank. Nodes adjacent to some node from a different cluster are selected as border-nodes. The implementation cost depends on the choice of rank. The algorithm is not very efficient because of the non-selective inclusion of all border-nodes. Also, the process for selecting cluster-heads may have to be serialized in some situations, such as in a linear network with monotonically increasing or decreasing ID distribution along the network.

Several approaches were proposed to construct a CDS by connecting clusterheads via nonclusterheads called connectors; that is, both clusterheads and connectors belong to CDS. In early schemes [9, 61], every nonclusterhead that has a neighbor in another cluster is designated as a connector, which results in a larger CDS. The objective here is to maximize the throughput and reliability, rather than to reduce the CDS size. Alzoubi *et al.* [7] proposed growing a tree to reduce the number of connectors. The root of this tree is the winner of a distributed election among clusterheads, and other clusterheads are connected to the tree via at most two connectors per clusterhead. This algorithm is an early version of [8]; it has an approximation ratio of 12 to the optimal size and a slow converging speed. Most approaches [26, 54] use a mesh structure, which is much faster to construct than a tree. In the mesh scheme, each clusterhead designates one or two connectors to form a path to each neighboring clusterhead (i.e., a clusterhead two or three hops away). The mesh scheme also has a constant approximation ratio, but this constant is much larger than 12.

In pure localized algorithms [17, 20, 83, 105], the status of each node depends on its h-hop topology only, where h is a small constant, and usually converges after at most h rounds of information exchange among neighbors.

Wu and Li [105] proposed the first simple and efficient pure localized algorithm that can quickly find a DS in a mobile ad hoc network. Each node is marked as white initially. Let $N(v)$ be the open neighbor set of vertex v, which means $N(v)$ includes all the neighbors of vertex v. And let $N[v]$ be the closed neighbor set of vertex v, the set of all neighbors and itself. By assumption, each node has a unique ID number. This algorithm runs in two phases. In the first phase, each node broadcasts its neighbor set $N(v)$ to all its neighbors, and after collecting all adjacency information from all neighbors every node marks itself as black if there exist two unconnected neighbors. All black nodes form the initial CDS. However, considering only the first phase, there are too many nodes in the dominating set. So in the second phase, the algorithm executes extensional rules to eliminate local redundancy. Wu and Li [20] proposed several dominant pruning rules, which are rule 1, rule 2 and a generalized rule K. Thus, the second phase removes some nodes from the original dominating set and the size of a dominating set is further reduced.

Qayyum *et al.* [83] proposed an efficient broadcast scheme called mutipoint relaying (MPR). In MPR, each host designates a small set of 1-hop neighbors (MPRs) to cover its 2-hop neighbors. In the broadcasting, a host u forwards a packet p from the last hop v only if 1) u has not received p before and 2) u is a MPR of v. For each broadcasting, forwarding hosts form a source-dependent CDS (i.e., a dynamic CDS depends on the broadcast process). By taking advantage of the broadcast history information, a source-dependant CDS is usually smaller than a source independent CDS constructed by above algorithms. It was proven in [83] that MPRs selected by a single host has $\log\Delta$ approximation ratio. However, it is unknown if a global approximation ratio exists for the entire CDS. Tseng *et al.* [101] proposed several efficient broadcasting schemes for ad hoc networks, but none of them form a CDS.

5.5 Problem Statement of Localized Algorithms for Computing CDS

In this dissertation, we focus on improving pure localized algorithms for computing CDS because of their low overhead and fast convergence, two essential requirements for a routing protocol in ad hoc networks. To the best of our knowledge, pure localized algorithms usually consist of two phases. In the first phase, called as marking process, some nodes declare themselves as dominating nodes according to collected neighboring information, which promise a large size CDS. Then, a node change its status from dominating to dominated if it neighbors and itself can be fully covered by one of its neighbors. Thus, the size of CDS is decreased.

Wu and Li [105] proposed a simple marking process, every node collects two-hop neighboring information, if a node finds that it has two unconnected neighbors, it declares itself as a dominating node. Chen *et al.* [17] proposed an approach similar to the marking process, called Span, to select a set of special hosts called coordinators. Ideally, coordinators form a CDS such that other hosts can switch to the energy saving mode without compromising the routing capability of the network. A host *v* becomes a coordinator if it has two neighbors that are not directly connected, indirectly connected via one intermediate coordinator, or indirectly connected via two intermediate coordinators. Before a host changes its status from noncoordinator to coordinator, it waits for a backoff delay which is computed from its energy level and 2-hop neighborhood topology. The backoff delay can be viewed as a priority value, such that nodes with shorter backoff delay have a higher chance of becoming coordinators. Span cannot ensure a CDS when two coordinators simultaneously change back to noncoordinators.

We can see the marking process of Wu and Li induces too many nodes that belong to a CDS especially in dense ad hoc networks because almost every node has two unconnected neighbors. The information of three-hop neighbors is required in Span marking process. We propose a new Marking Process by Classification of Neighbors (MPCN) in Chapter 6. Only the information of two-hop neighbors is required in MPCN, and a node declares itself as a dominating node only if it can cover as many as possible nodes or it acts as a connector to other parts of the whole ad hoc network. The detailed description and performance analysis of MPCN are shown in Chapter 6.

The size of initial CDS produced by marking process is large. Many algorithms have proposed "redundancy elimination" rules to reduce the size of CDS. The most popular one is that if a node finds its whole neighborhood can be covered by one of its neighbors, it changes its status as a dominated node instead of a dominating node. In order to avoid 'illegal simultaneous' removals from CDS, and thus disconnect the network, the comparisons of some key values are adopted to break the tie when two nodes or three nodes can be covered by each other. Wu, *et al.* described, in a series of articles (the first one being [105]), a lightweight pure localized redundancy eliminating scheme. A node is an intermediate node if it has two unconnected neighbors [105]. A node A is covered by neighboring node B if each neighbor of A is also neighbor of B, and key(A)<key(B). An intermediate node not covered by any neighbor becomes an inter-gateway node. A node A is covered by two connected neighboring nodes B and C if each neighbor of A is also a neighbor of either B or C (or both), key(A)< key(B), and key(A) <

key(C). An inter-gateway node not covered by any pair of connected neighboring nodes becomes a gateway node.

The concepts will be illustrated in Figure 5.1. Assume that key(X)=(degree(X), ID(X)), where degree(X) is number of its neighbors, and ID(X) is its alphabetical name (such keys are introduced in [96]). Let $X < Y$ if and only if degree(X)<degree(Y) or ((degree(X)=degree(Y) and ID(X) < ID(Y)). Assume that IDs are ordered alphabetically ($A < B < C < \ldots$), which is used in comparison when degrees are same. If Figure 5.1, nodes F, J, K, D do not have two unconnected neighbors and are not intermediate. Node X is covered by Y if any neighbor of X is neighbor of Y and key(X) < key(Y). In Figure 5.1, L is covered by H. Node X is covered by two connected neighbors Y and Z if any neighbor of X is neighbor of either Y or Z (or both), key(X) < key(Y) and key(X) < key (Z). In Figure 5.1, C is covered by A and H. If node A had one more neighbor, or its key was larger than G (say N) then node G would have been covered by two connected nodes A and H. With the degrees and keys as in Figure 5.1, and given definition [93, 105], dominating nodes are A, H and G. Nodes A and H do create a connected dominating set, but not with the given definition.

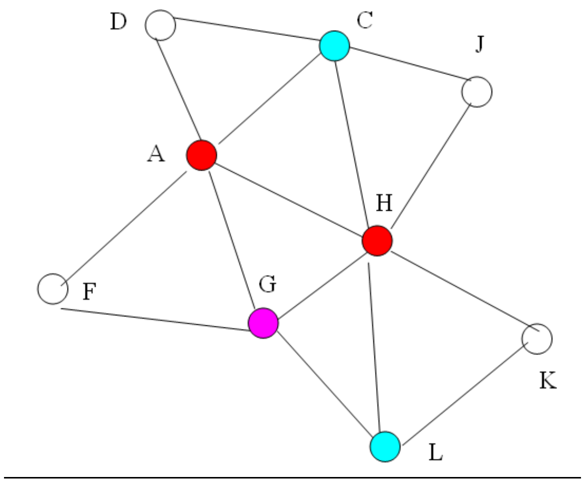


Figure 5.1. A connected dominating set A, G, H

In [107], key(X) = (energy(X), degree(X), ID(X)) is proposed as new key. This means that nodes with higher energy level are preferred for selection in a dominating set, that is, to be active. When energy levels of two nodes are same, degrees are used for comparison. If even degrees are same, then IDs resolve ties. Nodes are time synchronized, and at the beginning of each new round exchange information about their keys, and decide about active nodes for the next round. Experimental measures show increased network life, when network at the beginning is sufficiently dense (note that, for sparse networks, some nodes may be forced to be always active). In [89], a number of other keys were tested, in search for a best one. Among tested ones, the surprising winner was key(X) = (energy(X)/degree(X), degree(X), ID(X)). That is, lower degree nodes are preferred when energy levels are same. However, it [89] did not provide any explanation for this phenomenon.

The abstract version of this article [93] is the first one to consider changing relative priorities at the way to reduce CDS. More precisely, the IDs used by nodes remain fixed, but become secondary key in comparisons if some conditions regarding sets of neighbors of each node are satisfied. This idea was later further explored in [104]. The difference in their approach is that IDs of nodes are changed in iterations, and then existing CDS protocols are applied. The iterations and modifications are performed in variety of ways with the overall goal of reducing the size of CDS and maintain CDS in dynamic networks scenarios (mobility or changes in activity status), or even in static networks by using additional messages. Wu, Dai and Yang [104] propose the following general model. After gaining h-hop neighbor knowledge $(h>1)$, determines its status using any CDS protocol. Nodes then select their new priorities and exchange their decisions (and new priorities, if they cannot be predicted) with neighbors. Authors [104] propose several ways to change priorities. One is to apply, to each of n nodes, n distinct values ID(x) in range $[1,n]$ and apply, in round s, the priority determined by the value s+ID(x) mod n. Priorities can be predicted in this scenario; however, it is difficult to provide such unique identifiers in dynamic ad hoc networks. Other options, such as using new random numbers, require to inform neighbors about new IDs. This general approach is applied to over existing generalized CDS marking scheme [21] which can be briefly described as follows. A node is unmarked if its neighbors are pairwise connected, or are neighbors of a set of connected nodes with higher priorities [21]. Initially, all nodes are considered marked. At each round, only marked nodes use generalized rule [21] to determine their status, marked or unmarked, after this round. Unmarked nodes stay unmarked. When applying generalized rule, only marked nodes can be used as coverage nodes to unmark other marked nodes. In the SILS (Seamless iterative local solution) [104], at each round, generalized rule [21] is applied at all nodes, previously marked or unmarked, to determine their new status. Nodes that were unmarked in the previous iteration may increase their IDs in the meanwhile. The authors [104] also consider adding a watermark to nodes ID as the primary key in comparisons. Watermarks record the most recent iteration that a node was marked, so that a node with a higher watermark has higher priority. We note that so defined priorities may lead to oscillations in each iteration between neighbors, which ‘trade’ their CDS status in each iteration. We also note that, in addition to messages being exchanged at end of each iteration to inform about new CDS status, messages are sent, in basic solutions [21, 106] during the decision process within each iteration.

In Chapter 7, we propose the following Iterative Localized Algorithm for CDS (ILA_CDS), improving the concept by Wu and Li [105]. In our algorithm, each node sends five messages to all its neighbors, and the process is synchronized. Each node that decides not to be in CDS (connected dominating set) becomes passive; otherwise it is active and reevaluates the decision in the next round. First, each node verifies whether or not it has two unconnected neighbors, and informs neighbors about it. Next, each active node (that remains in CDS) after the first step will check whether it is covered by one neighbor with higher ID, and all nodes then send message informing about their CDS status. In the next step, each active node checks whether or not it is covered by one of remaining active neighbors with the lower ID. All nodes again send the message by the algorithm (although nodes, once declared as passive, do not need to send further messages). In the next iteration, active nodes covered by two active neighbors with higher IDs decide to become passive, and informs neighbors by a message. In the next iteration, active nodes that have two active neighbors that together cover it, one with higher ID and one with lower ID, decide to withdraw from CDS and become passive, informing neighbors about the status change. In the final iteration, all active nodes that are covered jointly by two active neighbors with lower IDs also decide to become passive and withdraw from CDS. They may send a sixth message if each node wants to know which of its neighbors remains in CDS. The experimental data show that the size of CDS has been reduced dramatically with respect to the concept originally proposed by Wu and Li [105].

We select one typical algorithm from each category, and the performance comparison of these algorithms and our two new algorithms is briefly summarized in Table 5.1. Since different assumptions and models are used in these proposed algorithms, it is difficult to make comparisons among them. The metrics used for comparison include computation and message complexities, the size of the generated CDS (this performance measurement is obtained either through theoretical analysis of the worst case in terms of approximation ratio to MCDS or through simulation on average case), whether the method supports locality of maintenance, and whether it requires global information.

	[22, 15, 92]	[96]	[24]	[105]	MPCN	ILA_CDS
Message	$O(n^2)$	$O(n^2)$	$O(n \log n)$	$O(n\Delta)$	$O(n\Delta)$	$O(n\Delta)$
Time	$O(n^2)$	$O(n^2)$	$O(n\Delta)$	$O(\Delta^2)$	$O(\Delta^2)$	$O(\Delta^2)$
Size	$\leq(2\ln\Delta+3)opt$	Sim	$\leq 8opt+1$	Sim	Sim	Sim
Maintenance Locality	*no*	*no*	*no*	*Yes*	*Yes*	*Yes*
Information	Global	Partial global	Partial global	Local	Local	Local

Table 5.1: Performance comparison of the algorithms in [22, 15, 92], [96], [24], [105] and that proposed in this paper. Here *opt* is the size of MCDS; *n* is the number of nodes; Δ is the maximum degree; Sim means that the size of the generated CDS is measured through simulation; *no* means it does not support locality of maintenance; *yes* represents it supports locality of maintenance; the last column labeled by New corresponds to our algorithm.

We see our algorithms are superior over those algorithms in [22, 15, 92] and [96]. The algorithm in [8] has good performance and approximation ratio to MCDS, but it needs to establish a global spanning tree, which makes local maintenance and propagation of information impossible. It is difficult for [96, 105] and our algorithms to obtain a theoretical approximation ratio of the generated CDS size to that of MCSD. Thus, we conducted extensive simulations to compare the sizes of the generated CDS in [105] and our algorithms on average cases. Simulation results demonstrate that our algorithms reduce the size of the CDS largely compared with that of [105], while the algorithm in [105] has better performance than those in [22, 15, 92]. As we know, the most important objective in designing a routing protocol is to reduce broadcast redundancy to save limited resources and to avoid the broadcast storm problem. This is achieved by generating a smaller CDS. Therefore our algorithms do contribute in this aspect and provide benefits, though there is an acceptable increase in computation complexity.

CHAPTER 6

COMPUTING CONNECTED DOMINATING SET BY CLASSIFICATION OF NEIGHBORS

6.1 Introduction

A common feature of the currently available cluster-based techniques for solving the MCDS problem is that the algorithms create the CDS from scratch, adding some vertices according to a greedy criterion at each iteration. Such algorithms have some shared disadvantages. One main disadvantage is that they require additional setup time to construct a CDS from scratch, which induces slow convergence.

In this chapter, we propose a new heuristic algorithm for computing approximate solutions to the minimum connected dominating set problem. In particular, we discuss in detail the application of this algorithm to the MCDS problem in unit-disk graphs. The algorithm starts with a feasible and near-optimal CDS solution via mark processing, and removes vertices from this solution by redundancy elimination, until an approximate CDS is found. Using this technique, the proposed algorithm forms a feasible and small-sized CDS solution at the first stage; therefore, there are no setup time requirements. The approach also has the advantage of being purely localized and only exploring the local structure of a given MANET. Experimental results show that it is comparable to the best existing algorithms and produces a smaller size CDS than Wu and Li's algorithm [105].

This chapter adopts standard graph-theoretical notations. Given a graph $G = (V, E)$, a subgraph of G induced by the set of vertices V' is denoted by $G[V']$. The set of adjacent vertices (also called neighbors) of $v \in V$ is represented by $N(v)$. Also we use $\delta(v)$ to denote the number of vertices adjacent to v, i.e. $\delta(v) = | N(v)|$. Δ represents the maximum degree of graph G.

The chapter is organized as follows. The following section presents the principle of classification of a vertex's neighbors, and then illustrates the new marking process. The connected dominating set reduction is discussed in Section 6.3. Section 6.4 presents the analytical performance evaluation. Performance evaluation by experiments is done in Section 6.5.

6.2 Mark Process by Classification of neighbors

Given a simple undirected graph $G = (V, E)$, where V is a set of vertices (hosts) and E is a set of undirected edges, an edge between u and v is denoted by an pair (u, v). A set $V' \subset V$ is a dominating set of G if every vertex $v \in V- V'$ is dominated by at least one vertex $u \in V'$. If a node $u \in V'$, we call u as a dominating node, otherwise it is a dominated node.

Neighborhood of a single vertex. [5] Consider a vertex $v \in V$ of the given graph $G = (V, E)$, let $N(v) = \{u \mid (u,v) \in E\}$ be the neighborhood of v. We partition the vertices of $N(v)$ of v into three different sets $N_1(v)$, $N_2(v)$, and $N_3(v)$ depending on what neighborhood structure these vertices have. More precisely, setting $N[v] = N(v) \cup \{ v \}$, called close neighboring subset of a node v, we define

$$N_1(v) = \{u \in N(v) \cap N(u) \setminus N[v] \neq \emptyset \}$$

$$N_2(v) = \{(u \in N(v) \setminus N_1(v)) \cap (N(u) \setminus N_1(v) \neq \emptyset)\}$$

$$N_3(v) = \{ N(v) \setminus (N_1(v) \cup N_2(v)) \}$$

An example that illustrates the partitioning of $N(v)$ into the subsets $N_1(v)$, $N_2(v)$, and $N_3(v)$ can be seen in Figure 6.1.

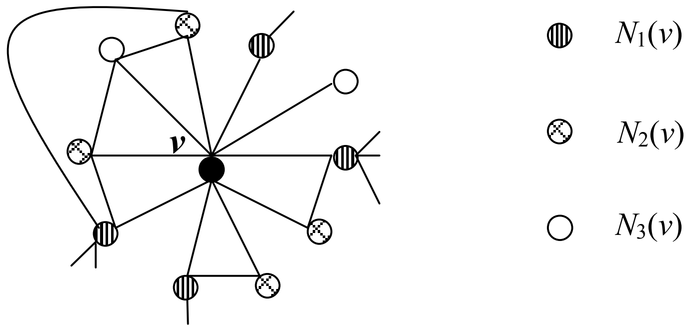


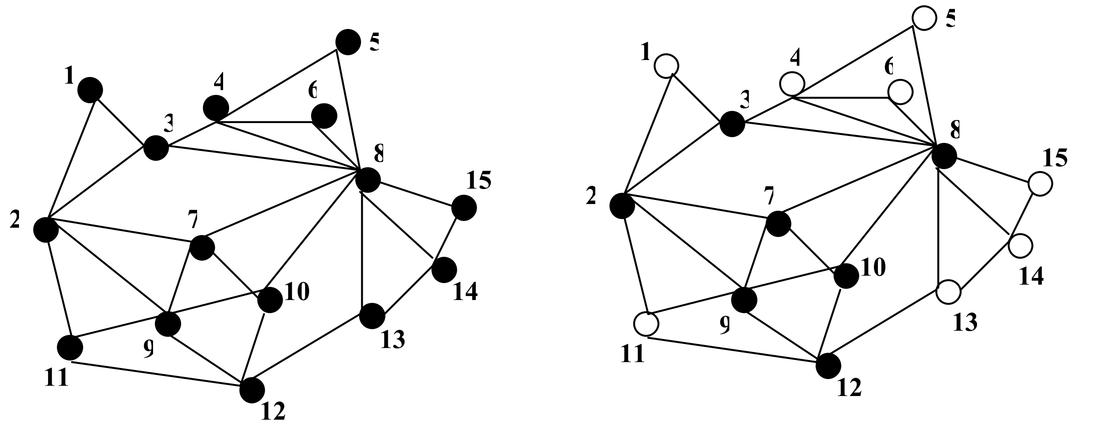
Figure 6.1. The partitioning of the neighborhood of a single vertex *v*

Note that, by definition of the three subsets, the vertices in $N_1(v)$, cannot be dominated by vertices from $N_1(v)$. A vertex in $N_3(v)$ can only be dominated by either v or by vertices in $N_2(v) \cup N_3(v)$. Since v will dominate at least as many vertices as any other vertex from $N_2(v) \cup N_3(v)$, it is safe to place v into the optimal dominating set we seek for. Based on the partitioning method of the vertex neighborhood, the following marking process can quickly find a connected dominating set in a given graph.

Algorithm 1 Marking Process by Classification of Neighbors.

```
1: Initially assign color $BLANK$ to each $u$ in $V$.
2: Each $u$ exchanges its neighbor set $N(u)$ with all its neighbors.
3: Each $u$ checks its color, if its color is WHITE already, STOP.
4: Each $u$ partitions all its neighbors into three subsets $N_1(u)$, $N_2(u)$, and $N_3(u)$.
5: If $N_2(u) \cup N_3(v) \neq \varnothing$
        $u$ sends a "dominating" message to all nodes in $N_2(u) \cup N_3(v)$.
7: If $u$ receiving a "dominating" message from one neighbor $v$
        If $N[u] \neq N[v]$ then    $u$ changes its color c($u$) to WHITE
        else
                If $ID(u) < ID(\mathrm{v})$  then    $u$ changes its color c($u$) to WHITE
```

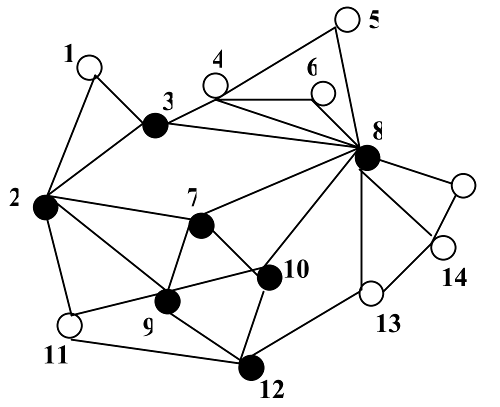
The marking process is a localized algorithm, where hosts only interact with others in the neighborhood. Unlike clustering algorithms, there is no "sequential propagation" of information. The marking process colors every vertex in G. $c(v)$ is a color value for vertex $v \in V$ that has two values, WHITE, and BLANK. Nodes remaining color BLANK form the initial CDS. Nodes with WHITE color are dominated by other nodes. An important issue in implementing the marking process is that during any step of this process, once a node receives a "dominating" message, it changes its color to "WHITE" from "BLANK" immediately, and then stops the marking process. Because all mobile hosts execute the marking process in a distributed manner, they do not know whether they are dominated by others and when they can stop the process except for receiving the notice messages. For each host v, its neighbor set $N(v)$ are embedded in the beacon packet sent periodically to each neighbor. Thus, each host v has the two-hop neighboring information.



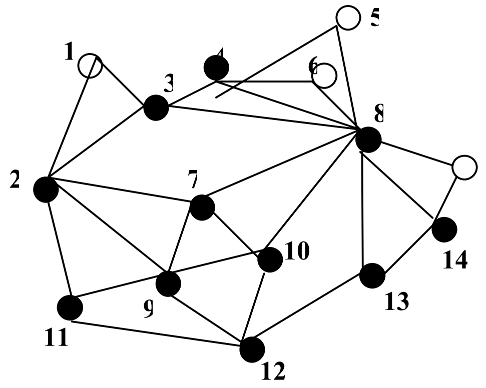
(a) Initial graph of MANET **(b) the result after the marking process**

Figure 6.2 Example of marking process by classification of neighbors

Figure 6.2 illustrates an example of the marking process by classification of neighbors. According to Algorithm 1, initially, each node *v* marks itself as BLANK as shown in Figure 6.2(a), and then partitions its neighbors into three subsets $N_1(v)$, $N_2(v)$, and $N_3(v)$ using the two-hop neighboring information collected by periodical beacons. If the union of a node's subsets $N_2(v)$ and $N_3(v)$ is not empty, it sends messages to notify nodes in $N_2(v)$, and $N_3(v)$ to change their color to WHITE, since they can be dominated by other nodes. For example, the close neighboring subset of node 8 is {3, 4, 5, 6, 7, 8, 10, 12, 13, 14, 15}, and the three subsets of its neighborhood is as follows: $N_1(8)$ = {3, 7, 10, 12}, $N_2(8)$ = {4, 13}, $N_3(8)$ = {5, 6, 14, 15}. Therefore, node 8 sends a "dominating" message to all nodes in $N_2(8)$. When nodes 4, 5, 6, 13, 14, 15 receive this message, they change their colors to WHITE, which indicates they are dominated. A node may receive several "dominating" messages, but it just changes its color to WHITE due to the first message and discards the following messages. Node 1 receives two "dominating" messages from its neighbors; node 2 and node 3.



(a) The result applying marking process of classification of neighbors



(b) The result applying marking process by Wu and Li's marking process

Figure 6.3. Comparison of two marking processes.

The marking process by classification of neighbors (MPCN) prefer to selecting nodes, which dominate as many as possible neighbors, into the initial CDS. While in Wu and Li's marking process [105], a node marks itself as a dominating node when it has two unconnected neighbors. We may wonder if MPCN produces a smaller CDS compared with Wu and Li's marking process [105] by intuition, especially in dense network or network of small size. Figure 6.3 gives an example of MPCD and Wu and Li's marking process [105] applying to a network.

Properties. Assume *V'* is the set of vertices that are colored BLANK in *V*, i.e., $V' = \{v \mid v \in V, c(v) = \text{BLANK}\}$. The reduced graph *G'* is the subgraph of *G* induced by *V'*, i.e., G'=G[V']. The following two theorems show that *G'* is a dominating set of *G* and it is connected.

Theorem 6.1: *Given a $G = (V, E)$ that is connected, the vertex subset V' induced by BLANK nodes after the marking process by classification of neighbors, forms a dominating set of G.*

Proof. Randomly select a vertex v in G. We show that v is either in V' (a set of vertices in V that remain color BLANK) or adjacent to a vertex in V'. Assume v is colored WHITE, if there is at least one neighbor colored BLANK, the theorem is proved.

We assume v is colored WHITE and all v's neighbors ($v_1, v_2, \ldots, v_k$) are colored WHITE. Based on the marking process by classification of neighbors, all nodes in graph G are initially colored as BLANK, and only if v receives a "dominating" message, it changes its color as WHITE. Let v_1 be the neighbor sending a "dominating" message to v. Then $N(v) \subset N(v_1)$. Since v_1 is also WHITE, and u is the neighbor sending a "dominating" message to v_1, u must be one of v's neighbors ($v_1, v_2, \ldots, v_k$) according to MPCN. Let u be just v_2, then $N(v_1) \subset N(v_2)$, and so on, until we can conclude $N(v_1) \subset N(v_2) \subset N(v_3) \subset \ldots \subset N(v_k)$. Thus, at least one neighbor of v can not be dominated by others, it should remain as BLANK. When two neighboring nodes have the same close neighboring subset, MPCN only changes the one with smaller ID to WHITE. This contradicts to the assumption that all v's neighbors are WHITE. □

Dominating sets were empty with definitions of Wu and Li's marking process [105], while this marking process by classification of neighbors selects node with lowest ID in the dominating set. Empty dominating sets in complete graphs are not an issue in broadcasting application since retransmissions are not necessary. However, in scheduling node activities it is a major issue, since it leaves such network without any active node (e.g. with no sensor to monitor given small area).

In order to prove the following theorem, we induce a new definition.

Definition 6.1: *$\{v, v_1, v_2, ..., v_k, u\}$ is a shortest path with the largest number of BLANK nodes between vertices v and u in G, if $\{v, v_1, v_2, ..., v_k, u\}$ has the least number of hops, and among the paths which have the least number of hops, it has the largest number of BLANK nodes.*

An example is illustrated in Figure 6.4. $\{v, v_1, v_2, u\}$ and $\{v, w, v_2, u\}$ have the same hops number between vertices v and u. However, $\{v, v_1, v_2, u\}$ is the shortest path with the largest number of BLANK nodes between vertices v and u, since Path $\{v, v_1, v_2, u\}$ has three BLANK nodes while $\{v, w, v_2, u\}$ has two BLANK nodes.

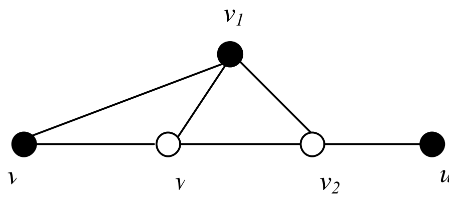


Figure 6.4. An example of the shortest path with the largest number of BLANK nodes

Theorem 6.2: *The reduced graph G' = G[V'] is a connected graph.*

Proof. We prove this theorem by contradiction. Assume G' is disconnected and v and u are two disconnected vertices in G'. Assume $dis_G(v, u) = k+1 > 1$ and $\{v, v_1, v_2, \ldots, v_k, u\}$ is a shortest path with the largest number of BLANK nodes between vertices v and u in G. Clearly, all $v, v_1, v_2, \ldots, v_k$ are distinct and among them we can find at least two nodes v_{i-1} and v_i, such that $c(v_{i-1})$ = BLANK and $c(v_i)$ = WHITE, from the node v_i this path becomes disconnected. (Otherwise, v and u are connected in G' and perhaps the node v_{i-1} is just v). On the other hand, the two adjacent vertices of v_i, v_{i-1} and v_{i+1}, are not connected in G. (otherwise, $\{v, v_1, v_2, \ldots, v_k, u\}$ is not a shortest path). When node v_i is colored as WHITE, we consider the following three cases: (1) node v_i receives a "dominating" message from node v_{i-1}. We can see $N(v_i) \subset N(v_{i-1})$ based on the marking process by classification of neighbors. Thus, v_{i+1} is connected to v_{i-1}. This contradicts to the assumption that the two adjacent vertices of v_i, v_{i-1} and v_{i+1}, are not connected in G. (2) node v_i receives a "dominating" message from node v_{i+1}. It is the same with case 1; nodes v_{i-1} and v_{i+1} are connected, This contradicts to the assumption that the two adjacent vertices of v_i, v_{i-1} and v_{i+1}, are not connected in G. (3) node v_i receives a "dominating" message from a BLANK node w, which is a neighbor of v_i and different from v_{i-1} and v_{i+1}, then w is connected to v_{i-1} and v_{i+1}, since $N(v_i) \subset N(w)$ according to the marking process. Thus, $\{v, v_1, v_2, \ldots, w, \ldots v_k, u\}$ has one more BLANK nodes than $\{v, v_1, v_2, \ldots, v_k, u\}$. This contradicts to the assumption that $\{v, v_1, v_2, \ldots, v_k, u\}$ is a shortest path with the largest number of BLANK nodes between vertices v and u in G. □

Vertices in a dominating set are called dominating nodes and vertices outside a dominating set are called dominated nodes. Consider a vertex $v \in V$ such that $N_2(v) \cup N_3(v) \neq \emptyset$. The vertices in $N_2(v) \cup N_3(v)$ can only be dominated by either v or by vertices in $N_2(v) \cup N_3(v)$. But clearly $N(w) \subset N(v)$ for every $w \in N_2(v) \cup N_3(v)$. This shows that an optimal way to dominate $N_2(v) \cup N_3(v)$ is given by taking v into the dominating set. The marking process tries to select nodes, which can dominate as many as possible nodes, into the initial CDS, and it only requires constant round computations. The marking process is efficient and induces smaller initial CDS for the ad hoc wireless mobile network in average case. Our performance analysis and simulation results (to be discussed later) confirm this observation.

6.3 Localized Connected Dominating Set Reduction

All dominating nodes derived from marking process form the initial CDS. Consider a random vertex $v \in V$ such that $N_2(v) \cup N_3(v) \neq \emptyset$. We take v into the dominating set. For every $w \in N_2(v) \cup N_3(v)$, w can be covered by vertex v. Actually, both vertex v and vertices in $N_1(v)$, which can have connection with vertices outside the neighborhood of vertex v. Actually, our marking process already includes the case illustrated as the extended Rule 1 in Wu and Li's algorithm [105]. However, there maybe multiple connections available among vertices in $\{v\} \cup N_1(v)$, it is not necessary to keep all of them. We also adopt the extended Rule 2 in [105] to reduce the size of the connected dominating set. The idea is as following: if a vertex is covered by two connected vertices, removing this vertex from V' will not compromise its functionality as a CDS. To avoid simultaneous removal of three vertices covering by each other, a vertex is removed only if it is covered by other two vertices with higher ID's. Node ID(v) of each vertex $v \in V$ of each vertex serves as a priority. Nodes with higher priorities have high probability of becoming dominating nodes. We revise the extended Rule 2 suited to our situation as follows.

Rule 2: *Assume u and w are two BLANK neighbors of BLANK vertex v in G'. If $N(v) \subseteq N(u) \cup N(w)$ in G and $ID(v) = \min\{ID(v), ID(u), ID(w)\}$, then change color of v to WHITE.*

It is easy to prove $G' - \{v\}$ is still a connected dominating set. Obviously, an additional step needs to be added at the end of our marking process: if a host v is colored WHITE, it sends its status to all its neighbors.

6.4 Performance Analysis

This section discusses the efficiency and overhead of our marking process by classification of neighbors through analytical study. The performance can be measured by computation and communication complexity.

Theorem 6.3: The computation complexity of Algorithm 1 (the marking process by classification of neighbors) is $O(\Delta^2)$, where Δ is the maximum vertex degree in the network.

Proof. To carry out Algorithm 1, for each vertex v of the given graph G we have to determine the neighbor sets $N_1(v)$, $N_2(v)$, and $N_3(v)$. By definition of these sets, one easily observes that it is sufficient to consider the two-hop neighborhood information of the vertex v which is obtained by the beacon sent periodically. Firstly, we determine the vertices from $N_1(v)$ by comparing the neighbor sets of v and every neighbor u of vertex v. This step takes at most $O(\Delta^2)$, where Δ is the maximum degree of the given graph G. Then, it remains to determine the sets $N_2(v)$ and $N_3(v)$. To get $N_2(v)$, one basically has to go through all vertices from the one-hop neighbors of vertex v that are not already marked as being in $N_1(v)$ but have at least one neighbor in $N_1(v)$. All this can be done within $O(\Delta^2)$ time. Finally, $N_3(v)$ simply consist of vertices from the one-hop neighbors of vertex v that are neither marked being in $N_2(v)$ nor marked being in $N_3(v)$. This step can be done in $O(\Delta)$ time. After partitioning of vertex v's neighbors, vertex v sends out "dominating" messages, and a node receiving "dominating" just changes its color. These only take constant time. In summary, this shows the computation of Algorithm 1 can be executed in time $O(\Delta^2)$. □

Theorem 6.4: *The communication complexity of Algorithm 1 (the marking process by classification of neighbors) is $O(n\Delta)$, where Δ is the maximum vertex degree in the network and n is the number of vertices in the network.*

Proof. The communication complexity is measured as the total number of control messages sent by the network in order to construct the approximate CDS. For each vertex $v \in V$, vertex v sends the "dominating" messages if it can cover its neighbors in $N_2(v)$ and $N_3(v)$. On the other hand, if a node $u \in V$ receives a "dominating" message, it changes its color to WHITE and sends its status change to its neighbors. We see for each vertex it sends at most $O(\Delta)$ messages. Therefore, the communication complexity of Algorithm 1 (the marking process) is $O(n\Delta)$. □

Clearly, our approach is as simple as Wu and Li's algorithm in all measurements, in particular, the number of rounds needed. Note that the number of rounds is an important metric measuring the performance of the algorithm because the topology of the ad hoc wireless network changes frequently with the movements of mobile hosts, therefore the dominating set has to be updated and recalculated frequently.

Another important measurement is the size of the dominating set generated. We can not theoretically prove that our approach generates smaller connected dominating set than Wu and Li's algorithm. However, we can show that our approach outperforms Wu and Li's algorithm on average through simulation discussed in the following section.

6.5 Experimental Results

In this section we conducted the simulation study which computes the average size of the CDS derived from our algorithm with those from several existing algorithms under different conditions. The smaller the size of the dominating set, the better the results.

In our simulation environments, random graphs are generated in 600×600 square units of a 2-D simulation area, by randomly inducing a certain number of mobile nodes. We assume that each mobile node has the same transmission range r, thus the generated graph is undirected. If the distance between any two nodes is less than radius r, then there is a connection link between the two nodes. If generated graph is disconnected, simply discard the graph. Otherwise continue the simulation.

Note that, for a constant r, the network density, in terms of the average vertex degree d, will increase rapidly as the network size (n) increases. Simulation is carried out by varying average degree d of the network (i.e. the average number of neighbors of a node in the network), such that the impact of network size can be observed independent of density. The transmission range can be set as a function of d, number of nodes n, and the network area using relation $r^2 = (d * 600 * 600)\ (n-1)$. In order to observe the impact of density, each simulation is repeated on various average vertex degrees (d = 6, 12, 18, 24, 30). All simulations are conducted in static ad hoc networks, where a simulation completes after a CDS formation algorithm converges after several rounds of information exchanges. Since the topology of ad hoc networks change very dynamically, our simulation takes snapshots on dynamic ad hoc networks. For each average vertex degree d, the number of nodes n is varied from 20 to 200. For each n, the number of running times is 500 times.

First, the performance of our approach (MPCN plus Rule 2 reduction), in terms of the size of the resultant connected dominating set, is compared with a centralized algorithm (MCDS [36]), a cluster-based algorithm (Tree [7]), and a pure localized algorithm (Wu and Li [105]). MCDS is a very good approximation to the optimal solution. We use it as a rough estimate to the real minimal connected dominating set since the brute force method to find the optimal solution is too slow to provide the result for a large size network. Tree is actually a centralized algorithm as all clusterheads are connected to a global infrastructure (i.e. the tree) controlled from a central point (i.e., the root). Our approach, named MPCN, uses the marking process by classification of neighbors to form the initial CDS, and then reduces the size of CDS further via revised Rule 2. We assume vertex ID's are used as priority values.

Figure 6.5 shows the performance of these algorithms. In MCDS, the size of CDS is about 37 percent of the network size in sparse ($d = 6$) networks, and 15 percent in dense ($d = 30$) networks. This performance is much better than other algorithms. Tree has a performance of 55 percent in sparse networks and 26 percent in dense networks. Wu and Li's approach produces a dominating set that is about 12 percent larger than Tree in sparse networks, and about 35 percent larger in dense networks. The performance of MPCN is even better than Tree in sparse networks, but produces a dominating set that is about 2 percent larger than Tree in dense networks. MPCN is a pure localized algorithm and is actually more efficient than a cluster-based algorithm. We can see MPCN reduces the size of the dominating set by about 10 percent more nodes in sparse networks, and about 20 percent more nodes in dense networks compared with another pure localized algorithm, Wu and Li's approach.

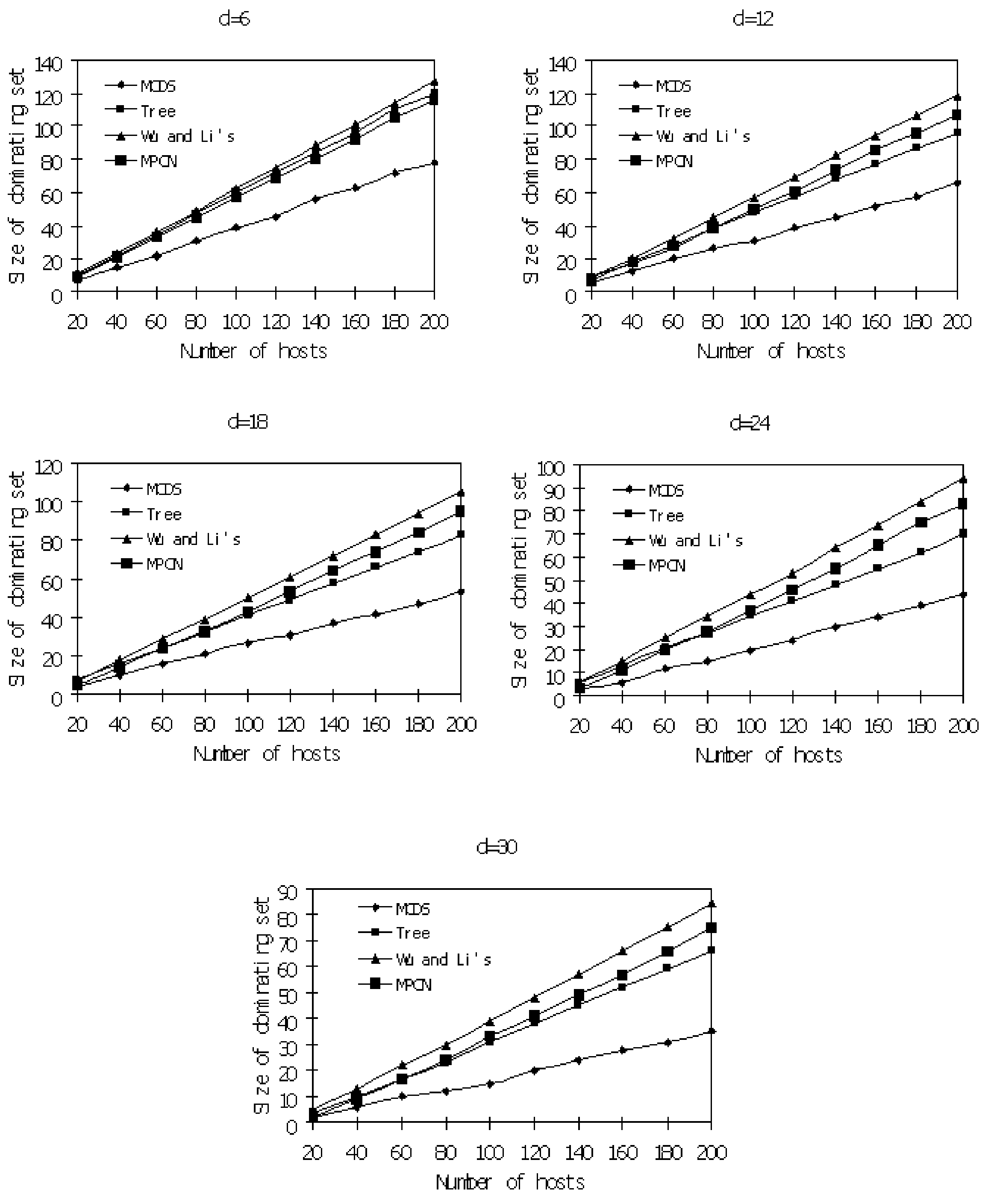


Figure 6.5. Comparison with existing algorithms

The second group of simulations compare performances of two marking process (MPCN and Wu and Li's) and two algorithms for computing CDS (MPCN + R). Figure

6.6 show all the situations in sparse and dense networks. Our marking process produces an initial dominating set that has almost the size of a dominating set produced by the Wu and Li’s algorithm after two rounds of redundancy reductions in sparse networks. Since Wu and Li’s marking process is very trivial, it produces an initial dominating set that almost has the same size with the whole network. That is, almost every host belongs to a dominating set. Our marking process is about 25 percent smaller than Wu and Li’s marking process in dense networks, about 30 percent smaller than Wu and Li’s marking process in sparse networks. From Table 6.1, we see that when the number of nodes in the network is small, only our marking approach without any redundancy reduction has a very comparable performance with Wu and Li’s algorithm including two rounds of redundancy reduction. This is because our marking process prefers to select optimal hosts, which can cover as many as possible nodes, into the initial connecting dominating set. This works especially in sparse or small size networks.

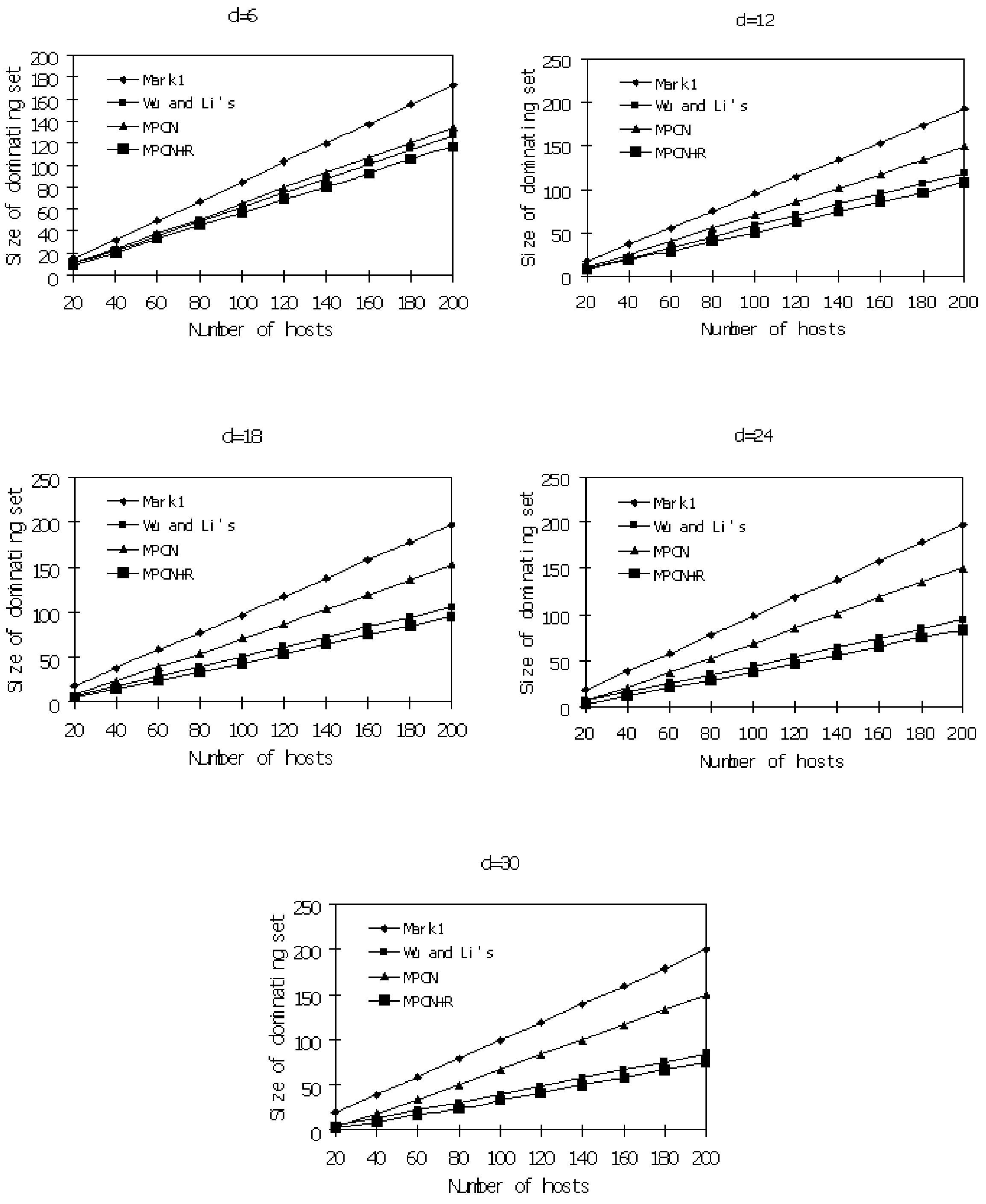


Figure 6.6. Comparison of different marking processes

Table 6.1 The comparison of different marking processes when N = 20.

Degree \ Size	Mark1	Wu and Li's	MPCN	MPCN+R
6	**15**	**11**	**11**	**9**
12	**17**	**9**	**10**	**7**
18	**18**	**7**	**8**	**5**
24	**18**	**6**	**6**	**3**
30	**19**	**5**	**4**	**2**

Simulation results can be summarized as follows:

1. The connected dominating set produced by our marking process is about the same size as those produced by the cluster-based schemes in dense networks, and smaller than those in sparse networks. This is achieved in a pure localized way without sequential propagation.
2. Our algorithm performs better than another pure localized algorithm, Wu and Li's algorithm, with lower cost and a faster converging speed, since our algorithm has less number of rounds for computing CDS than that of Wu and Li's algorithm.
3. Only our marking process without any redundancy reduction is comparable to Wu and Li's algorithm with two rules reduction in sparse or small size networks.

CHAPTER 7

COMPUTING CONNECTED DOMINATING SET BY ITERATIONS

7.1 Iterative Localized Algorithm for Connected Dominating Set

In this chapter, we propose an iterative localized algorithm to construct CDS [65]. This algorithm consists of three main phases: Dominators Election, Redundancy Elimination By One Neighbor and Redundancy Elimination By Two Neighbors. First, we form the initial CDS, *U*, which includes all the nodes once they have two unconnected neighbors. Then a node *u* is removed from the initial CDS, *U*, if there exists a neighbor of node *u* in *U* that can cover all other neighbors of *u*. In the third phase, we eliminate a redundant node *u* from *U* when node *u* has any two neighbors in *U* that can dominate all the neighbors of *u*. We will show that our algorithm is not only correct but also message and time efficient through proof.

7.1.1 Preliminaries

Wu and Li [105] proposed a simple and efficient distributed algorithm that can quickly find a DS in a mobile ad-hoc network. Each node is marked as white initially. Let N(v) be the open neighbor set of vertex v, which means N(v) includes all the neighbors of vertex v. Let N[v] be the closed neighbor set of vertex v, the set of all neighbors and itself. By assumption, each node has a unique ID number. This algorithm runs in two phases. In the first phase, each node broadcasts its neighbor set N(v) to all its neighbors, and after collecting all adjacency information from all neighbors every node marks itself as black if there exist two unconnected neighbors. All black nodes form the initial CDS.

Wu and Li introduced the concept of a gateway node. A node is a gateway node if it belongs to the DS. However, considering only the first phase, there are too many gateway nodes. So in the second phase, the algorithm executes two extensional rules to eliminate local redundancy. Extensional rule 1 is as follows: Consider any two nodes *u* and *v* belonging to the DS. If $N[v] \subseteq N[u]$ and $ID(v) < ID(u)$, then change *v*'s color to white. That means if all neighbors of *v* and itself are covered by *u*, and *v* is connected to *u* and has lower ID, *v* can be removed from the DS. Rule 2 is described as follows: Consider any three nodes *u*, *v*, and *w* belonging to a DS, such that *u* and *w* are two black neighbors of *v*. If $N(v) \subseteq N(u) \cup N(w)$ and *v* has the smallest ID of three nodes, then *v*'s color is changed to white. In other words, if each neighbor of *v* is covered by *u* and *w* together, where *u* and *w* are both connected neighbors of *v*, then *v* can be eliminated from the list of gateway nodes. Thus, the second phase removes some nodes from the original DS and the size of a DS is further reduced. Wu and Li provided simulation results to show that the cardinality of a DS is largely reduced.

Figure 7.1 shows the results of Wu and Li's algorithm using two phases and the two extensional rules. There are 10 nodes randomly located in 2-D space. Some nodes are connected if they are within the transmission range. Nodes are marked as black if the node has any two neighboring nodes that are not connected. For example, node 0 has neighbor nodes 6 and 9 that are not connected, therefore node 0 is marked black by the basic rule. In the figure, only node 3 is not marked because all of its neighbors are connected. (The neighbors of node 3 are node 1, node 4, and node 9; there are lines between 1 and 4, 4 and 9, 1 and 9.) Applying extension rule 1, nodes 0, 6, 4, 1, 5 are unmarked as shown by a black line. For example: $N[0] = \{0,2,6,9\}$, $N[2] = \{0,2,6,8,9\}$,

$N[0] \subseteq N[2]$ and ID(0) < ID(2), therefore, node 0 can be unmarked by extensional rule 1. $N[6] = \{0,2,6,7,8\}$, $N[8] = \{0,2,6,7,8\}$, $N[6] \subseteq N[8]$ and ID (6) < ID (8), therefore, node 6 also can be unmarked by extensional rule 1. Applying extensional rule 2, nodes 0,1,4,5 are unmarked as shown by black cross. For example: $N(0) = \{2,6,9\}$ and $N(2) = \{0,6,8,9\}$, $N(9) = \{0,2,7,1,3,4,1\}$. $N(0) \subseteq N(2) \cup N(9)$ and ID(0)=min{ ID(0), ID(2), ID(9)}. Node 0 can be unmarked by extensional rule 2. There is an overlap between extensional rule 1 and 2. Many nodes are unmarked by extensional rule 1, but they can also be unmarked by extensional rule 2. For example, node 0 can be unmarked by both extensional rule 1 and 2.

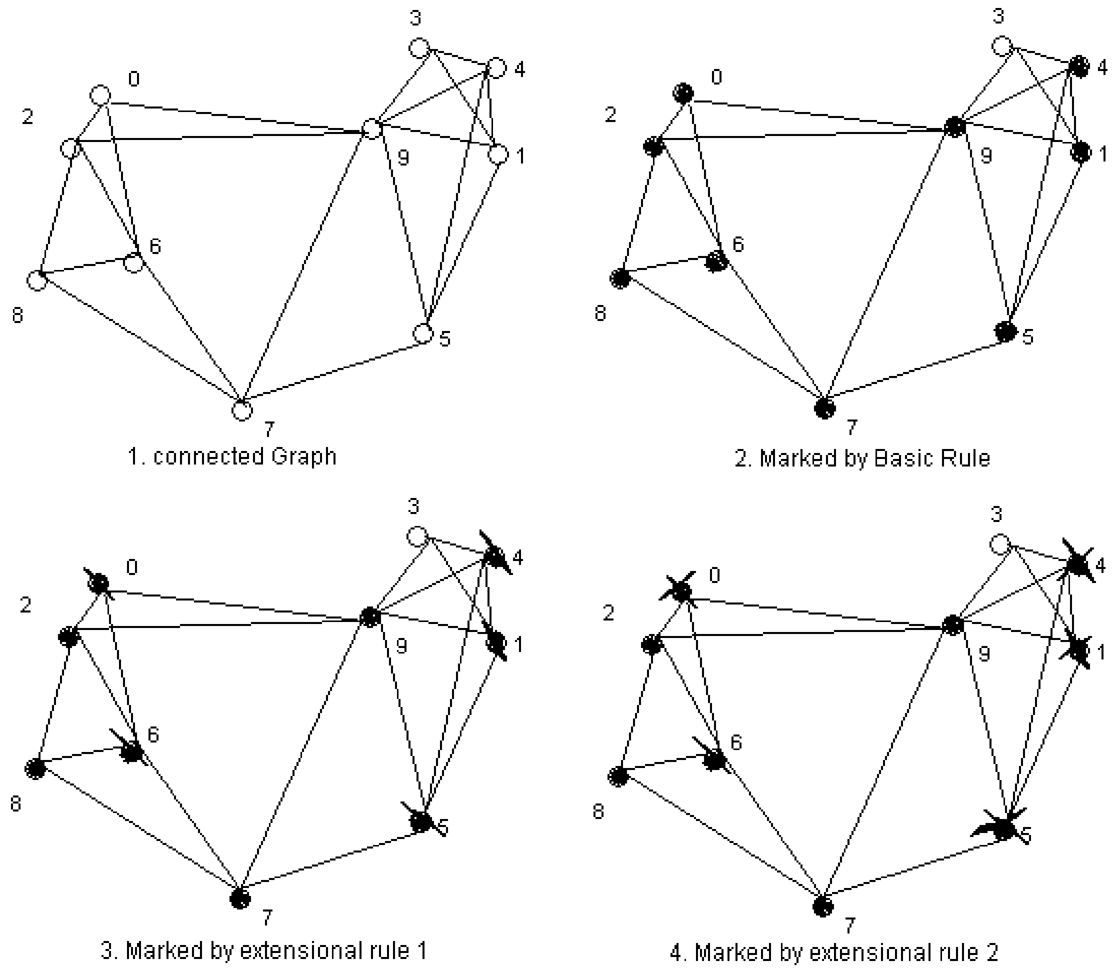

1. connected Graph
2. Marked by Basic Rule
3. Marked by extensional rule 1
4. Marked by extensional rule 2

Figure 7.1. Examples of Wu and Li's algorithm

In this algorithm, each node only needs to know local information and 2-hop neighborhood information. On the other hand it is proven in [105] that the black vertices form a DS of unit disk graph G. At the same time, G', induced from the DS is connected and includes all the intermediate nodes of any shortest path between two vertices in G. The paper claims that the cost of computation at each vertex is $O(\Delta^2)$, where Δ represents the maximum node degree of graph G. Also, the total amount of message exchange is $O(n\Delta)$, where n is the cardinality of the vertex set of graph G. This algorithm needs a fixed number of rounds. By introducing dominating sets, the routing process in a mobile ad-hoc network can be simplified.

7.1.2 Improved Algorithm for Computing CDS by Iterations

Initially each mobile host is colored white. Dominators will be colored black and form the CDS when the algorithm terminates. We assume that each host has a unique ID, and each vertex knows its one-hop neighbors and its degree d that can be collected by periodic or event-driven "hello" messages. Messages are used to exchange information for computation or control. Actually messages record the information of nodes that send them. Each message contains three fields. The first field contains a unique identifier ID; the second field contains a status number, i, which is used to specify what jobs a host has finished and will do next; and the last one is a set of all one-hop neighbors represented by NEIGHBOR. The algorithm proceeds in phases. At each phase, some of the hosts are active and generate messages. Each host x first broadcasts message (x.ID, 1, x.NEIGHBOR) to the hosts at one-hop distance. In our algorithm, after a destination host y collects messages from all its neighboring hosts such as x_1, x_2, x_3, … x_d, it performs some actions according to the status number i, its own ID and IDs of all its neighbors. We set a TIMEOUT value; a neighboring node is considered passive, if *y* does not receive any message from this node after this threshold time. We also induce a new concept "Synchronization Phase" to our algorithm. Synchronization Phase lies between two continuous phases. In Synchronization phase, destination node *y* collects messages from its neighbors until it has received messages sent by all its neighbors who have finished jobs of previous phase and will start jobs of next phase. Notice here neighboring nodes of *y* can only be classified into two types. Some neighboring nodes are passive, either *y* has received PASSIVE messages or after the threshold time passes, *y* has not received anything from them, and all other nodes have the status numbers to identify of what phase jobs will begin next. The following are the details of this algorithm:

Phase 1: If *y* is white and *i* equals 1, then it checks whether any two of its neighbors are connected. Since the information of *y*'s two-hops neighbors can be obtained through the third field of messages, *y* compares the neighbor sets of its two neighbors to see whether there exist any overlaps. As we know, if such overlaps exist, then these two neighbors are connected. Otherwise, they are unconnected. Once *y* finds two of its neighbors are unconnected, *y* is colored black, declares itself as a dominator and broadcasts message (y.ID, 2.1, *y*.NEIGHBOR) to all *y*'s neighbors. If *y* does not have unconnected neighbors, it remains white and broadcasts message (*y*.ID, PASSIVE). Phase 1 terminates when every host finishes this kind of judge about coloring.

Phase 2.1:

```
Synchronization Phase of y, status number equals to 2.1.
For each active neighbor $x_k$ do:
   if (color of y is black and y.ID < $x_k$.ID) then
      if ($x_k$ dominates y's neighbors and y itself) then
         y is colored white, removed from CDS and broadcasts
```

```
            message ($y$.ID, PASSIVE)
            break;
         end if
    end for
    $y$ sends message ($y$.ID, 2.2, $y$. NEIGHBOR) to all $y$’s
    neighbors.
```
Phase 2.2:
```
  Synchronization Phase of $y$, status number equals to 2.2.
  For each active neighbor $x_k$ do:
       if (color of $y$ is black and $y$.ID > $x_k$.ID) then
         if ($x_k$ dominates $y$’s neighbors and $y$ itself) then
            $y$ changes its color to white and is removed from CDS
            break;
         end if
    end for
    $y$ broadcasts message ($y$.ID, 3.1, $y$. NEIGHBOR).
```
Phase 3.1:
```
    Synchronization Phase of $y$, status number equals to 3.1.
    For any two active neighbors $x_k$ and $x_j$ do:
       if (color of $y$ is black and $y$.ID < $x_k$.ID and $y$.ID < $x_j$.ID)
       then
         if ($x_k$ and $x_j$ combine to dominate $y$’s neighbors and $y$
         itself) then
            $y$ changes its color to white and is removed from
            CDS
            break;
         end if
    end for
    $y$ sends message ($y$.ID, 3.2, $y$. NEIGHBOR) to all $y$’s
    neighbors.
```
Phase 3.2:
```
  Synchronization Phase of $y$, status number equals to 3.2.
  For any two active neighbors $x_k$ and $x_j$ do:
     if (color of $y$ is black and $y$.ID is larger than one of
     $x_k$ and $x_j$, but less than one of  them) then
       if ($x_k$ and $x_j$  combine to dominate $y$'s neighbors and $y$
       itself) then
          $y$ changes its color to white and is removed from
          CDS
          break;
       end if
  end for
  $y$ broadcasts message ($y$.ID, 3.3, $y$. NEIGHBOR).
```

Phase 3.3:

```
  Synchronization Phase of $y$, status number equals to 3.3.
  For any two neighbors $x_k$ and $x_j$ do:
      if (color of $y$ is black and $y$.ID is largest one among
      $x_k$.ID, $x_j$.ID and $y$.ID) then
        if ($x_k$ and $x_j$  combine to dominate $y$'s neighbors and $y$
        itself) then
           $y$ changes its color to white and is removed from
           CDS
           break;
        end if
   end for
   $y$ becomes passive.
```

When a black host finishes the jobs of phase 3.3 and becomes passive, it marks its local predicate true and propagates a token to detect termination of our algorithm as that described by Zou's algorithm [110]. Then a CDS is constructed by all the nodes remaining black, when the algorithm terminates.

7.2 Correctness and Complexity Analysis

We use a simple graph G (V, E) to represent a wireless ad hoc network, where V represents a set of mobile hosts and E denotes a set of edges. There exists an edge (u,v) in

E if nodes *u* and *v* are neighbors in a wireless ad hoc network. Assume *V*' is the set of black nodes in *V*, and *G*' is the subgraph induced by *V*'. The next theorem shows that *G*' is a connected dominating set.

Theorem 7.1: *If the given graph G = (V, E) is not a complete graph, the graph G' induced by V', derived from our proposed approach, forms a connected dominating set.*

Proof. It has been shown in [105] that nodes derived from phase 1, coloring process, form a CDS. We only need to show that whenever a node *v* is removed either by one neighbor or two neighbors, the remaining nodes (*G*'- {*v*}) still form a CDS. We look at the first case in which a redundant node *v* is removed from the dominating set by one of its neighbors. There is a requirement for removing such *v* that all neighbors of *v* and *v* itself must be covered by one of *v*'s neighbors in the dominating set, and without loss of generality we assume it is node *u* that makes *v* removed. Removing *v* only affects the neighbor nodes of *v* and itself. Since *u* and *v* are neighbors and *u* remains in CDS, *v* is adjacent to CDS. Also all neighbors of *v* are dominated by *u*, so all neighbors of *v* are adjacent to *u*, a dominator. Then in the second case, if *v* is removed by two of its neighbors in the dominating set, *u* and *w*, *u* and *w* combine together to dominate all neighbors of node *v*. Obviously *v* is adjacent to CDS, for *v* is a neighbor of both *u* and *w*. While all neighbors of *v* are adjacent to CDS because they are covered by either *u* or *w*. And it is easy to see *G*'- {*v*} in either case is still connected. In other word, the graph *G*' induced by *V*', derived from our proposed approach, forms a CDS. □

In Wu and Li's algorithm, it mentions that the role of ID is to avoid "illegal simultaneous" removal of vertices in *G*'. Vertex *v* cannot be removed even if $N[v] \subseteq N[u]$ unless ID(*v*) < ID(*u*). And *v* cannot be removed even if $N(v) \subseteq N(u) \cup N(w)$ unless *v*'s ID is the smallest one among v, u and w. This kind of avoidance wastes a lot of chances to reduce the cardinality of CDS and is too conservative. Actually by our approach, we can remove v only if $N[v] \subseteq N[u]$ regardless of v and u's IDs. We also can remove v only if $N(v) \subseteq N(u) \cup N(w)$ without considering the order of u, v and w's IDs. At the same time, "illegal simultaneous" removal of vertices in G' is avoided by control messages in which there exist status numbers to identify the end of previous phase and beginning of next phase. In synchronization phase, a node waits for information of all neighbors. This helps avoid illegal simultaneous removal.

In phase 1, each host only broadcasts messages (ID, 1, NEIGHBOR) at most once. The message complexity is dominated by nodes' degree. Thus, the message complexity of Phase 1 is $O(\Delta n)$. Each host needs to compare the neighbor sets of its any two neighbors, and it takes $O(\Delta^2)$ time. The time complexity is $O(\Delta^2)$. In every synchronization phase, hosts only wait for receiving messages from all their neighbors, so they do not have any computation or communication jobs. In phases 2.1 and 2.2, each host checks all its black neighbors one by one, thus the time complexity is $O(\Delta)$. The message complexity is also $O(\Delta n)$ since each active host needs to broadcast message either (ID, PASSIVE) or "phase 3.1 begins". Each node checks any two of its black neighbors one pair by one pair in phases 3.1, 3.2 and 3.3, so the time complexity is $O(\Delta^2)$. The message complexity remains $O(\Delta n)$. From the above analysis we have the following theorem:

Theorem 7.2: *The distributed algorithm has time complexity $O(\Delta^2)$ and message complexity $O(\Delta n)$.*

Theorem 7.3: *Our distributed algorithm for finding CDS is deadlock-free.*

Proof. To show that our algorithm for finding CDS is deadlock-free, we need to prove that there is no mutual waiting among mobile hosts, because mutual waiting is necessary condition of the deadlock situation in message communication. Mutual waiting occurs in message communication when each of a group of hosts is waiting for a message from another member of the group, but there is no message in transit. By the definition of synchronization phase, it is easy to see that there is no mutual waiting in the network, since we set timeout value, if one host has some problem to finish jobs or send messages, other hosts who wish to exchange messages with it will regard that host as PASSIVE without infinite waiting. Thus, no mutual waiting can be created. □

7.3 Simulation Results Analysis

We conducted a simulation study to measure the size of the CDS derived from our iterative algorithm and compared it with several existing algorithms. In our simulation environments, random graphs are generated in a 600×600 square units of a 2-D simulation area, by randomly including a certain number of mobile nodes. We assume that each mobile node has the same transmission range r, thus the generated graph is undirected. If the distance between any two nodes is less than radius r, then there is a connection link between the two nodes.

Note that, for a constant r, the network density, in terms of the average vertex degree d, will increase rapidly as the network size (n) increases. Simulation is carried out by varying average degree d of the network (i.e. the average number of neighbors of a node in the network), such that the impact of network size can be observed independent of density. The transmission range can be set as a function of d, number of nodes n, and the network area using relation $r^2 = (d * 600 * 600) / \pi * (n-1)$. In order to observe the impact of density, each simulation is repeated on various average vertex degrees ($d = 6$, 12, 18, 24). All simulations are conducted in static ad hoc networks, where a simulation completes after a CDS formation algorithm converges after several rounds of information exchanges. For each average vertex degree d, the number of nodes n is varied from 20 to 200. For each n, the number of running times is 500 times.

First, the performance of our iterative localized algorithm, in terms of the size of the resultant connected dominating set, is compared with a centralized algorithm (MCDS [36]), a cluster-based algorithm (Tree [7]), and a pure localized algorithm (Wu and Li [105]). MCDS is a very good approximation to the optimal solution. We use it as a rough estimation to the real minimal connected dominating set since the brute force method to find the optimal solution is too slow to provide the result for large size networks. Tree is actually a centralized algorithm as all clusterheads are connected to a global infrastructure (i.e. the tree) controlled from a central point (i.e. the root). Our iterative localized algorithm largely reduces the redundant nodes in CDS, and at the same then avoids the "illegal simultaneous" removal of dominating nodes via phase by phase operations. *MCDS*, *Tree*, *Wu and Li's*, and *Iteration* are four parameters used to represent the number of dominators calculated by the above four approaches. It is well known that the lower the number of dominators, the better the result. Thus, our goal is to generate a small connected dominating set to facilitate a fast routing process and reduce access and update overhead.

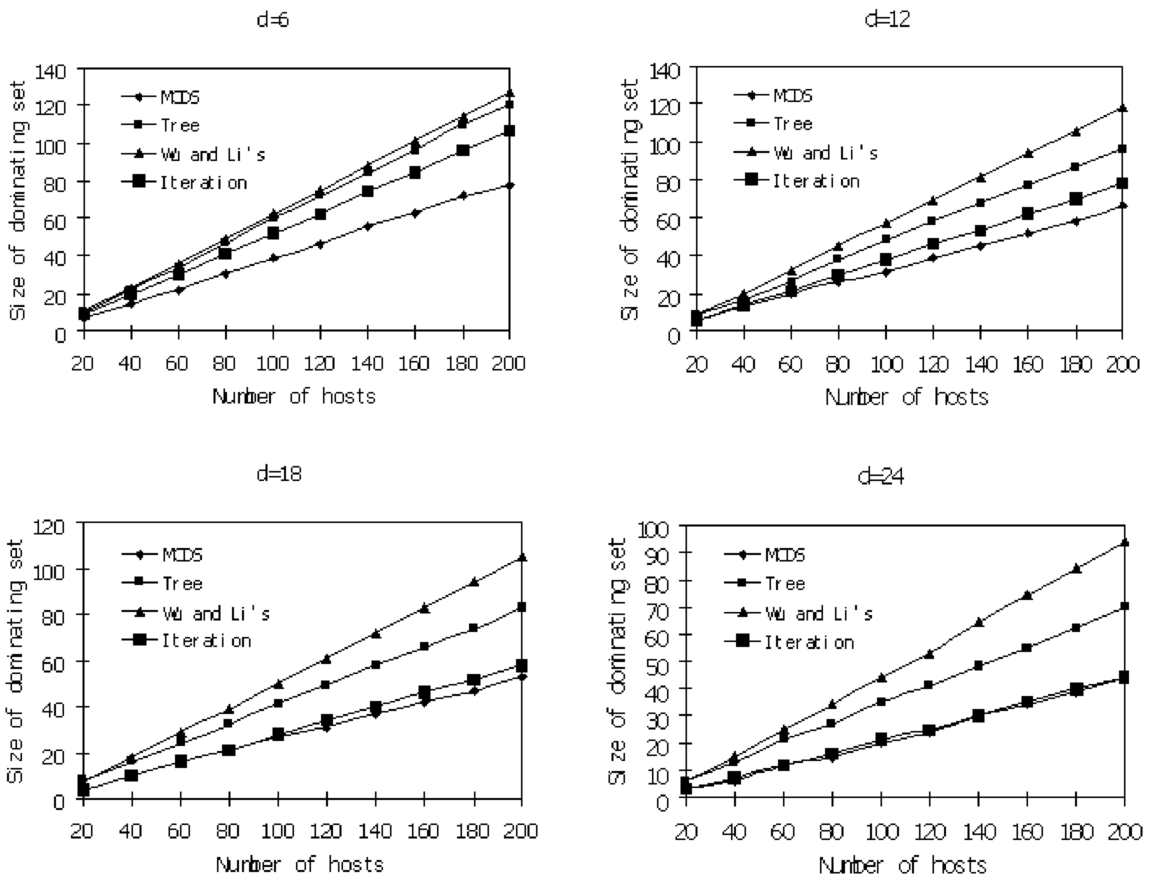


Figure 7.2. Comparison with existing algorithms

Figure 7.2 shows the performance of these algorithms. In MCDS, the size of CDS is about 37 percent of the network size in sparse (d = 6) networks, and 21 percent in dense (d = 24) networks. This performance is much better other algorithms, especially Tree, and Wu and Li's. Tree has a performance of 55 percent in sparse networks and 34 percent in dense networks. Wu and Li's approach produces a dominating set that is about 12 percent larger than Tree in sparse networks, and about 40 larger in dense networks. Our iterative localized algorithm always outperforms Tree, and Wu and Li's algorithm. Our approach produces a dominating tree that is about 10 percent larger than MCDS in sparse networks, however, its performance matches that of MCDS in dense networks. We can see the gap between *MCDS* and *Iteration* decreases as the network becomes dense.

Since there are more chances for a node to find any one or two neighboring dominators that cover its neighbors in dense networks. Our approach is more efficient because it is pure localized without any central administration and global information.

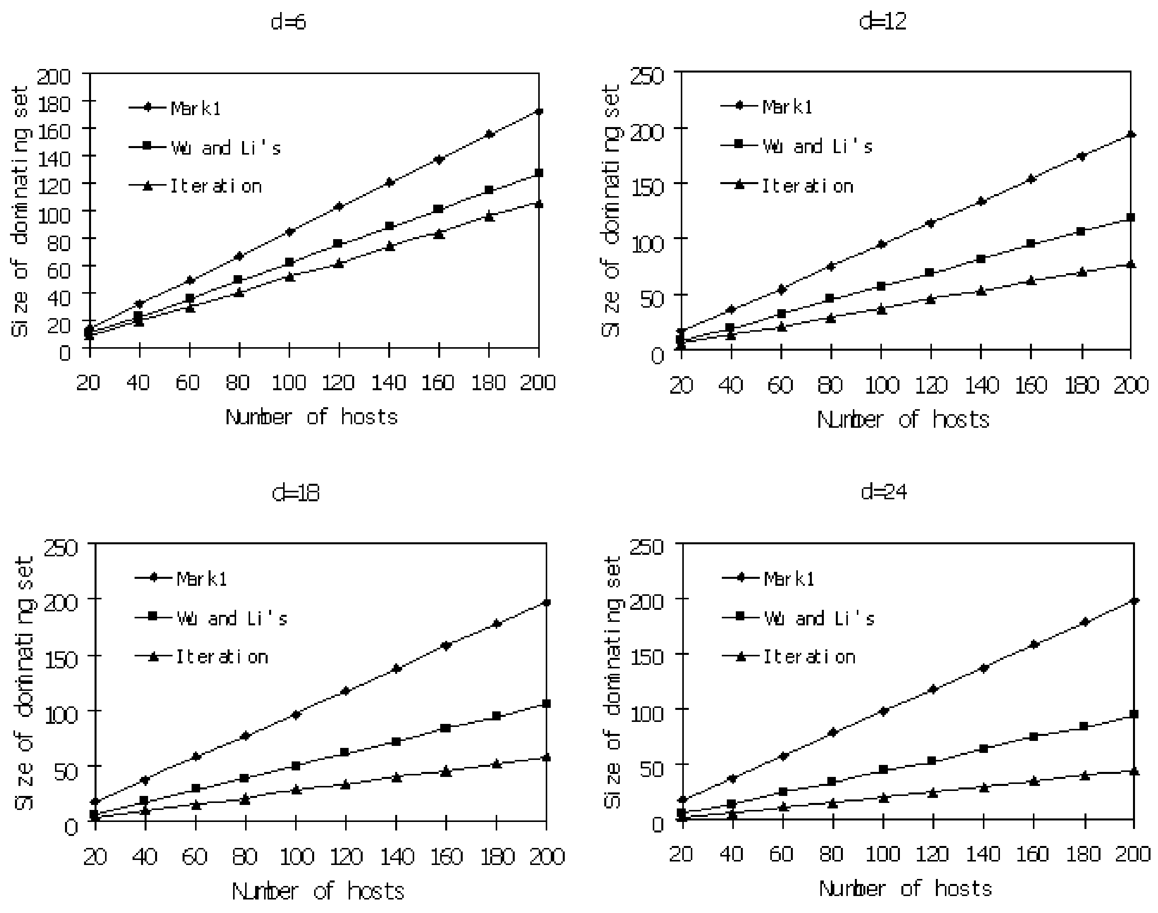


Figure 7.3. Comparison of different pruning methods

The second group of simulations compare the performances of two different dominating pruning rule, of which one includes the extend rule 1 and rule 2 proposed in [105], and the other is our iterative pruning rules. Figure 7.3 shows the average number of dominating nodes in networks with different densities. We can see the performance of the basic rule without extensional rules is very poor and the ratio of nodes in the dominating set to all nodes in the network is almost 1, specifically, 0.86. Moreover, almost every node belongs to a dominating set in dense networks. Because the basic rule is loose, when the density of generated graphs increases it is very easy to find two neighbors of a node that are not connected. After applying the two extensional rules of Wu and Li's algorithm, the size of the dominating set is largely reduced. The ratio of the number of dominating nodes over the total number of nodes in the network changes from 60% to 22% with the increase of the network density. The simulation also shows that our pruning rules consistently outperform Wu and Li's algorithm with two extensional rules. We can see the gap between our approach and Wu and Li's increases as the density of network increases. The ratio of dominating set size induced by our algorithm over the number of nodes changes from 41% to 8% when networks become dense. The number of dominating nodes produced by our algorithm is only half or one third of that derived by Wu and Li's algorithm.

Simulation results can be summarized as follows:

1. The connected dominating set produced by our iterative localized algorithm is about the same size as those produced by MCDS in dense networks, and 10 percent larger than those in sparse networks. This is achieved in a pure localized way without central administration.
2. Our algorithm performs better than another pure localized algorithm, Wu and Li's algorithm.

CHAPTER 8

GENETIC FUZZY MULTI-PATH ROUTING PROTOCOL

8.1 Background on Multi-Path Routing

Multipath routing has been explored in several different contexts [73]. Traditional circuit switched telephone networks used a type of multipath routing called alternate path routing. In alternate path routing, each source node and destination node have a set of paths (or multipaths) which consist of a primary path and one or more alternate paths. Alternate path routing was proposed in order to decrease the call blocking probability and increase overall network utilization. In alternate path routing, the shortest path between exchanges is typically one hop across the backbone network; the network core consists of a fully connected set of switches. When the shortest path for a particular source destination pair becomes unavailable (due to either link failure or full capacity), rather than blocking a connection, an alternate path, which is typically two hops, is used. Well known alternate path routing schemes such as Dynamic Nonhierachical Routing and Dynamic Alternative Routing are proposed and evaluated in [10] and [34] respectively.

Multipath routing has also been addressed in data networks which are intended to support connection-oriented service with QoS. For instance, Asynchronous Transfer Mode (ATM) [1] networks use a signaling protocol, Private Network-Node Interface (PNNI), to set up multiple paths between a source node and a destination node. The primary (or optimal) path is used until it either fails or becomes over-utilized, then alternate paths are tried. Using a crankback process, the alternate routes are attempted until a connection is completed.

Alternate or multipath routing has typically lent itself to be of more obvious use to connection-oriented networks; call blocking probability is only relevant to connection oriented networks. However, in packet-oriented networks, like the Internet, multipath routing could be used to alleviate congestion by routing packets from highly utilized links to links which are less highly utilized. The drawback of this approach is that the cost of storing extra routes at each router usually precludes the use of multipath routing. However, multipath routing techniques have been proposed for Open Shortest Path First (OSPF) [72], a widely used Internet routing protocol.

8.2 Multipath Routing for MANET

Over the last few years, many reactive routing protocols for MANET have been proposed [45, 51, 81]. However, most of them use a single path. The topology of MANET may change rapidly and unpredictably over time, which can cause the single path to be easily broken. When the single path fails, these protocols need to process a potentially costly route discovery to locate an alternate route for the given destination. Therefore, single path routing protocols directly expose nodes to long network delays and excessive overhead when the path fails. Single path routing has not considered the load balancing problems. An unbalanced assignment of data traffic leads to power deletion on heavily loaded hosts. With more hosts powered down, the connectivity of the network will be reduced, which lead to the failure of communications due to network partition. Multi-path reactive protocols [56, 75, 90, 102] are designed to alleviate these problems by selecting a set of redundant paths between the source and the destination in a single route discovery attempt. In [75], Nasipuri and Das prove that the use of multiple paths could keep correct end-to-end transmission for a longer time than a single path. In other words, a new routing discovery process is required only when all paths fail, thus, the frequency of searching for new routes is much lower if a node maintains multiple paths to the destination. The lower frequency of the routing rediscovery process means lower route discovery latency and less routing overhead.

The multipath routing has a lot of advantages. Firstly, it promises load balancing. Multipath routing distributes traffic over multiple routes, which can alleviate congestion. There are different ways to allocate traffic. For example, a per-connection granularity would allocate all traffic for one connection to a single path; a per-packet granularity would distribute the packets from multiple connections amongst the paths, which may need packet reordering in the destination node. Secondly, multipath routing provides fault-tolerance; When a link breaks, alternative routes still can be used to route the packets. A node disjoint route has a higher degree of fault-tolerance than a link disjoint and non-disjoint route, but it does not ensure transmission independence. Then, multipath routing has higher aggregate bandwidth: multiple paths can be used simultaneously to route data packets.

However, current multi-path protocols choose a set of multiple redundant paths through a single route selection parameter without considering the interplays of different selection parameters. For example, Split Multi-path Routing protocol (SMR) [56] uses the number of intermediate hops as the selection parameter. SMR chooses the shortest routing path as the primary route, and then computes the maximum disjointed path as the secondary route. The Stability-Based Multi-path Routing algorithm (SBMR) [90] also locates a set of totally independent multiple paths via the hops of the routes. Due to the special characteristics of MANET, the shortest path decided by the number of intermediate hops is not necessarily the most reliable or durable path since the topology of MANET is determined by many factors such as battery capacity, traffic pattern, link stability and nodal mobility. All of these factors are correlated. Thus, consideration of only one or two factors is not sufficient for choosing an optimal set of multiple paths.

This chapter addresses multiple path selection by proposing a simple and effective protocol called Genetic Fuzzy Multi-path Routing Protocol (GFMRP) [63], which considers the multiple correlated selection parameters, based on fuzzy set theory [86] and evolutionary computing [11]. The goal of GFMRP is for selecting a set of paths to maximize network lifetime and reliability. Fuzzy set theory is a promising tool for many design and control problems in telecommunication systems since there are so many uncertain factors in networking environments. The peculiarities of MANET bring more uncertain factors under which MANET routing decisions are made, i.e. latency, unknown node mobility leading to unpredictable topology changes, and inaccurate network state known by each node [69]. In this chapter, we will investigate four important parameters: (1) packet buffer occupancy rate at a node, (2) energy consumption rate at a node, (3) link stability between neighboring nodes, and (4) the number of intermediate hops in a route. Since every node can be a destination of packets delivery, GFMRP embeds a fuzzy inference system into every node. During routing discovery phase, the fuzzy inference system in the destination node takes these four selection parameters (the information of these parameters are collected through RREQ packet) as input and outputs a crisp rank to evaluate every potential route path between the source node and the destination node.

Then we classify the route paths into two categories according to their relative ranks. A path having a rank value above a certain threshold, α, belongs to a reliable set of multiple paths. Otherwise it belongs to an unreliable set. The routing paths in the reliable set are used by GFMRP as candidates for multi-path routing. GFMRP employs a simple load-balancing method to distribute the traffic load, by spreading data packets over all the paths of the reliable multiple paths set in a round-robin fashion. The performance of GFMRP is evaluated in terms of packet delivery ratio, average end-to-end delay, and the frequency of route rediscovery in Network Simulator (*ns2*) context. Simulation results demonstrate that GFMRP can be more adaptive to the ad hoc environment and outperforms DSR [45], SMR [56] and SBMR [90].

8.3 Related Works

In this section, we briefly describe two relevant fuzzy decision based routing protocols in MANET, which were recently proposed.

8.3.1 The Evolutionary Ad-hoc On-demand Fuzzy Routing (E-AOFR)

Evolutionary ad-hoc on-demand fuzzy routing (E-AOFR) is proposed in [69]. E-AOFR models the uncertainty in MANET by fuzzy set theory. It incorporates a fuzzy logic function into every mobile node, which measures three parameters at a node (remaining battery capacity, buffer length, link stability), taking these three parameters as input and producing a single cost metric. During the route discovery phase, the source node sends route discovery (RREQ) packets that collect the sum of the fuzzy costs for all the individual links along the path. When a node receives RREQ packets, it calculates its fuzzy cost of participating in the route, and then adds its cost to the previous cost of RREQ packets. The destination node waits a certain amount of time to collect the cost information of all possible routes, and then chooses the one with the least cost as the routing path between the source and destination, and then sends back its RREP. We can see that E-AOFR is a single-path source routing protocol without redundant paths for routing. In E-AOFR, every node needs to compute its fuzzy cost as long as it receives RREQ packets between any two pair of nodes. The frequency of this computation is very high, placing a heavy burden on the node. Moreover, their selection parameters are not very suitable since they do not consider the interplay of nodes on the whole path.

8.3.2 Fuzzy Logic Wireless Load Aware Multi-Path Routing (FLWLAMR)

Fuzzy logic wireless load aware multi-path routing protocol is described in [4]. FLWLAMR uses a similar method as SMR does for route discovery. FLWLAMR also chooses the route with the least delay as the primary route for delivering packets between the source node and the destination node, the second route is the path which is the maximally disjointed path with the primary one and has the shortest distance. However, in FLWLAMR the source node uses a fuzzy control system, of which the inputs are network status and the priority of data packets, and outputs are the number of paths used for routing specific data packets. Thus, the source node can differentiate resource allocation by considering traffic importance and network status. Traffic data is routed over zero or more maximally disjointed paths to the destination: important packets may be forwarded redundantly over multiple disjointed paths for guaranteed reliability, while the least important packets may be delayed at the source. The fuzzy control system of FLWLAMR is used to distinguish the wireless resource allocation, not to select an "optimal" set of multiple paths. For multi-path selection, it is almost the same as SMR which uses a crisp, simple selection parameter.

8.4 Design of Genetic Fuzzy Multi-Path Routing Protocol

In this section, we analyze the following four important parameters; packet buffer occupancy rate at a node, energy consumption rate at a node, link stability between neighboring nodes, and the number of intermediate hops in a route, and explain why they are suitable to act as input parameter for our Fuzzy Inference System. The procedure of Fuzzy Inference System is described in detail. Finally, we give a description about our multipath routing protocol based Fuzzy Inference System.

8.4.1 Multiple Selection Parameters

To the best of our knowledge, current multi-path protocols choose a set of multiple redundant paths via one or two route selection parameters without considering the correlations of the different selection parameters. However, there are lots of uncertain and varying conditions in MANET, and the topology of MANET is affected by many correlated parameters. Thus, consideration of only one or two parameters is not reasonable for determining an optimal set of multiple paths. We will investigate several parameters used to describe the network status and the mobile nodes' capacities in this section, and incorporate these route selection parameters in GFMRP in order to achieve the multi-path routing goal. The four routing parameters used are: (1) energy consumption rate at a node, (2) buffer occupancy rate at a node, (3) link stability between the neighboring nodes, and (4) the number of intermediate hops in a route.

As we know, mobile nodes in MANET have limited battery capacity. So the saving of battery power is a vital issue when determining the network route. Most energy aware routing protocols often base themselves only on the parameters related to the remaining battery capacity, which alone cannot help to establish the best route between the source and the destination nodes. Even if a node currently has enough remaining battery capacity, and it accepts all route requests, more traffic load will be injected through that node, and the battery will be consumed very quickly. The result is that the battery of the node is depleted too soon. To mitigate this problem, other parameters, based on the traffic load characteristics, could be employed as represented in [48]. GFMRP uses the energy consumption rate as the parameter to describe the battery power condition of every node N_i. We denote the battery power consumption rate as BPC_i. The value of BPC_i is evaluated in a very simple method, but has a similar principle used in [48]. Since the battery power consumption of a node is caused by the transmission, reception, and overhearing of packets activities, BPC_i is closely related to the amount of transmitted packets, received packets and overheared packets at a node and the power used for per packet transmission, reception or overhearing. So we define BPC_i as a linear function shown below:

$$BPC_i = \frac{(W_r \times M_r + W_s \times M_s + W_O \times M_O)}{T} \qquad (12)$$

where the W_r, W_s, and W_o are the battery power consumed by the network interface when a node sends, receives, or overhears a packet; M_r, M_s, and M_o are the amount of three types of packets respectively. Therefore, BPC_i is obtained by averaging the total amount of the power consumed for every T seconds sampling intervals.

Then the energy consumption rate at a node, N_i, is calculated in the following formula:

$$R_i = \frac{RBP_i}{BPC_i} \qquad (13)$$

where RBP_i denotes the remaining battery power at a node, N_i.

Because the maximum lifetime of a given path, L_p, is determined by the minimum value of R_i over the path, where we denote the set of all available paths as P, that is:

$$L_p = \min_{\forall n_i \in P} R_i \qquad (14)$$

The congestion status of MANET is also imperative for selecting a reliable routing path. GFMRP takes the buffer occupancy rate at a node as a parameter for selecting routes that are not congested. The congestion status of the network is measured as the work load at each node's interface, i.e., the number of the packets buffered at the interface. The ratio of $\frac{q_i}{b_i}$ denotes the buffer occupancy rate at a node, where q_i is the most recent packet queue length at a node, N_i, and b_i is the buffer capacity at that node. The destination node estimates the congestion status of a specific routing path using the formula below:

$$congestion = 1 - \frac{\sum q_i}{\sum b_i} \qquad (15)$$

Link stability parameter helps to select the routes, which are comparatively more stable and long-lived, in order to ensure lower packet loss rate, fewer route failures and less frequent route discovery. Link stability can be measured using the signal strength. The signal strength has a relationship with the receiver's antenna gain, and is inversely squared proportional to the square of the distance *d*. As *d* increases, the degree of signal strength becomes weak.

The number of intermediate hops is one of the most popular selection parameters for finding a routing path. It allows the routing protocols to find the routes having the shortest distance. To some degree, the shortest distance in network means the least end-to-end delay between the source and the destination. And if the data packets flow through the smallest number of intermediate hops, they will have fewer chances for path failures, which results in higher reliability and longer life span of a given path.

GFMRP adopts these selection parameters (L_p, congestion, signal strength and the number of intermediate hopes) in its fuzzy inference system as input values in order to choose the most optimal and reliable set of the multiple routing paths.

8.4.2 Design of Fuzzy Inference System

As we know, the human brain interprets imprecise and incomplete sensory information provided by perceptive organs. Fuzzy set theory represents and numerically manipulates such linguistic information in a natural way via membership functions and fuzzy rules. It can be a general methodology to incorporate knowledge, heuristics or theory into controllers and decision makers. Therefore, fuzzy set theory has been applied in a control decision system either to improve the performance or to handle the problems that conventional control theory cannot approach successfully because the latter relies on a valid and accurate model which does not always exist. Several special issues on computational intelligence in telecommunication networks have been published by the *IEEE Journal on selected Area in communications* [31], which shows researchers have recently taken intelligent techniques into telecommunication networks. We here apply the fuzzy inference system (FLS) to the multi-path routing problems in MANET.

The inputs into our FLS are: 1) packet buffer occupancy rate, 2) energy consumption rate, 3) signal strength, and 4) the number of intermediate hops. These four selection parameters, as we analyzed before, reflect the network status and the nodes ability to reliably deliver network packets. These four parameters are updated in RREQ packets as the route discovery progresses. Finally, when the RREQ packet arrives at the destination node, it inputs the four parameters of the route to the FLS, and produces an output value which represents the rank of the route. According to the ranks of all possible route paths, the destination node makes a decision to select a reliable multi-path set, and returns this result through the RREPs. The steps involved in the calculation of rank value of a route are elaborated as following:

Step1. Fuzzification of Inputs and Outputs

The four input variables to be fuzzified are the energy consumption rate, the packet buffer occupancy rate, the signal strength, and the number of intermediate hops. On the basis of existing knowledge of MANET, the terms “Empty” and “Full” are used to describe the energy consumption rate and the packet buffer occupancy rate. “Strong” and “Weak” are terms for representing the signal strength. The terms used to describe the number of the intermediate hops are “Small” and “Large”. One the other hand, in order to indicate whether a certain route path is reliable, we divide the rank value into 16 levels. T(F_m) = { F_1, through F_{16}} denotes the linguistic values of output, the rank value of a path. Even though the choice and specification of the membership functions are widely subjective, there are several principles for membership function selection that can produce good adequate results. The trapezoidal functions are chosen as the membership function since they have been extensively used in real-time applications due to their simple formulas and computational efficiency. We show these membership functions in Figure 8.1. We normalize the linguistic values of inputs and outputs in the range from 0 to 1.

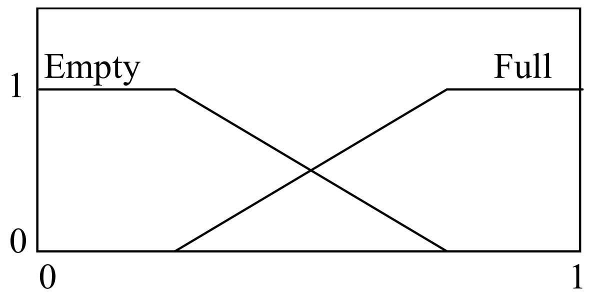


Figure 8.1(a). Membership function for energy consumption rate and packet buffer occupancy rate

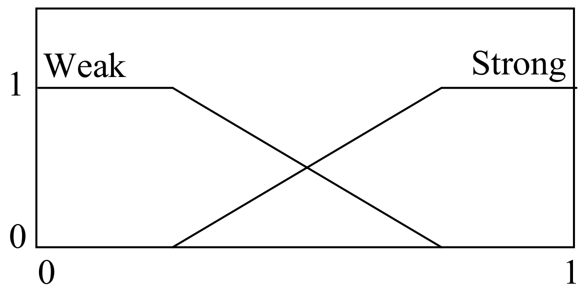


Figure 8.1(b). Membership function for signal strength

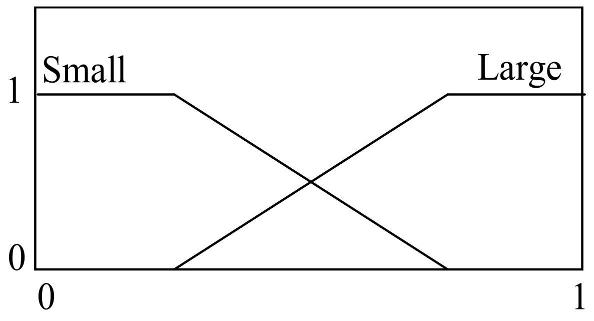


Figure 8.1(c). Membership function for the number of the intermediate hops

Step2. Knowledge Base Rule Structure

The knowledge base is a set of rules developed using expert knowledge. We design the knowledge based rules connecting the inputs and the output based on a thorough understanding of the system. The parameters and rules of our FLS are initially set, induced from many analytical results of MANET routing, and then further calibrated through simulations using an evolutionary optimization technique. The fuzzy rules have

IF-THEN structure. The inputs are then combined using the AND operator. The following is an example of rules which describes the input output mapping.

Ri: if battery power is Empty(A_i), buffer occupancy is Full(B_j), signal strength is Weak(C_k) and the number of hops is Large(D_l), then the path is F_m.

where A_i, B_j, C_k, and D_l are fuzzy sets defined in the corresponding input spaces, while F_m is the fuzzy set defined in the output space, rank value. Table 8.1 shows the initial fuzzy rules for the FLS. The three input variables have 16 combinations.

TABLE 8.1 Fuzzy rule base

No	Power Con.	Buffer Occ.	Signal Strength	Hop Num.	Rank
1	E	E	W	L	F1
2	E	E	W	S	F2
3	E	E	S	L	F3
4	E	E	S	S	F4
5	E	F	W	L	F5
6	E	F	W	S	F6
7	E	F	S	L	F7
8	E	F	S	S	F8
9	F	E	W	L	F9
10	F	E	W	S	F10
11	F	E	S	L	F11
12	F	E	S	S	F12
13	F	F	W	L	F13
14	F	F	W	S	F14
15	F	F	S	L	F15
16	F	F	S	S	F16

Step3. Parameter Optimization of the Fuzzy System

The parameters and rules of FLS are initially set, and then further calibrated to do a more efficient job. Because ad hoc wireless network traffic has self-similarity using variance-time-plotting, a common statistical method which has been widely used to verify self-similarity of time-series, the characteristics of ad hoc wireless network can be captured [59]. We obtain a training data set from *ns2*, and then use offline training to finalize the optimal value of parameters and fuzzy rules in FLS. We will illustrate the detailed procedure of optimization of membership functions in Chapter 9.

A traditional back-propagation algorithm is usually adopted for optimization, but it is extremely time consuming. The hybrid learning method that combines the steepest descent and least squares estimator is the fastest learning algorithm [44]. However, the method needs to fix the values of the premise parameters as the prerequisite. GFMRP uses the Genetic Algorithm (GA) that is one of the most popular evolutionary algorithms that model biological processes to optimize highly complex cost functions. A genetic algorithm allows a population including many individuals to evolve under specified selection rules to a state that maximizes the “fitness”. GA has a lot of advantages. Here we define the learning objective as minimizing the error function:

$$E = \sum_{i \in trainingdataset} \frac{1}{2} E_i^2 \tag{16}$$

$$E_i = | rank_i - realRank_i | \tag{17}$$

When building the training data set, we obtain the “realRank” for every routing path in the training data set. For each routing path, we send 1000 data packets, and then take the fraction of packet delivery success as the “realRank”, which is a value from 0 to 1.

Step4. Defuzzification

Defuzzification refers to the way a crisp value is extracted from a fuzzy set as a representation value. There are many kinds of defuzzifiers. Here we take the centroid of area strategy for defuzzification.

$$z_{coz=} \frac{\int_z \mu_A(z) z dz}{\int \mu_A(z) dz} \qquad (18)$$

where $\mu_A(z)$ is the aggregated output of the Membership Function (MF). This is the most widely adopted defuzzification strategy, which is reminiscent of the calculation of the expected value of probability distributions.

8.4.3 Description of Routing Protocol

GFMRP consists of a route discovery, a route reply, and a route maintenance phase. Every node in MANET acts as both a terminal and a router. Each node can become a destination for data traffic, thus, FLS described in Section 4 is embedded in every mobile node. Such FLS in a given destination node produces the crisp output rank values to indicate the fitness of all possible routing paths between the source node requiring route discovery and the destination node. According to the rank values, the destination node can make a decision for selecting an optimal reliable set of multiple paths as the candidates for delivering data traffic.

When a source needs to start a communication with a node, first, it checks its own routing cache. If it cannot find any available route path entries, it sends a route discovery (RREQ) packet that collects the information of power consumption rate, buffer occupancy rate, and signal strength for all the individual nodes along the path. In the RREQ packets, five items are included (Figure 8.2)

min_energy_rate	sum_queue_len	sum_buffer_len	min_sig	HopNum

Figure 8.2. RREQ packet format

- min_energy_rate: the minimum of energy consumption rates of the mobile nodes in a route path;
- sum_queue_len: the sum of the most recent packet queue lengths of the mobile nodes in a route path;
- sum_buffer_len: the sum of the buffer capacities of the mobile nodes in a route path.
- min_sig: the minimum of signal strengths of the mobile nodes in a route path;
- HopNum: the number of intermediate hops in a route path

When an intermediate node receives a RREQ packet, it piggybacks this RREQ packet, calculates its energy consumption rate, buffer occupancy rate and signal strength using the formulas represented in Section 3, and then updates the information in RREQs according to the simple algorithm shown in Figure 8.3. Thus, RREQ packets accumulate the effects of every intermediate node on network state and congestion status.

```
UpdateRREQ(node Ni){
   check the RREQ packet;
   if ($R_i$ < min_energy_rate)
        min_energy_rate = $R_i$ ;
   sum_queue_len = sum_queue_len + $q_i$;
   sum_buffer_len = sum_buffer_len + $b_i$;
   if (signal_strength < min_sig)
       min_sig=signal_strength;
     HopNum=HopNum+1;
}
```

Figure 8.3. Pseudo code for updating RREQ packet algorithm

While the destination node receives the first RREQ packet, it starts a timer for the route selection time window. During this period different RREQ packets reach the destination. After the timer window expires, the destination selects the multi-paths which have the highest rank values. Route reply (RREP) packets containing the multi-paths information are sent from the destination back to the source along the opposite path of the selected route with the highest rank. In order that the destination node can obtain the information of all possible routes, when getting the RREQ packet, the intermediate nodes are not allowed to send RREPs back to the source even when they have route information for the destination in their own caches. The destination node will get the multi-path information by the RREQ packets, and sends them to the source node by RREP. The overlapped route problem is avoided with a different packet forwarding approach. Instead of dropping all the duplicate RREQs, intermediate nodes forward the duplicate packets that traversed through a different incoming link than the link from which the first RREQ is received, and whose hop count is not larger than that of the first received RREQ.

If the source node has found multi-path routing information in its cache, it forwards data packets to the least used reliable route path. Thus, GFMRP distributes the traffic load evenly in a simple round-robin fashion, and the data packets are spread over all the available paths in a reliable multiple path set.

8.5 Simulation

We conducted experiments to evaluate and compare the performance of the following protocols: GFMRP, SMR, SBMR and DSR. In theses experiments, we used the discrete time network simulator, *ns2*, which offers high fidelity in wireless ad hoc network simulation by including an accurate implementation of data link and physical layers. Fifty mobile nodes were moved according to the random waypoint mobility model within a 1500 m * 300 m area. Each node had a radio propagation range of 250m and channel capacity was 2Mb/s. All simulations were run for 600 seconds of simulated time.

We did our experiments with movement patterns for 7 difference pause times: 0, 100, 200, 300, 400, 500 and 600 seconds. Thirty mobile nodes acted as traffic sources generating 4 packets/second each, and data traffic was generated using constant bit rate (CBR) UDP traffic sources. The medium access control protocol was the IEEE 802.11 DCF. The size of data packet was 512 bytes. The minimum and the maximum speeds were set constant to zero and 20m/s respectively.

Packet delivery ratio is important as it describes the loss rate that will be seen by the transport protocols, which in turn affects the maximum throughput that the network can support. Figure 8.4 presents that packet delivery ratio is the highest for GFMRP, which is due to its ability to select a set of stable and least congested routes thus having the lowest amount of congestion loss and very few route failures.

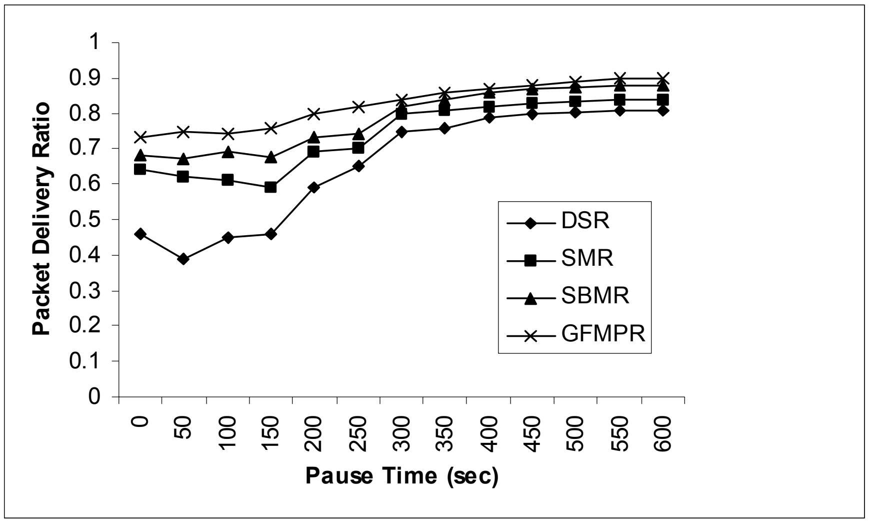


Figure 8.4. Fraction of successfully delivered data packets as a function of mobility

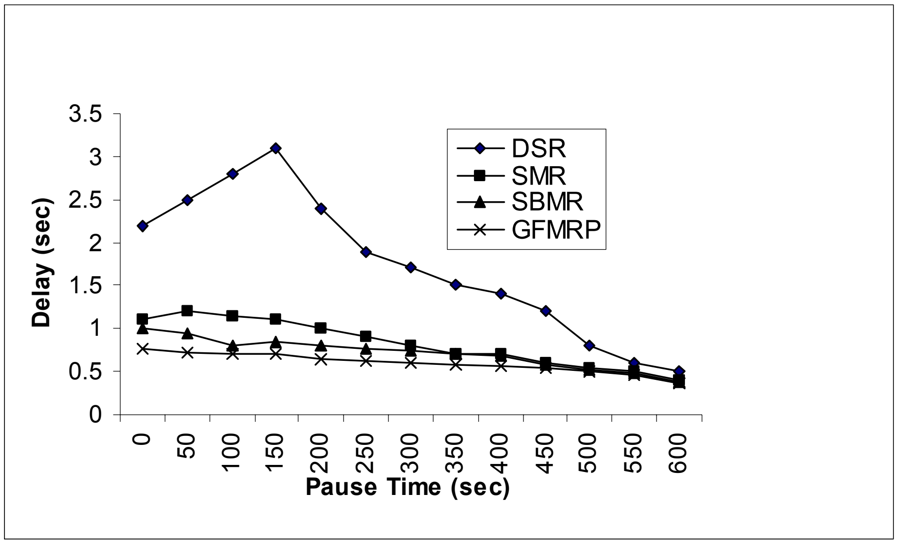


Figure 8.5. Average end-to-end delay as a function of mobility

The average end-to-end delay is the average elapsed time to deliver a packet from the source node to the destination node, and it includes all possible delays before data packets arrive at their destinations. The average end-to-end delay for GFMRP is the lowest compared with DSR, SMR and SBMR (Figure 8.5). It is obvious that DSR has the highest delay as DSR does route rediscovery frequently. The delay of SMR and SBMR are higher than GFMRP because of the higher route failures, overhead and congestions.

Route rediscovery is needed to locate an alternate route for the given destination. It is an expensive task. So the less frequency of routing discovery process means the less route discovery latency and lower routing overhead. Figure 8.6 shows that the frequency of route rediscovery of GFMRP is the lowest among those of the compared routing protocols. Because GFMRP deals with the uncertainty of MANET and considers the effects of different correlated parameters on network performance, GFMRP decreases the route failures significantly.

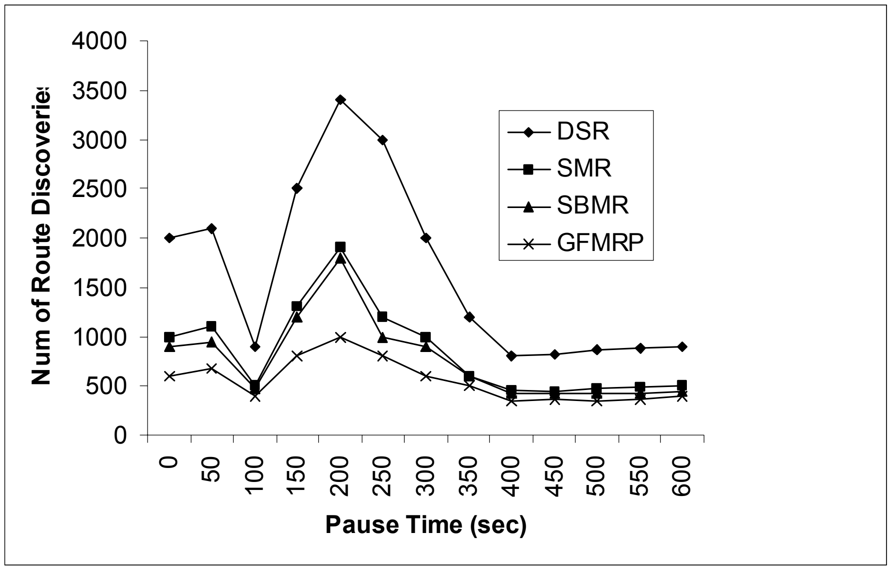


Figure 8.6. Number of route discoveries as a function of mobility

CHAPTER 9

GENETIC OPTIMIZATION OF FUZZY MEMBERSHIP FUNCTIONS

In Chapter 8, we propose a genetic fuzzy multipath routing protocol for MANET. In our protocol, each mobile host embeds a Fuzzy Logic System (FLS), which is responsible for evaluating the performance of all the potential routing paths between a source and a destination node, and then choosing the most optimal paths as the candidates for multiplath routing. FLS has shown good performance on many existing control and design systems. However, FLS is an ill-defined function for analyzing, and it is difficult to design a good performing FLS. Generally, the design of FLSs involves determination of the number of fuzzy rules, the structure of the rules, and membership function parameters. Fortunately, Genetic Algorithm (GA) is a good optimizer for FLSs, and it has attracted many researchers [49, 91]. We illustrate here how to optimize the membership functions in GFMRP using a GA in this chapter.

This chapter is organized as follows. In the next subsection we provide basic knowledge about genetic algorithms. Section 9.2 introduces operators of genetic algorithms. The process of the membership functions optimization for GFMRP is illustrated in Section 9.3.

9.1. Introduction of Genetic Algorithms

The genetic algorithm is a subset of evolutionary algorithms that model biological processes to optimized highly complex cost functions. I. Rechenberg introduced the idea of evolutionary computing in 1960s in his work “Evolution strategies” [35, 55]. Many researchers developed his idea. Genetic algorithms were invented by John Holland over the course of the 1960s and 1970s and finally popularized by his students and colleagues [55]. As simulating the survival of the fittest among individuals over consecutive generation for solving a problem, each generation of the genetic algorithm consists of a population of character strings that are analogous to the chromosome that we see in our DNA. Each individual represents a point in a search space and a possible solution. The individuals in the population are then made to go through a process of evolution. This is motivated by a hope, that the new population will be better than the old one during the evolution.

The basic genetic algorithm includes the following steps as shown in Figure 9.1:

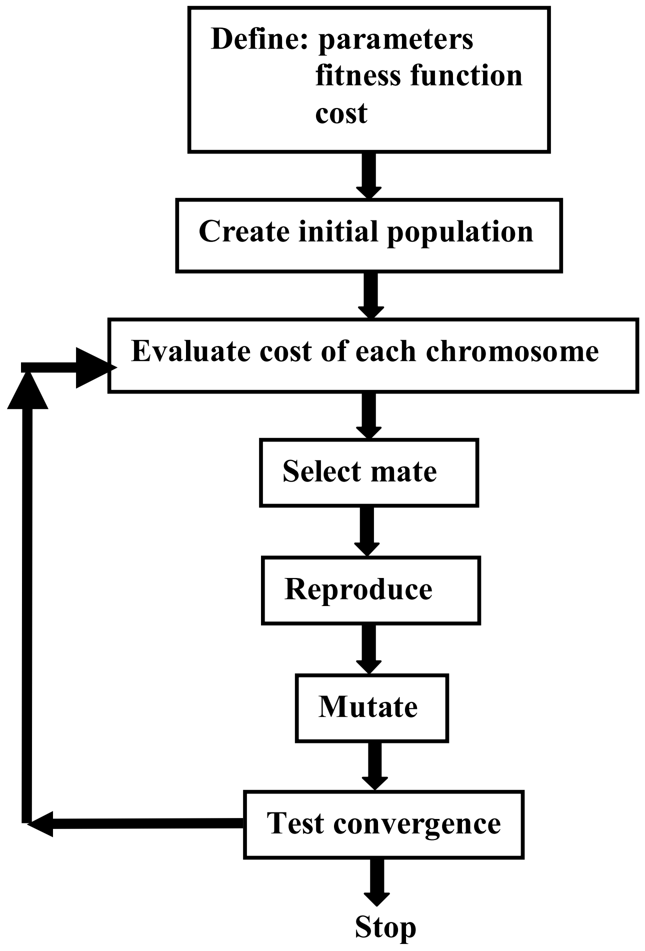


Figure 9.1 Flow chart of a genetic algorithm

1. Start: Define the optimization parameters, the cost function, and the cost.
2. Generate random initial population of *n* chromosomes or produce suitable solutions for the problem.
3. Fitness: Evaluate the fitness *f(x)* of each chromosome *x* in the population
4. New population: Create a new population by repeating the steps from 3 – 6 until the new population is complete.

(1) Selection: Select two parent chromosomes from a population according to their fitness. The better the fitness the higher the chance it gets to be selected.

(2) Crossover: With a crossover probability cross over the parents to from a new offspring. If no crossover is performed, then the offspring is an exact copy of the parents.

(3) Mutation: With a mutation probability mutate the new offspring at each locus, which is each position in the chromosome.

(4) Accepting: Place the new offspring generated in the new population.

5. Replace: Use new generated population for a further run of the algorithm
6. Test for convergence: If the end condition is satisfied, stop the execution and return the best solution in the current population.
7. Loop: Go to step 2 again.

In general, a genetic algorithm has two popular implementations. One represents parameters as an encoded binary string and works with the binary strings to minimize the cost, while the other works with the continuous parameters themselves to minimize the cost. In this research, we will optimize the membership functions of our fuzzy logic system used in multipath routing. Recalling in Chapter 8, the values of fuzzy inputs are normalized to a range from 0 to 1. Thus, each locus of the chromosome represents the value of each parameter in membership functions, which is a floating point number ranging from 0 to 1. Since we take trapezoidal functions as membership functions for input and output parameters, there are total 76 parameters optimized in membership functions (16 comes from four input parameters, and 60 comes from fuzzy sets of output parameters). So, we are interested in permutation encoding in which the index represents the parameter in a membership function and the floating point number represents the value of its corresponding parameter as shown below. The chromosome has 76 parameters (an 76-dimensional optimization problem) given by $p_1, p_2, \ldots, p_{76}$. For example, below are two chromosomes.

Chromosome 1: 0.121 0.142 0.205 0.227 … 0.871 0.912 0.965

Chromosome 2: 0.161 0.182 0.216 0.264 … 0.844 0.965 0.988

9.2. Operators of Genetic Algorithm

There are three important operators in genetic algorithm: crossover, mutation and selection.

9.2.1. Crossover

Crossover selects genes from parent chromosomes and creates a new offspring. Many different approaches have been tried for crossing over in continuous parameter genetic algorithms. The simplest methods choose one or more points in the chromosome to mark as the crossover points. Then the parameters between these points are merely swapped between the two parents. For example purpose, consider the two parents to be

$$\text{Parent 1: } [p_{m1}, p_{m2} \underset{\uparrow}{,} p_{m3}, p_{m4} \underset{\uparrow}{,} p_{m5}, p_{m6}, \ldots, p_{m76}]$$

$$\text{Parent 2: } [p_{d1}, p_{d2} \underset{\uparrow}{,} p_{d3}, p_{d4} \underset{\uparrow}{,} p_{d5}, p_{d6}, \ldots, p_{d76}]$$

Crossover points are randomly selected shown by the arrows, then the parameters in between are exchanged. The new offsprings are shown below:

Parent 1: $[p_{m1}, p_{m2}, p_{d3}, p_{d4}, p_{m5}, p_{m6}, \dots, p_{m76}]$

Parent 2: $[p_{d1}, p_{d2}, p_{m3}, p_{m4}, p_{d5}, p_{d6}, \dots, p_{d76}]$

Different crossover methods can be used on different problems. The type of crossover made for a specific problem may improve the performance of the genetic algorithm.

9.2.2. Mutation

Mutation takes place after the crossover is performed. The reason to use mutation is to prevent the solutions of the population from falling into a local optimum of solved problem [35, 55]. To avoid this problem of overly fast convergence, we force the routine to explore other areas of the cost surface by randomly introducing changes, or mutations, in some parameters. In binary encoding, we can randomly switch a few bits from 1 to 0 or vice versa. The basic method of mutation is not much more complicated for the continuous parameter genetic algorithm. In this research, we randomly delete the parameter in a chromosome and replace that with a new uniform random number between 0 and 1. The following parameter is mutated: p_1 of $chromosome_{10}$.

Chromosome 10: [0.121 0.142 0.205 0.227 … 0.871 0.912 0.965]

Chromosome 10: [0.136 0.142 0.205 0.227 … 0.871 0.912 0.965]

9.2.3. Selection

When we have many solutions, we need to select some of them to produce the offspring in pairs. The selection is used to try the parents on the fitness function to get the better fitting parents. The selected parents are used as parents in the next generation.

9.3. Parameters of Genetic Algorithm

There are four basic parameters in genetic algorithm, which are crossover probability and mutation probability, the size of population and the number of generations.

Crossover probability is the parameter that determines how often the crossover will be performed. If the crossover probability is 0%, it means no crossover is performed, and the offspring is an exact copy of the parent. If the crossover probability is 100%, then the offspring is completely made by crossover.

Mutation probability is the parameter to control how often the parts of the chromosome will be mutated. If no mutation is performed, offspring is taken after crossover without any change. If mutation is performed, part of the chromosome is changed. If mutation probability is 100%, the whole chromosome is changed. The reason to use mutation is to prevent the genetic algorithm from falling into the local extreme, but it should not occur very often, because it will change the genetic algorithm to a random search.

Population size is the parameter to determine how many chromosomes are in the population for one generation. The population size cannot be too small or too large. A large population provides the genetic algorithm with a nice sampling of the search space. However, if the population size is very large, it will slow down the execution of the whole process. With larger population, even if we get better results, but if it is too time consuming, it is not suitable. Researches show that after some limit (which depends mainly on the encoding and the problem), there is no use even if we increase the population size, because it makes solving the problem much slower [35, 55].

Generation is an important parameter in genetic algorithm. The more generations a test case has the better chance that the genetic algorithm will find an optimal solution. In each generation, we use the fitness function to select the better solutions to be the members of the next generation. Then, this new generation will produce its next possible better offspring. With more generations a genetic algorithm can produce a better solution. However, as the generation size gets bigger and bigger, the test cases take more and more time to execute. The performance is related to the number of generations.

9.4. The Genetic Algorithm Used in This Research

We describe the key points about the procedure of the genetic algorithm used in our research work in this section.

9.4.1. Parameters and Fitness Function

An important point in the implementation of a GA is representation of the problem domain. In fuzzy logic system of GFMRP, we adopt trapezoidal membership functions in both input and output universes of discourse as shown in Figure 9.2. Our FLS consists of 4 inputs variables $x = (x_1, x_2, x_3, x_4)^T$, one output variable $y(x)$ and 16 knowledge rules. Trapezoidal membership functions are described by.

$$f_i^l(x_i) = \left\{ \begin{array}{c} 0, x \le a. \\ \dfrac{x-a}{b-a}, a \le x \le b. \\ \dfrac{d-x}{d-c}, c \le x \le d \\ 0, d \le x. \end{array} \right\}$$

The parameters $\{a, b, c, d\}$ (with $a < b \le c < d$) determine the x coordinates of the four corners of the underlying trapezoidal Membership Function (MF).

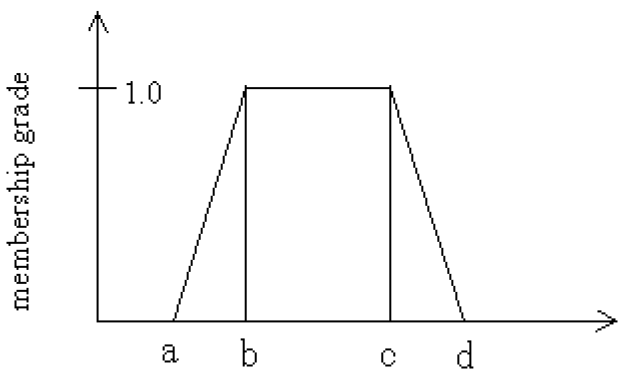


Figure 9.2. Trapezoid membership function

A typical rule in this FLS has the form.

R^l: if battery power (x_1) is A_1^l, buffer occupancy (x_2) is A_2^l, signal strength (x_3) is A_3^l and the number of hops (x_4) is A_4^l, then the path is F_m.

where A_i^l is labels of fuzzy sets characterized by the fuzzy membership functions defined as above.

The goal is to construct the accurate high-dimensional fuzzy logic system via optimizing the parameters of membership function. Therefore, we begin the process of fitting this optimization problem to a genetic algorithm by defining a chromosome as an array of real numbers, which represents the parameters of membership functions to be optimized. Each chromosome of the population should have enough "genetic" information in its strings to represent four parameters {*a*, *b*, *c*, *d*} for each membership function. Say we have two membership functions for four of the fuzzy inputs, and sixteen membership functions for the fuzzy output. That would make a total of 76 parameters which define the membership functions, since we can set zero to *a* and *b* of the first membership functions of fuzzy values, and set zero to *c* and *d* of the last membership functions for fuzzy values as shown in Figure 9.3. This can reduce the number of parameters which need to be optimized in membership functions. One hundredth is the accuracy of real numbers encoding the parameters of all membership functions. For example purpose (Figure 9.4.), below shows MF parameters encoding substring for all system variables. Note, there exists some constraints among each pair of {*a*, *b*, *c*, *d*} with $a < b \leq c < d$.

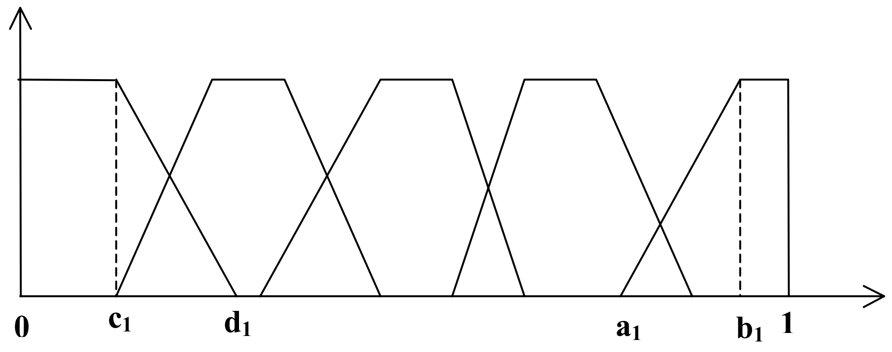


Figure 9.3. Example of parameters for fuzzy inputs and outputs

c_1	d_1	a_2	b_2		A_{23}	B_{23}
0.11	0.23	0.16	0.24	…	0.95	0.98

Figure 9.4. MF parameters substring for all system variables

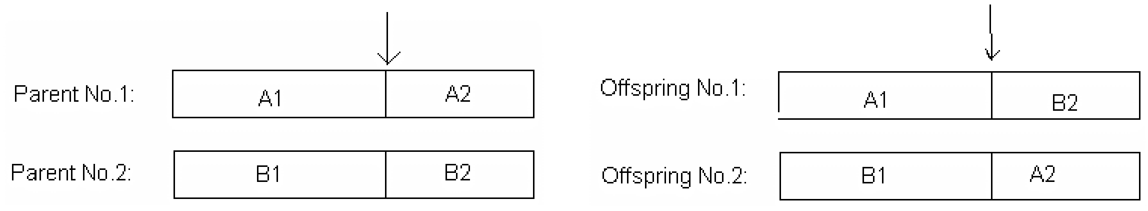
Here, we define the fitness function as the error function, so our objective is to minimize the error function. We select three paths with different reliability from *ns2*. We denote the three paths as P_1, P_2, and P_3 with the decreasing order of reliability. For each routing path, we send 1000 data packets, and then take the fraction of packet delivery success as the "realRank", which is a value from 0 to 1. Then, the fitness function is calculated as below.

$$E = \sum_{i \in \{P_1, P_2, P_3\}} \frac{1}{2} E_i^2$$

$$E_i = | rank_i - realRank_i |$$

9.4.2. Crossover

Crossover is the most important operator in the genetic algorithm. Two types of crossover operations are used in this research. One is called single point crossover; the other is double point crossover. The single point crossover is to randomly generate one point, then operate the crossover. The parents look like the following: The parent no.1 is *A1+A2*, the parent no.2 is *B1+B2*, and the point that the arrow refers is the randomly generated single point as shown in Figure 9.5(a). After the single point crossover, the offspring no.1 will be *A1+B2*; offspring no.2 will be *B1+A2* as shown in Figure 9.5(b).



(a) Parents **(b) offspring after cross over**

Figure 9.5. Example of single point crossover

Double point crossover is to generate two points randomly. The parents are divided into three parts with those two points as bounds. Suppose the parent no.1 is: *A1+A2+A3*, and the parent no.2 is *B1+B2+B3*, then the offspring no.1 will be *A1+B2+A3*; the offspring no.2 will be *B1+A2+B3*. The procedure is shown in Figure 9.6.

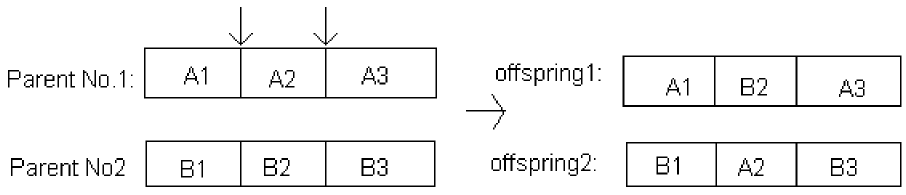


Figure 9.6. Example of double point crossover

The population size for the GA described above is fixed at 40 members. Once the fitness of each member of the population is evaluated, the weakest members are killed. The fittest members then reproduce according to crossover and cloning. In cloning, the population member is reproduced identically, that means bring an exact copy of few parent chromosome into the next generation. In this research, the fittest 10 percent of the population was cloned at the end of each generation. The least fit 50 percent of the population was discarded. Then the fittest 50 percent of the population (including the clones) mated to restore the population size to 40.

9.4.3. Mutation

Mutation is another important operator in genetic algorithm. We applied a mutation at a rate of 4%. Since there are 40 chromosomes in one generation, this amounts to a total of two mutations. The mutation rate represents how often each gene (parameter in MFs) in one chromosome gets changed to another real number. Normally the mutation rate should be small because if it is too big, it will change the original population too much and will lead a random search. We delete the real number, which needs to mutate, and replace it with a new uniform random number between 0 and 1.

9.4.4. Selection

The goal of using genetic algorithm in this research is to optimize the fuzzy logic system. We actually want to minimize the estimate error function. The first step is to evaluate the estimate error for each chromosome for each generation. The smaller the estimate error of a member, the more chance it survives. Some of the members which have larger estimate errors are discarded.

9.4.5. Convergence

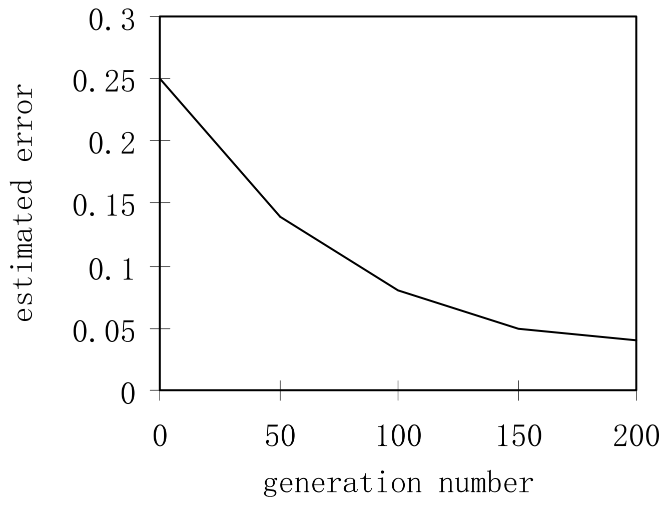


Figure 9.7. The improvement in fitness values as the population evolved

Generation is an important parameter in genetic algorithm. The more generations a test case has the better chance that the genetic algorithm will find a better solution. In each generation, we use fitness function to pick up the better solutions to be the members of the next generation. Then, this new generation will produce its next possible better offspring. With more generations the genetic algorithm can produce a better solution. However, as the generation size gets bigger and bigger, the test case takes more and more time to execute. The performance is related to the number of the generations. Therefore, there should be a limit on the generation size. Figure 9.7 shows the improvement in fitness as the population evolves.

9.5. Summary

In order to determine the accurate FLS in our multipath routing, we constrain the membership functions to a shape of trapezoids then each membership function can be parameterized by a small number of variables and the membership optimization problem can be reduced to a parameter optimization problem. We take the genetic algorithm to solve the optimization problem. The values of parameters are initially set through the expert knowledge, and then tuned by GA. From Figure 9.7, we can see the GA has reduced the estimate error from 0.26 to about 0.05. Therefore, the resulted FLS is accurate and very suitable to model the real environment.

CHAPTER 10

SUMMARY AND FUTURE WORK

This final chapter concludes the dissertation and identifies some future research directions.

This research develops a novel and effective method for using and delivering location information in MANETs. Our proposal location management service has two main advantages. Firstly, it maintains the property: long distance queries are proportionally penalized. Secondly, it maps a node's unique ID to the central coordination of a specific minimum region via a hash function. Its packet forwarding strategy allows that all packets destined for an arbitrary location (possibly unoccupied by a node) to be forwarded consistently to the same node in the neighborhood of that location. Then the node receiving the location maintenance packet acts as one of the location servers. In the process of location server's selection and update, it only requires one occurrence of control packet forwarding and hash function computation. Thus, this location service is very efficient and scalable. Currently, most prior work focused on designing different MANET routing protocols and providing simulation-based performance support. 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. We develop a mathematical model to compare the performances between our new location service and the classical location service, GLS, based on a quantitative study of the control overhead of both location services. The analytical results show that the control overhead of GLS grows as $O(vN^{\frac{3}{2}})$ while that of EnSLS grows as $O(N^{\frac{3}{2}} + vN \log N)$. Therefore, EnSLS has better scalability than that of GLS theoretically.

It is a popular topic to approximate virtual backbone in MANETs via connected dominating sets. However, finding a minimum connected dominating set in a graph is an NP-complete problem. In this research, we proposed the following iterative localized algorithm for connected dominating sets, improving the concept by Wu and Li [106]. In our new algorithm, each node sends five messages to all its neighbors, and the process is synchronized. Each node that decides not to be in the connected dominating set (CDS) becomes passive; otherwise it is active and reevaluates the decision in the next round. First, each node verifies whether or not it has two unconnected neighbors, and informs neighbors about it. Next, each active node (that remains in CDS) after the first step will check whether it is covered by one neighbor with higher ID, and all nodes then send message informing about their CDS status. In the next step, each active node checks whether or not it is covered by one of remaining active neighbors with the lower ID. All nodes again send the message by the algorithm (although nodes, once declared as passive, do not need to send further messages). In the next iteration, active nodes covered by two active neighbors with higher IDs decide to become passive, and informs neighbors by a message. In the next iteration, active nodes that have two active neighbors that together cover it, one with higher ID and one with lower ID, decide to withdraw from CDS and become passive, informing neighbors about the status change. In the final iteration, all active nodes that are covered jointly by two active neighbors with lower IDs also decide to become passive and withdraw from CDS. They may send sixth message if each node wants to know which of its neighbors remains in CDS. The experimental data show that the size of CDS has been reduced dramatically with respect to the concept originally proposed by Wu and Li [106].

The iterative methods reduce the size of CDS through avoiding illegal simultaneous removal of dominating nodes by control messages and synchronization. We also present a new marking process, which prefer to selecting “optimal” nodes into the CDS by classification of neighboring nodes. The correctness and efficiency are proved in the dissertation. The experimental results support that the new marking process reduces the size of initial CDS largely.

One the other hand, this dissertation addresses the correlation of multiple routing objectives. It shows that multiple routing objectives can be meet altogether if we consider correlated different route selection metrics at the same time. We present a simple and effective protocol called Genetic Fuzzy Multi-path Routing Protocol (GFMRP), which considers the multiple correlated selection parameters, based on fuzzy set theory and evolutionary computing. GFMRP employs a simple load-balancing method to distribute the traffic load, by spreading data packets over all the paths of the reliable multiple paths set in a round-robin fashion. The performance of GFMRP is evaluated in terms of packet delivery ratio, average end-to-end delay, and the frequency of route rediscovery in *ns2* context. Simulation results demonstrate that GFMRP can be more adaptive to the ad hoc environment and outperforms some famous existed multiple path routing protocols. In the design of fuzzy logical system, the parameters and knowledge based rules are initially set via analytical models of MANETs, and are calibrated through genetic algorithms.

In order to determine the accurate FLS in our multipath routing, we constrain the membership functions to a shape of trapezoids then each membership function can be parameterized by a small number of variables and the membership optimization problem can be reduced to a parameter optimization problem. We take the genetic algorithm to solve the optimization problem. The values of parameters are initially set through the expert knowledge, and then tuned by GA. Therefore, the resulted FLS is accurate and very suitable to model the reality environment.

Currently, we use offline training to tune the parameters of FLS in GFMRP that cannot reflect the real time conditions of MANETs. In the future research, I would like to explore the area of self-tuning multipath routing protocol. For example, I plan to add the online training to the proposed multipath routing protocol just through the information collected by RREQ and RREP packets, and make the protocol more intelligent and efficient.

I still believe that biologically inspired techniques have promising potential applications in MANETs since they are very suitable to build distributed, robust, adaptive, scalable, and environmentally aware systems. Biological inspired techniques include artificial neural networks, evolutionary computation, swarm intelligence, etc. A variety of research problems in MANETs can be addressed in a unified way as complex distributed adaptive systems and solved via biologically inspired techniques. In the future, I would like to propose some biologically inspired routing protocols in MANETs or sensor networks, aiming at improving intelligent cooperation among mobile nodes.

In the future, I also plan to apply the inter-disciplinary approaches (game theory, swarm intelligence, time-series forecasting and etc) into the open issues in wireless ad hoc and sensor networks. For example, Mobile hosts can control connections and disconnections by the willingness to collaborate during the formation of short-lived networks. This often leads to inefficient, unstable routes. Cooperative game concepts can be applied to develop protocols that would encourage intermediate nodes to collaboratively select routes beneficial to all mobile hosts involved. I believe there are many similar topics that could lead to new discoveries when studied from other inter-disciplinary perspectives, i.e. game theory, swarm intelligence and time series forecasting.

BIBLIOGRAPHY

[1] *ATM forum, ATM user-network interface specification, version 3.0.*

[2] *USCG Navigation Center GPS page*. http://www.nis-mirror.com/systems/gps/

[3] G. Aggelou and R. Tafazolli, *RDMAR: a bandwidth-efficient routing protocol for mobile ad hoc networks*, *Proceedings of the ACM International Workshop on Wireless Mobile Multimedia (WoWMoM)*, Seattle, WA, 1999, pp. 26-33.

[4] G. Alandjani and E. E. Johnson, *Fuzzy routing in ad hoc networks*, *Proceedings of the 2003 IEEE International Performance Computing and Communications Conference*, 2003, pp. 525 - 530.

[5] J. Alber, M. R. Fellows and R. Niedermeier, *Polynomial-time data reduction for dominating set*, Journal of the ACM (JACM), 51 (2004), pp. 363-384.

[6] G. Altun, *Security in wireless ad hoc network routing protocols*, *Georgia State University*, 2003.

[7] K. M. Alzoubi, P.-J. Wan and O. Frieder, *Distributed heuristics for connected dominating sets in wireless ad hoc networks*, J. Comm. and Networks, 4 (2002), pp. 22-29.

[8] K. M. Alzoubi, P. J. Wan and O. Frieder, *New distributed algorithm for connected dominating set in wireless ad hoc networks*, *Proc. 35th Hawaii Int. Conf. On System Sciences*, 2002, pp. 1-7.

[9] A. D. Amis, R. Prakash, T. H. P. Vuong and D. T. Huynh, *Max-min D-cluster formation in wireless ad hoc networks*, *Proc. IEEE INFOCOM*, 2000, pp. 32-41.

[10] G. H. Ash, A. H. Kafker and K. R. Krishnan, *Servicing and real-time control of networks with dynamic routing*, Bell System Technical Journal, 60 (1981).

[11] T. Back, *Evolutionary algorithms in theory and practice*, Oxford University Press, New York, 1996.

[12] B. S. Baker, *Approximation algorithms for NP-complete problems on planar graphs*, Journal of the ACM, 41 (1994), pp. 153-180.

[13] S. Basagni, I. Chlamtac, V. R. Syrotiuk and B. A. Woodward, *A distance routing effect algorithm for mobility (DREAM)*, *In Proc. ACM/IEEE MobiCom*, 1998, pp. 76-84.

[14] R. E. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.

[15] V. Bharghavan and B. Das, *Routing in ad hoc networks using minimum connected domination sets*, *in Proc. Int. Conf. Commun*, Montreal, Canada, 1997.

[16] P. Bose, P. Morin, I. Stojmenovic and J. Urrutia, *Routing with guaranteed delivery in ad hoc wireless networks*, *In Proc. Of 3rd ACM Intl. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications DIAL M99*, 1999, pp. 48-55.

[17] B. Chen, K. Jamieson, H. Balakrishnan and R. Morris, *Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks*, ACM Wireless Networks J. , 8 (2002), pp. 481-494.

[18] T.-W. Chen and M. Gerla, *Global state routing: a new routing scheme for ad-hoc wireless networks*, *Proceedings of the IEEE International Conference on Communications (ICC)*, Atlanta, GA, 1998, pp. 171-175.

[19] M. S. Corson and A. Ephremides, *A distributed routing algorithm for mobile wireless networks*, ACM/Baltzer Wireless Networks, 1 (1995), pp. 61-81.

[20] F. Dai and J. Wu, *Distributed dominant pruning in ad hoc wireless networks*, In Proc. Of IEEE International Conference on Communications (ICC), 2003.

[21] F. Dai and J. Wu, *An extended localized algorithm for connected dominating set formation in ad hoc wireless networks*, IEEE Trans. Parallel and Distributed Systems, 15 (2004), pp. 908-920.

[22] B. Das, R. Sivakumar and V. Bharghavan, *Routing in ad hoc networks using a spine*, *in Proc. Int. Conf. Comput. and Commun. Networks*, Las Vegas, NV., 1997.

[23] S. M. Das, H. Pucha and Y. C. Hu, *Performance comparison of scalable location services for geographic ad hoc routing*, *Proc. IEEE INFOCOM*, 2005.

[24] S. Datta, I. Stojmenovic and J. Wu, *Internal nodes and shortcut based routing with guaranteed delivery in wireless networks*, Cluster Computing, 5 (2002), pp. 169-178.

[25] R. Dube, C. D. Rais, K.-Y. Wang and S. K. Tripathi, *Signal Stability based Adaptive routing (SSA) for ad hoc mobile networks*, IEEE Personal Communications Magazine . 4 (1997), pp. 36-45.

[26] A. Ephremides, J. Wieselthier and D. Backer, *A design concept to reliable mobile radio networks with frequency hopping signaling*, *Proceedings of the IEEE 75(1)*, 1987, pp. 56-73.

[27] L. R. Ford and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.

[28] K. Gabriel and R. Sokal, *A new statistical approach to geographic variation analysis*, Systematic Zoology, 18 (1969), pp. 259-278.

[29] J. J. Garcia-Luna-Aceves and M. Spohn, *Scalable link-state internet routing*, *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, Austin, TX, 1998, pp. 52-61.

[30] J. J. Garcia-Luna-Aceves and M. Spohn, *Source-tree routing in wireless networks*, *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, Toronto, Canada, 1999, pp. 273-282.

[31] E. Gelenbe, I. W. Habib, S. PaLazzo and S. Douligeris, *Special Issue on Intelligent Techniques in High Speed Networks*, IEEE J. Selected Areas in Communications, 18 (2000), Feb.

[32] M. Gerla, X. Hong, L. Ma and G. Pei, *Landmark routing protocol (LANMAR) for large scale ad hoc networks*, *Internet Draft: draft-ietf-manet-lanmar-04.txt*, 2002.

[33] M. Gerla and J.-C. Tsai, *Multicluster, mobile, multimedia radio network*, ACM J. Wireless Networks, 1 (1995), pp. 255-265.

[34] R. J. Gibbons, F. P. Kelley and P. B. Key, *Dynamic alternative routing - modeling and behavior*, *Proceedings of the 12th International Teletraffic Conference*, 1988.

[35] D. E. Goldberg, *Genetic algorithms in search, Optimization and Machine Learning*, Addison Wesley, 1989.

[36] S. Guha and S. Khuller, *Approximation algorithms for connected dominating sets*, Algorithmica, 20 (1998), pp. 374-387.

[37] Z. Haas, M. Pearlman and P. Samar, *The interzone routing protocol (IERP) for ad hoc networks*, *Internet Draft: draft-ietf-manet-zone-ierp-01.txt*, 2001.

[38] Z. Haas, M. Pearlman and P. Samar, *The intrazone routing protocol (IARP) for ad hoc networks*, *Internet Draft: draft-ietf-manet-zone-iarp-01.txt*, 2001.

[39] Z. J. Haas and B. Liang, *Ad hoc mobility management with uniform quorum systems*, IEEE/ACM Transaction on Networking, 7 (1999), pp. 228-240.

[40] T.-C. Hou and V. O. K. Li, *Transmission range control in multihop packet radio networks*, IEEE Trans. On Communications, 34 (1986), pp. 38-44.

[41] H.-Y. Hsieh and R. Sivakumar, *On using the ad-hoc network model in cellular packet data networks*, *Proc. ACM MobiHoc*, Lausanne, SZ, 2002.

[42] P. Jacquet, P. Muhlethaler and A. Qayyum, *Optimized link state routing protocol*, *Internet-Draft, draft-ietf-manet-olsr-00.txt*, 1998.

[43] R. Jain, A. Puri and R. Sengupta, *Geographical routing using partial information for wireless ad hoc networks*, IEEE Personal Communications (2001), pp. 48-57.

[44] J. S. R. Jang, C.-T. Sun and E. Mizutani, *Neuro-fuzzy and soft computing, a computational approach to learning and machine intelligence*, Prentice Hall, Upper Saddle Rive, NJ, 1996.

[45] D. B. Johnson and D. A. Maltz, *Dynamic source routing in ad hoc wireless networks*, in T. Imielinski and H. Korth, eds., *Mobile computing*, Kluwer Publishing Company, 1996, pp. 153-181.

[46] R. Kahn, *Advanced in packet radio technology*, *Proceedings of the IEEE 66*, 1978, pp. 1468-1496.

[47] B. Karp and H. T. Kung, *GPSR: greedy perimeter stateless routing for wireless networks*, *Proc. ACM/IEEE MobiCom*, 2000.

[48] D. Kim, J. J. Garcia-Luna-Aceves, K. Obraczka, J. C. Cano and P. Manzoni, *Routing mechanisms for mobile ad hoc networks based on the energy drain rate*, IEEE Transactions on Mobile Computing, 2 (2003), pp. 161 – 173.

[49] J. Kim, Y. Moon and B. P. Zeigler, *Designing fuzzy net controllers using genetic algorithms*, IEEE Control Systems Magazine, 15 (1995), pp. 66-72.

[50] L. Kleinrock and J. Silvester, *Spatial reuse in multihop packet radio networks*, *Proc. IEEE*, 1987, pp. 156-167.

[51] Y.-B. Ko and N. H. Vaidya, *Location-aided routing (LAR) in mobile ad hoc networks*, *Proceedings of ACM/IEEE MOBICOM*, Dallas, TX, 1998, pp. 66-75.

[52] U. C. Kozat, G. Kondylis, B. Ryu and M. K. Marina, *Virtual Dynamic Backbone for Mobile Ad Hoc Networks*, *in IEEE International Conference on Communications (ICC)*, Helsinki, Finland, 2001.

[53] E. Kranakis, H. Singh and J. urrutia, *Compass routing on geometric networks*, *In Proc. Of 11th Canadian Conf. on Computational*, Geometry, Vancouver, 1999.

[54] T. J. Kwon and M. Gerla, *Efficient flooding with passive clustering (PC) in ad hoc networks*, ACM SIGCOMM Computer Comm. Rev., 32 (2002), pp. 44-56.

[55] D. Lawrence, *Handbook of genetic algorithms*, Van Nostrand Reinhold, New York, 1991.

[56] S. J. Lee and M. Gerla, *Split multipath routing with maximally disjoint path in ad hoc networks*, *Proceedings of IEEE ICC*, Helsinki, Finland, 2001, pp. 3201-3205.

[57] B. Leiner, R. Ruth and A. R. Sasty, *Goals and challenges of the DARPA GloMo program*, IEEE Personal Communications (1996), pp. 34-43.

[58] J. Li, J. Jannotti, D. Couto, D. R. Karger and R. Morris, *A Scalable location service for geographic ad hoc routing*, *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MOBICOM 00)*, 2000, pp. 120-130.

[59] Q. Liang, *Ad hoc wireless network traffic self-similarity and forecasting*, IEEE Communications Letters, 6 (2002), pp. 297-299.

[60] W.-H. Liao, Y.-C. Tseng and J.-P. Sheu., *Grid: a fully location-aware routing protocol for mobile ad hoc networks*, Telecommunication Systems, 18 (2001), pp. 37-60.

[61] C. R. Lin and M. Gerla, *Adaptive clustering for mobile wireless networks*, IEEE J. Selected Areas in Comm., 15 (1996), pp. 1265-1275.

[62] C. R. Lin and J.-S. Liu, *QoS routing in ad hoc wireless networks*, IEEE Journal on Selected Areas in Communications special issue on Wireless Ad Hoc Networks, 17 (1999), pp. 1426-1438.

[63] H. Liu, J. Li, Y. Pan and Y. Zhang, *An adaptive genetic fuzzy multi-path routing protocol for wireless ad-hoc networks*, *the 1st ACIS International Workshop on Self-Assembling Wireless Networks (SAWN 2005)*, Maryland, U.S.A, 2005.

[64] H. Liu and Y. Pan, *A scalable location management scheme for position-based routing in mobile ad hoc networks*, in Y. Xiao, J. Li and Y. Pan, eds., *Security and Routing in Wireless Networks*, Nova Science Publishers, 2005.

[65] H. Liu, Y. Pan and J. Cao, *An improved distributed algorithm for connected dominating sets in wireless ad hoc networks*, *Proc. Int. Symp. on Parallel and Distributed Processing and Applications (ISPA) Lecture Notes in Computer Science* 2004, pp. 340-351.

[66] J. Macker and M. S. Corson, *Mobile ad hoc network and the IETF*, ACM Mobile Computing and Communication Review, 2 (1998), pp. 9-12.

[67] G. Malkin, *RIP Version 2 - carrying additional information*, *Internet Draft, draft-ietf- ripv2-protocol-v2-05.txt*, 1998.

[68] M. V. Marathe, H. Breu, H. B. H. III, S. S. Ravi and D. J. Rosenkrantz, *Simple Heuristics for Unit Disk Graphs*, Networks, 2 (1995), pp. 59-68.

[69] S. Marwaha, D. Srinivasan, K. T. Chen and A. Vasilakos, *Eutionary fuzzy multi-objective routing for wireless mobile ad hoc networks*, Congress on Eutionary Computation 2004, 2 (2004), pp. 1964 - 1971.

[70] M. Mauve, J. Widmer and H. Hartenstein, *A survey on position-based routing in mobile ad-hoc networks*, IEEE Network (2001).

[71] J. M. McQuillan, I. Richer and E. C. Rosen, *The new routing algorithm for the ARPANET*, IEEE Transactions on Communications, 28 (1980), pp. 711-719.

[72] J. Moy, *OSPD Version 2. Internet Request For Comments 1247*, (1991).

[73] S. Mueller, R. P. Tsang and D. Ghosal, *Multipath routing in mobile ad hoc networks: issues and challenges*, *Performance tools and applications to networked systems, LNCS*, springer-verlag, 2004.

[74] S. Murthy and J. J. Garcia-Luna-Aceves, *An efficient routing protocol for wireless networks*, ACM/Baltzer Mobile Networks and Applications special issue on Routing in Mobile Communications Networks, 1 (1996), pp. 183-197.

[75] A. Nasipuri and S. R. Das, *On-demand multipath routing for mobile ad hoc networks*, *Proceedings of the IEEE International Conference on Computer Communications and Networks (ICCCN)*, Boston, MA, 1999, pp. 64-70.

[76] C. A. S. Oliveira and P. M. Pardalos, *Ad hoc networks: optimization problems and solution methods*, in D.-Z. Du, M. Cheng and Y. Li, eds., *Combinatorial Optimization in Communication Networks*, Kluwer, Dordrecht, 2006, pp. 147-169.

[77] V. Park and S. Corson, *Temporally-ordered routing algorithm (TORA) version 1 functional specification*, *Internet Draft: draft-ietf-manet-tora-spec-04.txt*, 2001.

[78] M. R. Pearlman and Z. J. Haas, *Determining the optimal configuration for the zone routing protocol*, IEEE Journal on Selected Areas in Communications special issue on Wireless Ad Hoc Networks, 17 (1999), pp. 1395-1414.

[79] G. Pei, M. Gerla and T.-W. Chen, *Fisheye state routing: a routing scheme for ad hoc wireless networks*, *Proceedings of the IEEE International Conference on Communications (ICC)*, New Orleans, LA, 2000, pp. 70-74.

[80] C. E. Perkins and P. Bhagwat, *Highly dynamic destination-sequenced distance Vector routing (DSDV) for mobile computers*, *Proceedings of the ACM SIGCOMM Conference on Communications Architectures Protocols and Applications*, London, UK, 1994, pp. 234-244.

[81] C. E. Perkins and E. M. Royer, *Ad-hoc on demand distance vector routing*, *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, New Orleans, LA, 1999, pp. 90-100.

[82] M. B. Pursley and H. B. Russell, *Routing in frequency-hop packet radio networks with partial-band jamming*, IEEE Transactions on Communications, 41 (1993), pp. 1117-1124.

[83] A. Qayyum, L. Viennot and A. Laouiti, *Multipoint relaying for flooding broadcast message in mobile wireless networks*, *Proc. 35th Ann. Hawaii Intl Conf. System Sciences (HICSS)* 2002, pp. 298.

[84] J. Raju and J. J. Garcia-Luna-Aceves, *A new approach to on-demand loop-free multipath routing*, *Proceedings of the IEEE International Conference on Computer Communications and Networks (ICCCN)*, Boston, MA, 1999, pp. 522-527.

[85] S. Ratnasamy, B. Karp, L. Yin and F. Yu, *GHT: a geographic hash table for data-centric storage*, *In WSNA*, 2002.

[86] T. J. Ross, *Fuzzy logic with engineering applications*, McGraw Hill, 1995.

[87] E. M. Royer and C. K. Toh, *A review of current routing protocols for ad hoc mobile wireless networks*, IEEE Personal Communications, 6 (1999), pp. 46-55.

[88] R. Ruth, *Global mobile information systems program overview*, 1998.

[89] J. Shaikh, J. Solano, I. Stojmenovi and J. Wu, *New metrics for dominating set based energy efficient activity scheduling in ad hoc networks*, *Proc. WLN Workshop at IEEE Conf. on Local Computer Networks*, Bonn, Germany, 2003, pp. 20-24.

[90] J. Shi, Z. Ling, S. Dong and Z. Jie, *A stability-based multipath routing algorithm for ad hoc networks*, *14th IEEE Proceedings on Personal Indoor and Mobile Radio Communications*, 2003, pp. 516 - 520.

[91] D. Simon and H. El-Sherief, *Fuzzy logic for digital phase-locked loop filter design*, IEEE Transactions on Fuzzy Systems, 3 (1995), pp. 211 – 218.

[92] R. Sivakumar, B. Das and V. Bharghavan, *An improved spine-based infrastructure for routing in ad hoc networks*, *in Proc. IEEE Symp. Comput. And Commun.*, Athens, Greece, 1998.

[93] I. Stojmenovic, *Data gathering and activity scheduling in ad hoc and sensor networks*, *Proc. International Workshop on Theoretical Aspects of Wireless Ad Hoc, Sensor, and Peer-to-Peer Networks*, Chicago, Illinois, 2004.

[94] I. Stojmenovic, *Home region based location updates and destination search schemes in ad hoc wireless networks*, SITE, University of Ottawa, 1999.

[95] I. Stojmenovic and X. Lin., *Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks*, IEEE Transactions on Parallel and Distributed Systems, 12 (2001), pp. 1023-1032.

[96] I. Stojmenovic, M. Seddigh and J. Zunic, *Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks*, *Proc. IEEE Hawaii Int. Conf. On System Sciences*, 2001.

[97] H. Takagi and L. Kleinrock, *Optimal transmission ranges for randomly distributed packet radio terminals*, IEEE Trans. On Communications, 32 (1984), pp. 246-257.

[98] A. S. Tanenbaum, *Computer Networks*, Prentice Hall, Upper Saddle River, NJ, 1996.

[99] C.-K. Toh, *Associativity-based routing for ad-hoc mobile networks*, Wireless Personal Communications Journal, special issue on Mobile Networking and Computing Systems, Kluwer Academic Publishers, 4 (1997), pp. 103-139.

[100] G. Toussaint, *The relative neighborhood graph of a finite planar set*, Pattern Recognition, 12 (1980), pp. 261-268.

[101] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen and J.-P. Sheu, *The broadcast storm problem in a mobile ad hoc network*, Wireless Networks, 8 (2002), pp. 153-167.

[102] L. Wang, L. Zhang, Y. Shu and M. Dong, *Multipath source routing in wireless ad hoc networks*, *2000 Canadian Conference on Electrical and Computer Engineering*, 2000, pp. 479-483.

[103] S. C. M. Woo and S. Singh, *Scalable routing protocol for ad hoc networks*, Wireless Networks, 7 (2001), pp. 513-529.

[104] J. Wu, F. Dai and S. Yang, *Iterative local solutions for connected dominating sets in MANETs*, *submitted to ICDS*, 2005.

[105] J. Wu and H. Li, *A dominating-set-based routing scheme in ad hoc wireless networks*, Telecomm. System, special issue on wireless networks, 18 (2001), pp. 13-36.

[106] J. Wu and H. Li, *On calculating connected dominating sets for efficient routing in ad hoc wireless networks*, *Proc. ACM DIALM*, 1999, pp. 7-14.

[107] J. Wu, B. Wu and I. Stojmenovi, *Power-aware broadcasting and activity scheduling in ad hoc wireless networks using connected dominating sets*, Wireless Communications and Mobile Computing, 4 (2003), pp. 425-438.

[108] Y. Xue, B. Li and K. Nahrstedt, *A scalable location management scheme in mobile ad-hoc networks*, *Proc. IEEE conference on local computer networks (LCN)*, 2001.

[109] L. Zhou and Z. Haas, *Securing ad hoc networks*, IEEE Network Magazine, 13 (1999), pp. 24-30.

[110] H. Zou, *An algorithm for detecting termination of distributed computation in arbitrary network topologies within linear time*, *International Symposium on Parallel Architectures Algorithms and Networks (ISPAN '96)*, 1996, pp. 168-172.