Multipoint Relay and Connected Dominating Set Based Broadcast Algorithms for Wireless Ad Hoc Networks

Ou Liang

Department of Electrical & Computer Systems Engineering
Monash University
Melbourne, Australia

Thesis Submitted for
the Degree of Master of Engineering Science

May 2007

Abstract

Recent advances in wireless communication technology and microelectro-mechanical systems have greatly expanded the potential applications of wireless ad hoc networks. This allows the formation of temporary networks without any centralized administration or support from base stations. Individual nodes dynamically discover connections, and thus being capable of relaying packets in order to deliver data across the network. Due to their infrastructure-less and self-organizing properties, wireless ad hoc networks are especially suitable for applications when there is no fixed communication infrastructure available, such as in times of emergency, tactical military operations on battlefields or temporary networks for disaster recovery.

Substantial research has been conducted on wireless ad hoc networks focusing on a wide range of areas. Among them, developing efficient data dissemination algorithms is considered to be an important research focus for two reasons. First, the broadcast method is widely used in a large number of routing protocols. Applications developed to distribute various kinds of messages such as route query, control messages and alarm signals in these networks. Second, since the devices in these networks usually have a limited power supply, simply adopting the broadcast methods used in wired networks may lead to a high number of unnecessary transmissions, thus resulting in energy wastage and congestion in wireless ad hoc networks. Therefore, improving the efficiency of broadcast protocols has a very significant impact on the stability of these networks. Even though many algorithms have been proposed to achieve greater efficiency in wireless ad hoc networks, most do not perform well in the aspects of operational simplicity, minimization of redundant transmissions, and scalability in dense networks.

Within the scope of this research project, the existing broadcast algorithms for

wireless ad hoc networks were evaluated first. Operational details of the algorithms, their weaknesses and possible improvements were also critically reviewed. Three new efficient broadcast algorithms called Gateway Multipoint Relay (GMPR), Enhanced Gateway Multipoint Relay (EGMPR) and Low-Cost Flooding (LCF) were then proposed to achieve better performance in the aspects outlined in the previous paragraph. They were developed based on the Multipoint Relay (MPR) and connected dominating set (CDS) methods. These algorithms were validated and their performance bounds were determined through theoretical analyses and a series of proofs. To further demonstrate the efficiency of the proposed algorithms, simulation based experiments were also conducted to compare them with a number of leading algorithms. The results show that the proposed broadcast algorithms are more efficient in minimizing redundant transmissions, and they perform satisfactorily as the network densities are scaled up. In particular, the LCF algorithm performs best among all and is suitable for operating in dense networks since its computational and communication complexities increase linearly as network density increases, and its signaling message size and approximation ratio are bounded by constants.

Acknowledgements

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

First I would like to thank my supervisors Dr. Ahmet Şekercioğlu and Dr. Nallasamy Mani for their excellent guidance. Especially Dr. Ahmet, he taught me how to do the research and gave me so many valuable pieces of advice on the research topic. He was always patient and helpful whenever his assistance is needed, and he has contributed much of his valuable time to help me to improve my technical writing skills. Therefore, I do want to show my deep appreciation to him.

I am also grateful to my colleagues and friends in the department for their companionship, help and support. Especially Johnny Lai for his help on OMNeT++ and LaTeX 2_ε, Steve Woon on physical training, and Dr. Himal Suraweera for his suggestions on my research.

Finally, I would like to show my great appreciation to my parents for their continuous support during these years. Also, my wife, Viona Xin, for her persistent encouragement and delicious meals. Without them, this work would not have been possible.

The majority of the simulations in this research are run on the computational facilities provided by the Victorian Partnership for Advanced Computing (VPAC).

Declaration

I declare that, to the best of my knowledge, the research described herein is original except where the work of others is indicated and acknowledged, and that the thesis has not, in whole or in part, been submitted for any other degree at this or any other university.

Ou Liang
Melbourne
May 2007

Contents

1 Introduction **1**

1.1 An Overview of Wireless Ad Hoc Networks 1

1.2 Broadcast Algorithms in MANETs 5

1.2.1 Blind Flooding Problem 6

1.2.2 Categories of Existing Approaches 7

1.3 Aims and Contribution of This Research 8

1.4 Outline of the Thesis 9

2 A Survey of Efficient Broadcast Algorithms for MANETs **10**

2.1 Introduction 10

2.2 Overview of the Multipoint Relay Method 11

2.3 Connected Dominating Set 14

2.4 Costs of Broadcast Algorithms 16

2.5 MPR Based Algorithms 19

2.5.1 Pure MPR Algorithms 19

2.5.2 QoS based MPR Algorithms 27

2.5.3 Summary of the MPR Based Algorithms 31

2.6 CDS Based Algorithms 36

2.6.1 Direct-CDS Based Algorithms 37

2.6.2 Cluster-CDS Based Algorithms 45

2.6.3 Summary of CDS Based Algorithms 52
2.7 Summary 56

3 Proposed Efficient Broadcast Algorithms 58
3.1 Introduction 58
3.2 Gateway MPR Algorithm 58
3.2.1 Generating Gateways in Networks 61
3.2.2 Generating a CDS 62
3.2.3 Self-Pruning Procedure 63
3.3 Enhanced Gateway MPR Algorithm 66
3.3.1 Drawbacks of GMPR Algorithm 67
3.3.2 Extended Hello Message Format 68
3.4 Low-Cost Flooding Algorithm 70
3.4.1 Hello Message Formats of LCF Algorithm 72
3.4.2 Connecting Two-Hop Away Dominators 74
3.4.3 Connecting Three-Hop Away Dominators 75
3.5 Summary 79

4 Theoretical Analysis of Proposed Algorithms 81
4.1 Introduction 81
4.2 Validation of Proposed Algorithms 82
4.2.1 Verification of GMPR Algorithm 82
4.2.2 Verification of EGMPR Algorithm 86
4.2.3 Verification of LCF Algorithm 86
4.3 Performance Evaluation of Proposed Algorithms 88
4.3.1 Performance of GMPR Algorithm 88
4.3.2 Performance of EGMPR Algorithm 90
4.3.3 Performance of LCF Algorithm 90

4.4 Summary 96

5 Simulation Studies of Proposed Algorithms 97

5.1 Introduction 97

5.2 Simulation Framework 98

5.3 Simulation Scenario 101

5.4 Analysis of the Results 104

5.4.1 Comparison of the Number of Forwarding Nodes 104

5.4.2 Comparison of the Number of Signaling Messages 110

5.4.3 Comparison of the Average Signaling Message Size 113

5.5 Summary 116

6 Conclusions and Recommendations for Future Work 117

6.1 Conclusions 117

6.2 Recommendations for Future Work 121

A Architecture of Mobility Framework Model 123

A.1 Overview of Mobility Framework Model 123

A.1.1 Mobile Host Model 124

A.1.2 Mobility Architecture and Dynamic Connection Management 126

B Configuration of Simulation 129

B.1 OMNeT++ Configuration File 129

B.2 C++ Program to Calculate Positions of Hosts 133

C Publications 137

List of Figures

1.1 A possible wireless ad hoc network. 2

1.2 Basic flooding problem. 6

2.1 An MPR flooding example. 12

2.2 A possible CDS in a given graph. 15

2.3 An example of the in-degree MPR algorithm. 21

2.4 An example of the minimum overlapping MPR algorithm. 23

2.5 An example of the UMPR algorithm. 26

2.6 QoS problem in the original MPR heuristic. 28

2.7 An example of the worst scenario in the original MPR heuristic. . . 35

2.8 Two drawbacks of the MPR-CDS algorithms. 39

2.9 Extended Rule 1 in the DEMPR algorithm. 41

2.10 A pair of MPRs in the EEMPR algorithm. 42

2.11 Examples of redundant connectors generated in the GCDS algorithm. 48

3.1 The state transition diagram of a node in the GMPR algorithm. . . . 60

3.2 The hello message format of the GMPR algorithm. 61

3.3 An example of constructing a CDS by using the GMPR algorithm. . 64

3.4 Three example networks to show the drawbacks of the GMPR algorithm. 67

3.5 hello message formats of the LCF algorithm (fixed length source and destination address, and sequence number fields are not shown). 72

3.6 An example of utilization of two types of connectors in the LCF algorithm (the letters next to the nodes represent node IDs). 74

3.7 An example of connecting three-hop dominators. Arrows represent hello messages and their sending directions. 80

4.1 Illustration of the validation of IS. 82

4.2 Illustration of the proof of MIS. 83

4.3 Illustration of the proof of Lemma 4.3. 85

4.4 Illustration of the proof of Lemma 4.4. 87

5.1 The protocol stack used in Mobility Framework. 99

5.2 An illustration of the simulation network area used in this research. . 102

5.3 A comparison of the number of forwarding nodes generated by different algorithms for a given 100-node network topology. 105

5.4 Average number of forwarding nodes generated by the algorithms for the transmission ranges of 15m. 107

5.5 Average number of forwarding nodes generated by the algorithms for the transmission ranges of 25m. 108

5.6 Average number of forwarding nodes generated by the algorithms for the transmission ranges of 50m. 108

5.7 Average number of signaling messages generated by each node during the CDS construction process for the transmission ranges of 15m. 110

5.8 Average number of signaling messages generated by each node during the CDS construction process for the transmission ranges of 25m. 111

5.9 Average number of signaling messages generated by each node during the CDS construction process for the transmission ranges of 50m. 111

A.1 An example simulation network setup with 10 hosts by using Mobility Framework model. 124

A.2 Structure of a mobile host model in the MF. 125
A.3 The structure of the nic module. 126
A.4 Mobility architecture. 127

List of Tables

2.1 Classification of MPR based algorithms 19

2.2 Cost comparison of the MPR based algorithms. 32

2.3 Classification of CDS based algorithms 36

2.4 The Cost comparison of the CDS based algorithms. 53

4.1 Cost comparison of proposed algorithms. 95

5.1 Average message size (in units of node IDs) for $R = 15m$. 113

5.2 Average message size (in units of node IDs) for $R = 25m$. 114

5.3 Average message size (in units of node IDs) for $R = 50m$. 115

Listings

B.1 The *omnetpp.ini* file used in this study. 129

B.2 The C++ program to generate a connected random topology. 133

List of Symbols

n	The total number of nodes in a network
V	The set that contains all nodes of a network
E	The set that contains all edges of a network
$\deg(u)$	The node degree of a node u
Δ	The maximum node degree in a network
Φ	The maximum number of two-hop neighbors of a node in a given network
$N_1(u)$	The one-hop neighbor set of a node u
$N_2(u)$	The two-hop neighbor set of a node u
$\mathrm{MPR(u)}$	The MPR set of a node u
M	The maximum size of an MPR set generated in the network
P_u	The preference of a node u
$D_k(u)$	The set of dominators that are k hops away from node a u
$A(u)$	The set of active connectors of a node u
$I(u)$	The set of isolated dominators of a node u
$P(u)$	The set of passive connectors of a node u
$S(u)$	The set of special dominators of a node u

List of Acronyms

AODV . . . . . On-Demand Distance Vector Routing Protocol

CDS . . . . . Connected Dominating Set

CI . . . . . Confidence Interval

DEMPR . . . . . Degree-Based Multipoint Relay Algorithm

DoD . . . . . Department of Defends

DS . . . . . Dominating Set

DSR . . . . . Dynamic Source Routing Protocol

ECDS . . . . . Efficient Connected Dominating Set Algorithm

EEMPR . . . . . Extended Enhanced Multipoint Relay Algorithm

EGMPR . . . . . Enhanced Gateway Multipoint Relay Algorithm

ETSI . . . . . European Telecommunication Standards Institute

GCDS . . . . . Geometric Connected Dominating Set Algorithm

GLOMO . . . . . Global Mobile Information Systems

GMPR . . . . . Gateway Multipoint Relay Algorithm

GPS . . . . . Global Positioning System

HIPERLAN . . . High Performance Radio Local Area Network

ID-MPR . . . . . In-Degree Multipoint Relay Algorithm

IEEE . . . . . Institute of Electrical and Electronic Engineers

IETF . . . . . Internet Engineering Task Force

IS . . . . . Independent Set

LAN . . . . . Local Area Network

LCF . . . . . Low-Cost Flooding Algorithm

MAC . . . . . Medium Access Control

MADN . . . . . Metropolitan Ad Hoc Network

MANET . . . . . Wireless/Mobile Ad Hoc Network

MCDS . . . . . Minimum Connected Dominating Set

MF . . . . . Mobility Framework

MIS . . . . . Maximum Independent Set

MO-MPR . . . . . Minimum Overlapping Multipoint Relay Algorithm

MOCDS . . . . . Message Optimal Connected Dominating Set Algorithm

MPR . . . . . Multipoint Relay Algorithm

MPR-CDS . . . . . Multipoint Relay Based Connected Dominating Set Algorithm

MPR-HR . . . . . MPR-Based Hybrid Routing Protocol

MPRDV . . . . . Multipoint Relay Distance Vector protocol

OLSR . . . . . Optimized Link State Routing Protocol

OMNET++ . . . . Objective Modular Network Testbed in C++

P-MPR . . . . . Prioritized Multipoint Relay Algorithm

PRNET . . . . . Packet Radio Network

QMPR . . . . . QoS-Based Multipoint Relay Algorithm

QOS . . . . . Quality of Service

RFC . . . . . Request for Comments

RP-MPR . . . . . Random Prioritized Multipoint Relay Algorithm

SP . . . . . Self-Pruning Procedure

SURAN . . . . . Survivable Adaptive Radio Network

TCP . . . . . Transmission Control Protocol

TTL . . . . . Time To Live

UBF . . . . . Utility-Based Flooding Algorithm

UMPR . . . . . Utility-Based Multipoint Relay Algorithm

VANET . . . . . vehicular Ad Hoc Network

WSN . . . . . Wireless Sensor Network

Chapter 1

Introduction

1.1 An Overview of Wireless Ad Hoc Networks

The rapid development of wireless communications, micro-electronics and mobile computing devices has dramatically affected our information society. Cell phones, handheld computers and other portable digital devices are becoming smaller, cheaper and more powerful in terms of data processing and communication capabilities. In addition, advances in wireless communication technologies have also boosted the growth of applications and network services for these mobile devices thus enabling the emergence of wireless ad hoc networks. This chapter gives a brief overview of the development of these networks, and highlights some important research issues.

Wireless ad hoc networks, also called mobile ad hoc networks (MANETs) [1, 2], are collections of autonomous mobile nodes or terminals that communicate with each other by forming a multi-hop wireless radio network. Each node in a MANET can act as both a host or a router to receive and forward packets, and it can randomly move around, leave the network or switch off. Moreover, new nodes may

Figure 1.1: A possible wireless ad hoc network.

join the network unexpectedly. These characteristics make a MANET an unstable network where links between nodes frequently break. In such dynamic topologies, each node is responsible for establishing connections with neighbor nodes and for relaying packets on behalf of other nodes, and therefore, temporary networks can be set up with no or limited infrastructure support. Due to these properties, MANETs have great application potential in various scenarios, such as battle field communication, emergency services, disaster recovery, personal entertainment and mobile conferencing [3, 4]. Thorough surveys of MANETs can be found in [3, 5]. A possible example of a MANET is shown in Fig. 1.1. As can been seen in the figure, handheld devices connect to each other via wireless links thus creating a multi-hop network. Messages generated by the cellphone on the left of the figure can be relayed by other devices and reach any destination. Furthermore, a wireless router also performs as a gateway to connect the network to the Internet. Such a wireless network can construct itself automatically, and ubiquitously support users in accessing information or communicating with others.

Despite its popularity, the concept of an ad hoc network is not new. Significant research has been ongoing in this area for nearly three decades. The history of ad hoc networks can be traced back to the Packet Radio Network (PRNET) in

1972, and the Survivable Adaptive Radio Network (SURAN) in the early 1980s [6]. Both programs were for military use and they aimed to provide packet switching networking to the battlefield where fixed infrastructures can be easily targeted. Enlightened by the fast growth of the Internet and notebook computers, the idea of the commercial use of ad hoc networks was firstly put forward in the early 1990s. The Global Mobile Information Systems (GloMo) program [7] initiated by the U.S. Department of Defense (DoD) in 1994 is a particular example, whose goal was to provide office-environment Ethernet-type multimedia connectivity anytime, anywhere, in handheld devices. However, up to that time, tactical networks were the only real (non-prototypical) applications in the ad hoc paradigm. The interest in commercial usage of ad hoc networks was largely increased in the mid to late 90s due to the introduction of some low-cost wireless technologies such as Bluetooth[1] and IEEE 802.11[2]. Moreover, the establishment of the Internet Engineering Task Force (IETF) MANET Working Group, which sought to standardize routing protocols for ad hoc networks, also spurred the interest of developing ad hoc network applications outside the military field. Recently, some commercially oriented applications such as Mesh Network [8], Vehicular Ad Hoc Network (VANet) [9] and Metropolitan Ad Hoc Network (MADN) [10] have been proposed, which show possible solutions that can turn a MANET into a commodity.

In the MANET domain, many research areas have potential study value and thus attract much attention [3, 11]. Currently, the popular research issues can be categorized in the following groups:

- **Routing:** Routing functionality is an essential part of the ad hoc domain. Since topologies change frequently in MANETs, efficient routing protocols

[1] Bluetooth official info website http://www.bluetooth.com/bluetooth
[2] IEEE 802.11 working group: http://www.ieee802.org/11

are required to adapt to unstable networks and guarantee the packet delivery.

- **Multicasting / Broadcasting:** Multicast and broadcast have been widely used by many applications and routing protocols to distribute messages to a group of nodes or all nodes in the network. Efficient multicast and broadcast protocols are important for achieving energy efficiency in MANETs.

- **Location Service:** Applications that use Global Positioning System (GPS)[3], or distributed networked based techniques to obtain the physical position of a mobile node.

- **TCP and Reliable Transport:** The TCP protocol is originally designed for a fixed network to provide flow control and congestion control. Finding ways of enhancing it in order to provide efficient transport layer support in MANETs is an important problem.

- **Medium Access Control:** Developing efficient medium access protocols that optimize spectral reuse in MANETs is actively pursued by research community.

- **Radio Interface:** Since nodes in MANETs use wireless communication, better antenna design is critical to reduce interference during information transmissions.

- **Quality of Service:** Due to the highly dynamic topology and the low data rate in wireless links, provisioning Quality of Service (QoS)[4] is a complex problem in MANETs. Applications such as multimedia communication in MANETs require QoS support.

[3]Introduction of GPS: http://www.colorado.edu/geography/gcraft/notes/gps/gps_f.html
[4]QoS overview: http://www.objs.com/survey/QoS.htm

- **Power Management:** Since mobile nodes usually have low power supplies, power saving mechanisms are important to help prolong battery life and extent lifespan of MANETs.

Finding efficient solutions to these fundamental issues could significantly increase the survivability of MANETs.

1.2 Broadcast Algorithms in MANETs

Broadcast is a well-known *one-to-all* communication task, where one host wishes to send a message to all other nodes in the network. It has been widely used in both wired and wireless networks, for example, many applications and routing protocols in fixed networks use broadcast to send global information such as control and topology messages to all hosts in the network. Base stations in cellular networks also rely on broadcasting to page mobile users. In wireless ad hoc networks, broadcast also plays an important role. In MANETs, many routing protocols, such as Ad Hoc On-Demand Distance Vector (AODV) [12], Dynamic Source Routing (DSR) [13] and Optimized Link State Routing protocol (OLSR) [14, 15], depend on flooding mechanisms to broadcast data and control packets throughout the network in order to establish routes between each source-destination pair. Moreover, since the topologies of MANETs change rapidly, nodes in these networks have to generate and distribute control messages regularly to update connection states, and therefore, broadcast processes are expected to be conducted even more frequently than in other networks.

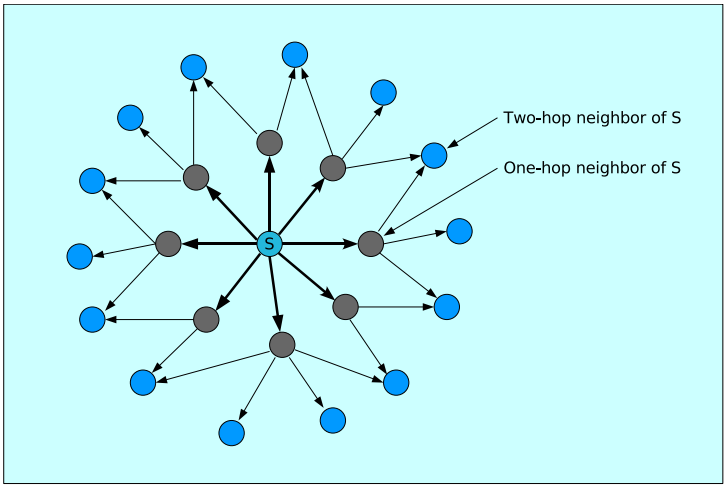


Figure 1.2: Basic flooding problem.

1.2.1 Blind Flooding Problem

The simplest way of broadcasting a packet is basic flooding or blind flooding [16], which allows each node to retransmit a packet to its neighbors only if it has not received this packet before. This rebroadcasting continues until all nodes in the network receive a copy of the packet. A prominent advantage of basic flooding is that it can always find the shortest path between sources and destinations since topology packets have been through every possible path in parallel. Due to this feature and also its computational simplicity, basic flooding is widely adopted in wired networks.

However, the basic flooding mechanism can trigger a large number of redundant packets forwarded in the network which may overwhelm a network that mainly relies on wireless links. Fig. 1.2 illustrates this problem. In the multi-hop wireless network shown, after source S sends out a broadcast packet, all the one-hop away nodes transmit copies of it at almost the same time to all two-hop neighbors of S. This results in overly redundant rebroadcasting (some nodes receive multiple copies of the same packet), contention, collision and significant energy consump-

tion, which are referred to as the broadcast storm problem [17]. Considering the restriction on energy and bandwidth of MANETs, the broadcast storm problem can have a significant impact on the network performance and energy efficiency. Therefore, there needs to be efficient broadcast schemes that can minimize redundant retransmissions while still guaranteeing the reliability of broadcasting.

1.2.2 Categories of Existing Approaches

To achieve efficient broadcasting and solve the broadcast storm problem, many approaches have been proposed [18, 19, 20]. In general, existing efficient broadcast protocols can be categorized into three groups: *probability-based methods*, *area-based methods* and *neighbor knowledge methods*.

The probability-based methods are similar to basic flooding, except that each node rebroadcasts packets with a predetermined probability. This mechanism might work in dense networks when multiple nodes have similar neighbor coverages, but will not have a significant effect in sparse networks. In the area-based methods, a node rebroadcasts a packet based on the distance between itself and the node from which that packet is received. A rebroadcast occurs only when the distance is longer than a predefined threshold, so that a larger additional area can be reached. However, the area-based methods do not consider whether some nodes actually exist within that additional area, which can lead to inefficient broadcasting. For the neighbor knowledge methods, it can be further classified as *neighbor-designated methods* (also referred as *source-dependent methods*) and *self-pruning methods* (also referred as source-independent methods). In the neighbor-designated methods, a node that transmits a packet specifies which one of its one-hop neighbors should forward the packet, while in the self-pruning methods, a node receiving a packet will decide

whether or not to transmit the packet by itself. It is noted that the existing probability and area based protocols are not reliable (i.e. a broadcast packet may not reach all nodes in the network under these protocols) since their performances largely depend on the predetermined parameters and thresholds, which may not be the best values for a given network. For this reason, a majority of the research work for improving the efficiency of broadcast in wireless networks focus on the neighbor knowledge methods.

1.3 Aims and Contribution of This Research

This research aims to develop new efficient broadcast algorithms for wireless ad hoc networks. The proposed algorithms need to be simple in both computation and communication load, and should be scalable and efficient in terms of minimizing redundant broadcast retransmissions in various situations.

This thesis first presents a comprehensive investigation of efficient broadcast algorithms in wireless ad hoc networks, especially for the Multipoint Relay (MPR) and connected dominating set (CDS) based algorithms. A thorough survey on these algorithms is conducted, which discusses details of their operations. Performance of the algorithms are also evaluated and compared in order to provide the reader clear guidelines for designing broadcast protocols in MANETs. Furthermore, three new efficient broadcast algorithms are also proposed in this thesis. Theoretical analysis is conducted to validate the algorithms and evaluate their performance boundaries. Simulation models are also created to test the proposed algorithms under various network scenarios. It is shown that the new algorithms outperform the existing ones reported in the research literature in many aspects including simplicity of computation and communication load, efficiency on reducing redundant transmissions and

scalability under network densities.

1.4 Outline of the Thesis

The remaining chapters are organized as follows. Chapter 2 presents a literature survey of related work. Chapter 3 introduces the detail of the new proposed algorithms. Theoretical analysis and evaluation of the proposed algorithms are presented in Chapter 4. Chapter 5 provides an extensive simulation study of the proposed algorithms. Finally, the conclusions of the research and recommendations of future work are presented in Chapter 6.

Chapter 2

A Survey of Efficient Broadcast Algorithms for MANETs

2.1 Introduction

Broadcasting is a well studied topic in MANETs, many algorithms have been proposed so far and they adopted various methods to achieve efficiency. Among those methods, Multipoint Relay (MPR) [15] and connected dominating set (CDS) [21] based approaches are popular and frequently used. In the literature, many broadcast schemes have been proposed based on the MPR and CDS methods due to their remarkable efficiency of reducing redundant retransmissions. These schemes are put forward to improve different aspects of broadcasting performance in MANETs such as limiting the number of generated forwarding nodes, collision avoidance, efficient power usage and quality-of-service (QoS).

Based on these two methods, three new efficient broadcast algorithms have been proposed in this research project. This chapter first introduces the concepts of

the MPR and CDS approaches in order to give readers the background knowledge. Then a comprehensive survey of leading existing MPR and CDS based broadcast algorithms in MANETs are presented.

2.2 Overview of the Multipoint Relay Method

Multipoint Relay (MPR) is a neighbor designated method that exhibits both efficiency and simplicity. The concept of the MPR was first introduced in The High Performance Radio Local Area Network (HIPERLAN) type 1 standard [22], which was a MAC layer protocol developed by the European Telecommunications Standards Institute (ETSI) to provide a substitute for wired LAN. It was then successfully extended to MANETs and effectively implemented in the OLSR routing protocol [14], which is a proactive routing protocol ratified as a Request for Comments (RFC) in the Internet Engineering Task Force (IETF) MANET chapter. The goal of the MPR is to reduce the flooding of broadcast packets in the network by minimizing redundant retransmissions locally. In the MPR method, each node in the network selects a subset of its one-hop neighbor nodes called *Multipoint Relays* (MPRs) as the forwarding node set to retransmit broadcast packets. Other nodes that are not MPRs can read but not retransmit broadcast packets. The MPRs guarantee that all two-hop neighbor nodes of each node receive a copy of the broadcast packets, and therefore, all nodes in the network can be covered without retransmissions by every single node.

An example of MPR flooding is shown in Fig. 2.1, where source node S in the center selects only a subset of its one-hop neighbors as MPRs to forward the broadcast messages, so that all two-hop neighbor nodes of S can be covered by the selected five MPRs. Upon receiving a broadcast message, a node forwards

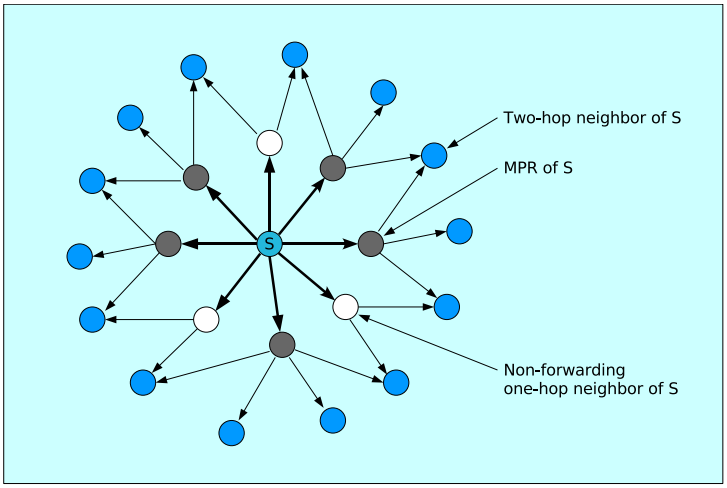


Figure 2.1: An MPR flooding example.

it if and only if the message is received for the first time and the sender of the message has selected the node as an MPR. This scheme can dramatically reduce the number of re-transmitters thus decreasing the rebroadcast and redundant messages in the network. Furthermore, the signaling messages disseminated throughout the network only contain the information of a node's MPR selectors, i.e. other nodes that have selected the node as their MPR. Therefore, only partial link information is included in the messages, which makes the overhead of control traffic relatively low. For these important merits, the MPR mechanism produces efficient routing schemes. It provides shortest-path routes for routing protocols while at the same time minimizing the flooding of broadcast messages and reducing the overhead of control traffic. For these reasons, the MPR scheme is also favored in other routing protocols such as the Multipoint Relay Distance Vector protocol (MPRDV) [23] and the MPR-based hybrid routing (MPR-HR) [24].

It is noted that the main gain obtained by introducing a set of MPRs is that: the broadcasting process can be completed by using only a small set of nodes in the network thus greatly reducing the redundant retransmissions. The smaller the MPR

set is, the fewer retransmissions that will occur. Unfortunately, it has been proven in [15] that finding a minimum size of an MPR set is NP-Complete [25], and only heuristic methods can be applied to find MPR set with good approximation. The rest of this section discusses the MPR selection heuristic presented in the OLSR routing protocol as described in [14], which is referred as the *original MPR heuristic*.

The original MPR heuristic follows a greedy algorithm [26] that works well for computing an MPR set. To select MPRs for a node x, first it is defined that the set of all one-hop neighbors of x is $N(x)$, and the set of all two-hop neighbors of x is $N_2(x)$. Also, it is defined that for each one-hop neighbor node y of x, its *out-degree* value is $D(y)$, which represents the number of two-hop neighbors of x that can be covered by y. Let the MPR set of node x be $\mathrm{MPR}(x)$. The heuristic of the MPR selection can be described as follows:

1. Start with an empty Multipoint Relay set $\mathrm{MPR}(x)$.
2. Calculate $D(y)$ for each node y in $N(x)$.
3. Add to $\mathrm{MPR}(x)$ the nodes in $N(x)$, which are the *only* nodes that can reach some nodes in $N_2(x)$. For example, if node a in $N(x)$ is the only neighbor of node b in $N_2(x)$, then add node a to $\mathrm{MPR}(x)$. Remove nodes from $N_2(x)$ which are now covered by nodes in $\mathrm{MPR}(x)$.
4. While there are still some nodes in $N_2(x)$ which are not yet covered by the nodes in $\mathrm{MPR}(x)$:
   (a) For each node in $N(x)$ which is not yet selected as the MPR, calculate the number of the two-hop neighbor nodes of x it can cover which are not yet covered by the nodes in $\mathrm{MPR}(x)$.

(b) Add a node to $\mathrm{MPR}(x)$ which covers maximum number of remaining two-hop neighbors of x. In case of multiple choices, select the node as MPR whose $D(y)$ is larger. Remove nodes from $N_2(x)$ which are now covered by nodes in $\mathrm{MPR}(x)$.

5. To optimize the $\mathrm{MPR}(x)$, remove a node in $\mathrm{MPR}(x)$ if all the two-hop neighbor nodes it covered can also be covered by the remaining nodes in $\mathrm{MPR}(x)$. Therefore, the lack of this node will not affect the overall coverage of the nodes in $\mathrm{MPR}(x)$.

In order to recognize neighbor nodes and calculate $D(y)$ for each one-hop neighbor, hello messages are exchanged between one-hop neighbors periodically. A hello message generated by a node may contain information such as node IDs (can be IP addresses), IDs of MPRs it has selected, and all related information about its one-hop neighbors. These hello messages are exchanged in fixed time intervals or in event driven manner so that necessary information for the MPR calculation can be obtained and the status of the network can also be updated.

2.3 Connected Dominating Set

The concept of the connected dominating set (CDS) comes from the graph theory [21]. It defines a set of nodes for a given connected graph. The definition of a CDS can be described as follows:

Definition 2.1. *For a given connected graph (network) $G = (V, E)$, where V is the set of vertexes (nodes) and E is the set of edges (links) that provides the available communications. a dominating set (DS) is a subset V' of V, where for each vertex*

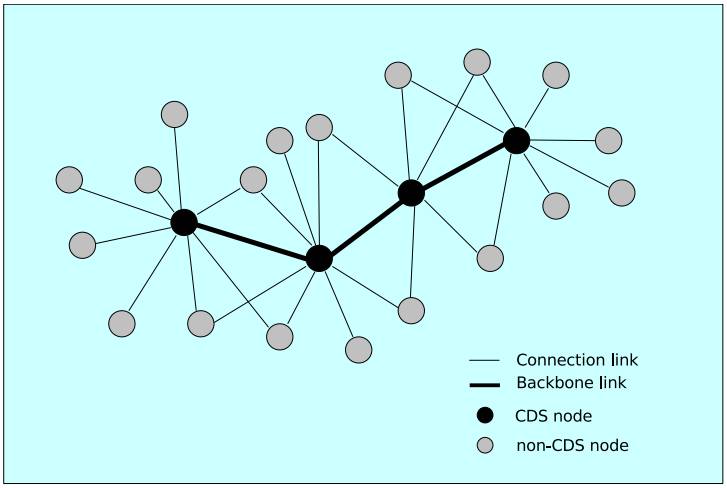


Figure 2.2: A possible CDS in a given graph.

u of V, u is either in V' or at least one neighbor vertex of u is in V'. A DS is called a CDS if the sub-graph induced by the vertexes in the DS is connected.

From the definition it can be seen that, nodes in a CDS is capable of reaching every other node in the network and they themselves are also connected, i.e. there is no separated island node(s) in the CDS. A possible example of a CDS is shown in Fig. 2.2. In this network, nodes in black form a CDS and they are connected through the bold lines, which represent the backbone of the network. All other nodes that are marked in gray can be reached by the nodes in the CDS. In such a network, broadcast messages only need to be relayed by nodes in the CDS in order to reach all the nodes in the network, and thus dramatically reduce the redundant transmissions.

Recently, some researches have proposed to construct a virtual backbone or a *spine* [27, 28, 29] by nodes in a connected dominating set (CDS) to improve performances of routing and broadcasting protocols in wireless ad hoc networks. Since only nodes in a CDS broadcast messages, it is desirable to find a CDS of small number of nodes for a given network. However, finding a minimum connected

dominating set (MCDS) is a well-known NP-complete problem [25] in the graph theory, and therefore, heuristic methods are normally used to obtain sub-optimal results. Generally, a CDS can be constructed by using either global network information or localised information, and the CDS calculation process can be done in either a centralized way or a distributed way. However, due to the characteristics of wireless ad hoc networks, it is hard to obtain and maintain global network information. Also, it is not computationally efficient to conduct a CDS calculation in a single mobile node. Therefore, most of the proposed approximate algorithms are distributed and based on local information only. This thesis only focuses on distributed and localized broadcast algorithms.

2.4 Costs of Broadcast Algorithms

In order to evaluate the performance of broadcast algorithms, costs of the algorithms are defined here and they can be summarized as follows:

- **time complexity:** In order to calculate the set of forwarding nodes, a certain number of processes need to be done. These processes may take different time to complete depending on the algorithms they used. For each broadcast algorithm, the time required to complete the forwarding node set calculation is referred to as the *time complexity* or *computation complexity*, which can be used to evaluate the simplicity of an algorithm. An algorithm that requires long time to run may be too complex to be implemented in a node. Furthermore, when the network topology changes rapidly, the frequency of the forwarding node calculations also increases, and thus the time required to finish the calculation of a complex algorithm may become extremely long.

Therefore, time complexity is critical when designing broadcast algorithms in MANETs.

- **Message complexity:** For each algorithm, a number of hello messages (signaling messages) need to be exchanged between nodes during the forwarding node set calculation. The signaling messages contain necessary information for an algorithm to implement the forwarding node set calculation. The amount of communication required for a broadcast algorithm is referred to as the *message complexity* or *communication complexity* of the algorithm. Frequent signaling message exchanges will consume the limited bandwidth in wireless networks and also accelerate the energy consumption of mobile nodes. Hence, low message complexity is one of the important criteria for identifying efficient broadcast algorithms in MANETs.

- **Signaling message size:** Similar to the message complexity, the size of signaling messages also affects the performance of broadcast algorithms. Since long messages require more time to be transmitted and processed, a broadcast algorithm that requires too much information in its signaling messages may increase overall end-to-end delay in the network, which significantly influences the quality-of-service (QoS) performance in the network. Furthermore, large messages also have higher probabilities to get corrupted in a MANET's wireless environment, and retransmissions waste even more time and energy.

- **Approximation ratio:** Since only approximate algorithms are used to calculate the MPR set and the CDS in the network, how close their results to the optimal ones is also an important criterion to judge their efficiency. The approximation ratio of an approximate algorithm is the largest ratio between the result obtained from the algorithm and the optimal result. Detailed infor-

mation on the approximation ratio can be found in [30].

- **Information range:** The surveyed algorithms are all distributed, and therefore, they need to collect neighbor information in order to conduct calculations. Referring to the example of the original MPR algorithm shown in Fig. 2.1, source node S has to know all neighbors in its one-hop neighborhood in order to identify all its two-hop neighbors and decide which one-hop neighbor has the largest out-degree value. In other words, node S has to obtain information of its one-hop and two-hop neighbors in order to proceed the MPR calculation. Such information requirement is referred to as the *information range*, and for the original MPR algorithm, it has an information range of two hops. Generally, the larger information range an algorithm requires, the more time and message exchanges it will need. Hence, an *information range* up to four hops may not be efficient since messages need long time to be transmitted to the source node, and the information they carry may be outdated by then.

- **Source dependent:** Some broadcast algorithms are *source dependent*, that is, they need to know from which node the packet was received in order to determine whether or not to retransmit this packet. Such requirement increases the complexity of both the message sending and receiving process in an algorithm. Thus, it is desirable for a broadcast algorithm to be source independent.

Above costs define the overall efficiency, scalability and stability of a broadcast algorithm. Merits and drawbacks of each algorithm can be clearly highlighted by analysing these costs. In the following sections, details of the surveyed broadcast algorithms are presented and their performances are also evaluated based on the above costs.

2.5 MPR Based Algorithms

Many proposed broadcast algorithms in MANETs are based on the original MPR heuristic [14]. In this thesis, the MPR based algorithms are classified into two groups regarding their objectives. Table 2.1 shows the classification and their objectives.

Table 2.1: Classification of MPR based algorithms

Groups	Objectives
Pure MPR algorithms	These algorithms are still based on the concept of the original MPR heuristic, while several extensions are applied in order to improve some specified performances such as reducing the number of generated forwarding nodes, collision avoidance and power usage efficiency.
QoS-based MPR algorithms	These algorithms consider the quality-of-service (QoS) constraints in the network, and they select MPRs that meet some QoS requirements, so real-time applications such as voice and video communications can be better supported by providing paths with larger bandwidth and lower delay.

2.5.1 Pure MPR Algorithms

Based on the concept of the original MPR heuristic, pure MPR algorithms [31, 32, 33, 34] try to modify the MPR selection procedures in order to choose nodes as MPRs that have some special effects such as minimum collisions and efficient power consumption.

Mans and Shrestha proposed four algorithms in [31, 32] which aim to reduce the cardinality of the MPR set and limit collisions in the network.

The first algorithm namely the *in-degree MPR* (ID-MPR) indicates that the

complexity of the original MPR heuristic is mainly due to the maximum value of the *out-degree* D of one-hop neighbor nodes. Considering this property, a new concept called *in-degree* denoted as Din was presented in this heuristic as a new criterion for MPR selection. The value of the in-degree of a node y is the number of shared neighbors between node y and node x, where x is a one-hop neighbor of source node S, and y is a two-hop neighbor of S. Due to the intrinsic connectedness, wireless networks may be dense and highly clustered. In such case, it is believed that the maximum value of the in-degree $Din(y)$ of a two-hop neighbor node y is likely to be smaller than the maximum value of the out-degree $D(x)$ of a one-hop neighbor node x, and thus, when applying the in-degree to the MPR selection, the computational complexity might be lower than the original MPR heuristic.

The operation of this algorithm can be concluded as follows:

1. For a source node S that needs to calculate the MPR set, apply first three steps used in the original MPR selection heuristic (Section 2.2) to cover some two-hop neighbors that are solely covered by some one-hop neighbors.

2. If there are still some uncovered two-hop nodes, randomly pick up a node among those uncovered two-hop nodes, among all one-hop neighbor nodes that can cover this two-hop node and have not been selected as MPRs by the source node S, select a node as an MPR that has minimum number of uncovered two-hop neighbors.

3. Repeat this step until all two-hop neighbors of S have been covered.

Fig. 2.3 depicts an example of the in-degree MPR algorithm. In this network, one-hop neighbor nodes 2 and 7 will be chosen as MPRs first since they solely cover two-hop nodes b and k respectively. Among the uncovered two-hop neighbors, the

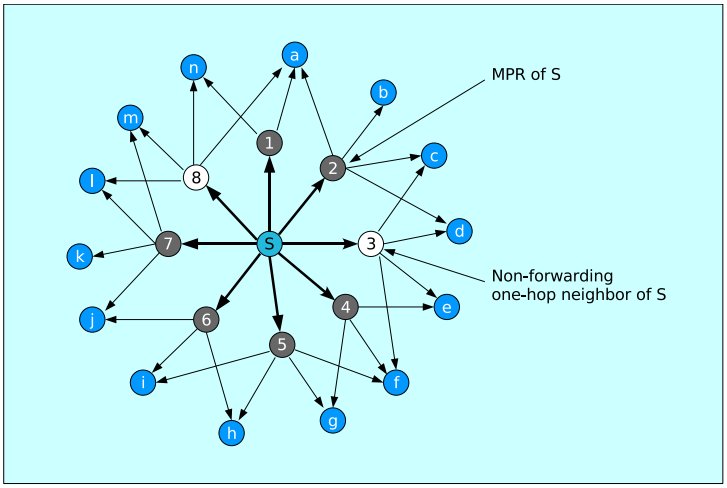


Figure 2.3: An example of the in-degree MPR algorithm.

algorithm will randomly pick up a node (assume node i) to start the MPR calculation. Based on the selection process, node 6 will be chosen as an MPR because it covers less number of uncovered two-hop neighbors of S. The remaining MPRs are also selected by following the same process when two-hop neighbor nodes f, n and e are randomly chosen.

The initial aim of this algorithm is to reduce the computational complexity by introducing the in-degree concept to the original MPR heuristic. It estimates that the in-degree of each two-hop node of source S is a smaller value compared to the out-degree of each one-hop node of S, so that less time is spent on the MPR calculation for each two-hop neighbor of S. However, this scheme increases the size of the MPR set. As shown in Fig. 2.3, the number of MPRs in this scheme is six, whereas only five MPRs (node 2, 4, 5, 7 and 8) are chosen when the original MPR heuristic is used. The increase of MPR nodes is caused by the iterative step, which selects an one-hop neighbor that covers minimum number of uncovered two-hop neighbors of source S. It is believed that a possible improvement of this algorithm is to use maximum number of uncovered two-hop neighbors instead of minimum to enlarge

the coverage of an MPR, and thus less number of MPRs are needed to cover all two-hop neighbors of the source node.

The second algorithms proposed by Mans and Shrestha, referred to as the *minimum overlapping MPR* (MO-MPR), tries to minimize overlaps between MPRs. The overlap is defined as the shared two-hop neighbors covered by two or more MPRs. For example, in Fig. 2.3, the overlaps of MPRs 5 and 6 are two-hop nodes i and h, since they are covered by both of the MPRs. The overlaps are detrimental to the message reception when considering the signal interference. The received signal at node i and h may be corrupted if both MPRs broadcast messages simultaneously, and their messages interfere each other. To reduce overlaps, the heuristic tries to spread MPRs as evenly as possible around the source node, and thus limits the overall interference in the network.

In this algorithm, instead of using the out-degree value, a covering ratio of a one-hop neighbor is introduced to determine an MPR set. The covering ratio is defined as the ratio of covered two-hop neighbors over uncovered two-hop neighbors that a one-hop neighbor has. Similar to the in-degree algorithm, the minimum overlapping MPR algorithm also follows the first three steps used in the original MPR heuristic. Then, if there are still some uncovered two-hop nodes, the algorithm chooses a node with the minimum covering ratio as an MPR among the one-hop neighbors that are not selected as MPRs. If multiple choices exist, it randomly picks one as an MPR. The above steps are repeated until all two-hop neighbors of the source node are covered. Fig. 2.4 illustrates an example of the minimum overlapping MPR algorithm.

This algorithm cannot reduce the maximum amount of overlaps per node due to the fact that the overlaps per node mainly depends on the topology which can

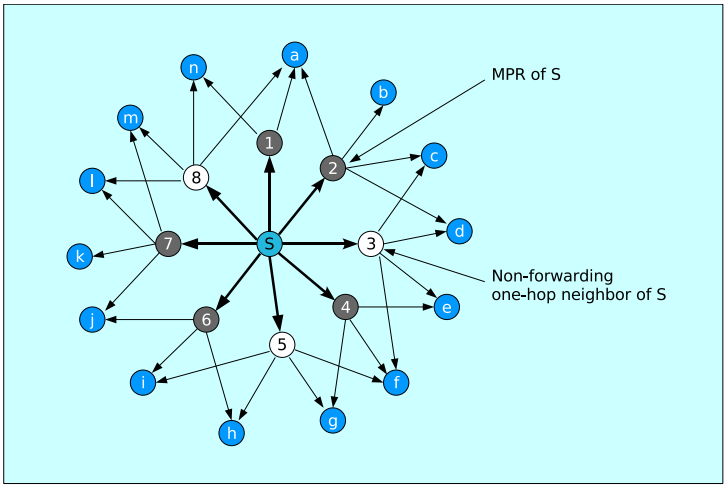


Figure 2.4: An example of the minimum overlapping MPR algorithm.

change arbitrarily. However, it is possible to limit the impact of overall overlaps in a network. As shown in Fig. 2.4, nodes a and j are overlapping nodes, whereas node a, f, g, i and m are the overlapping nodes when the original MPR heuristic is applied to the same network and node 2, 4, 5, 7 and 8 are selected as MPRs accordingly. It is obvious that the number of overlaps in the network are reduced. However, the algorithm may increase the number of MPRs in some scenarios. For instance, a node is chosen as an MPR by the source node even if it has the smallest coverage of two-hop nodes among all the one-hop neighbors of the source. This leads to more time to finish calculating an MPR set, and more number of MPRs are generated. Another drawback of this algorithm is that, there may be numerous nodes with the same covering ratio at the beginning of the calculation. In this case, the algorithm randomly chooses one node as an MPR, which may not have a large coverage of the uncovered two-hop nodes. The worst case scenario is that all randomly-selected MPRs cover the smallest number of uncovered two-hop nodes. One possible solution for this problem is to choose a node as the MPR that can cover the highest number of uncovered two-hop nodes if multiple nodes have the same covering ratio.

For the sake of minimizing overlaps without increasing the number of MPRs, two algorithms called, the *prioritized MPR* (P-MPR) and the *random prioritized MPR* (RP-MPR) are proposed. Both of them follow the same steps of the original MPR heuristic except for the tie breaking procedure. In the P-MPR algorithm, when there are multiple one-hop nodes that cover the same number of uncovered two-hop nodes, instead of using the maximum out-degree, a node with a minimum out-degree is selected as an MPR. In the RP-MPR algorithm, while multiple choices exist, it randomly chooses a node as an MPR.

The P-MPR and PR-MPR algorithms are combinations of the heuristic used in the original MPR and minimum overlapping MPR algorithms. In the heuristic of the original MPR, the reason to use the maximum out-degree as the tie breaker aims to add some redundancy to the network, through this, nodes can still be covered even if some MPRs are temporarily away. This strategy can stabilize the performance of the network when the topology changes rapidly. However, such redundancy may cause a large number of overlaps thus increasing the number of collisions. The P-MPR algorithm tries to minimize overlaps in the network by sacrificing the redundancy in the network while the PR-MPR algorithm aims to balance both properties.

Lipman proposed a distributed broadcast algorithm referred to as the *utility-based Multipoint Relay Flooding* (UMPR) [33]. The algorithm aims to reduce unnecessary retransmissions by limiting the rebroadcasting to only essential nodes in a similar fashion to the original MPR heuristic. Furthermore, it intends to extend the lifespan of the network by fully utilizing the energy of nodes. In the UMPR algorithm, a network is assumed to be heterogeneous. Wireless mobile devices can have different characteristics even if all devices consist identical hardware. By considering this characteristic, the UMPR algorithm tries to distribute the broadcast load to the *most suitable* nodes, so that the overall energy in the network can be efficiently

used. In order to decide the desirability of a node, the UMPR algorithm calculates a *forwarding utility* for each one-hop neighbor of source node S. A forwarding utility U_f is a function that consists of an one-hop node's power utility U_p and neighbor utility U_n. The function is defined as follows:

$$U_f = U_p U_n \tag{2.1}$$

The power utility U_p represents the value of the remaining power of a node. The larger the value is, the more remaining power a node has. However, how to monitor the remaining power of a device is not presented in the paper. The neighbor utility U_n represents the ratio of uncovered two-hop nodes over all the two-hop nodes that a one-hop neighbor node covers. The value of forwarding utility is updated each time when a node is allocated into the MPR set.

The forwarding node calculation process of the UMPR algorithm is still based on the original MPR heuristic. It applies the first three steps used in the original MPR heuristic. When there are still some uncovered two-hop nodes, from those one-hop neighbors of source S that have not yet been chosen as MPRs, the UMPR algorithm selects a node as an MPR that has the highest forwarding utility value. This step is repeated until all two-hop neighbors are covered.

The UMPR algorithm can achieve efficient use of the overall energy in a network by choosing forwarding nodes with higher remaining power. Furthermore, the neighbor utility U_n makes the UMPR algorithm tend to choose nodes that are less overlapping thus reducing the number of collisions in the network. Due to these advantages, the UMPR can provide better performance than the original MPR scheme in terms of prolonging lifespan of MANETs. However, the scheme may increase the cardinality of the generated MPR set. This is due to two reasons. First, nodes

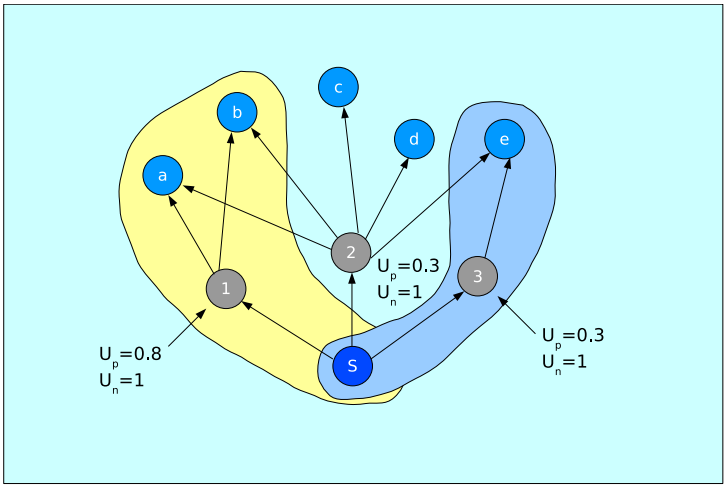


Figure 2.5: An example of the UMPR algorithm.

with more remaining power may not cover many uncovered two-hop nodes. Second, the neighbor utility U_n may result in more MPRs in the network. An example is shown in Fig. 2.5. Assume that all three one-hop nodes of S initially have the same neighbor utility U_n. Based on the forwarding utility function U_f, node 1 will be chosen as an MPR first since it has the highest remaining power. Then the U_n is recalculated for the residual one-hop nodes. In this case, node 3 has a higher U_n than node 2 and it is selected as an MPR. Finally, in order to cover all two-hop neighbors of S, node 2 is selected as the MPR. However, it is noted that a possible smaller MPR set in this network should only contain node 2 since it can sufficiently cover all the two-hop nodes of S.

Lipman later proposed another broadcast algorithm called the *utility-based flooding* (UBF) [34], which intends to extend the UMPR algorithm to provide full resource awareness. It points out a problem in the UMPR algorithm that nodes selected by the first three steps used in the original MPR heuristic tend to dominate the MPR set and limit the use of the forwarding utility. The UBF algorithm avoids this problem by eliminating these steps, and therefore, nodes selected as MPRs are

solely based on the forwarding utility. It guarantees that each node in the MPR set is selected based on the remaining energy, so that a more energy efficient broadcast can be achieved.

The MPR selection process of the UBF algorithm differs from the one of the UMPR algorithm in the first step, where source node S chooses a one-hop node with the highest forwarding utility as an MPR. This step is repeated until all two-hop nodes are covered. Although the UBF algorithm gains more resource awareness, it still has the same drawback as the UMPR algorithm. Furthermore, without applying the first three steps, which are used to optimize the calculation of the algorithm, the UBF may take more time to calculate an MPR set.

2.5.2 QoS based MPR Algorithms

Quality-of-service (QoS) is an important issue and it has been deployed in traditional wired networks for more than a decade. Inevitably, it will also be a key feature in MANETs to provide multimedia service. To support QoS, the link state information such as bandwidth and delay should be available and manageable. This requires broadcast protocols to be able to efficiently disseminate the QoS information throughout the wireless network. Regarding the original MPR heuristic, where MPRs are chosen based on non-QoS criteria and each MPR can only propagate information of links between it and its MPR selectors, good quality links may be hidden to other nodes in the network. Fig. 2.6 illustrates this problem in the original MPR protocol. The number above each link represents its corresponding bandwidth. In such a network, node a selects node b as an MPR since it covers more uncovered two-hop neighbors and it has a smaller node ID. Following the same heuristic, node b chooses node f as an MPR. Therefore, g knows that it can reach a

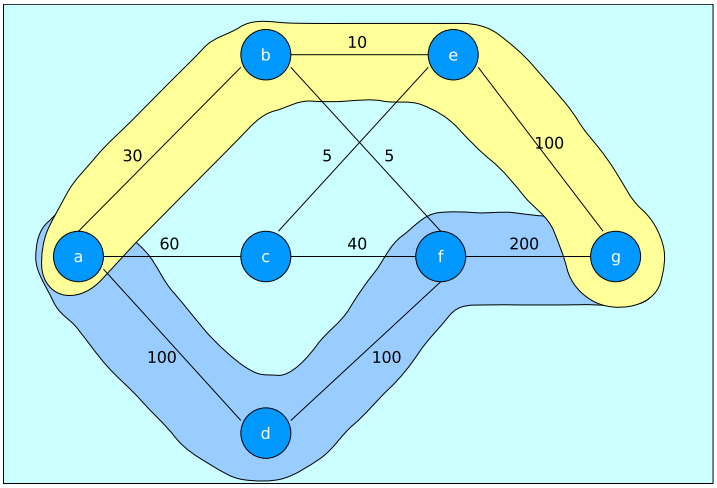


Figure 2.6: QoS problem in the original MPR heuristic.

via the route $\{g, f, b, a\}$ which has the bottleneck bandwidth of 5. However, it is obviously that a better route should be $\{g, f, d, a\}$ which has the bottleneck bandwidth of 100. This high bandwidth route is hidden from node g when using the original MPR heuristic. Therefore, the MPR selection has to consider QoS information such as bandwidth and delay in order to provide suitable links for some specific applications. This section discusses the QoS-based MPR algorithms that aim to select MPR sets based on some QoS requirements.

Badis et al. [35] proposed two algorithms for selecting MPRs based on QoS measurements. The purpose of these algorithms are to extend the QoS routing protocol proposed in [36]. The essence of these two algorithms is to utilize QoS conditions, such as the bandwidth and the delay of links between one-hop neighbors and the source node, to select MPRs that can provide better QoS. Additional QoS information is piggybacked into hello messages and exchanged between neighbors, and thus no extra control messages are generated.

The first proposed algorithm, referred to as the *QoS-based MPR–1* (QMPR–1), follows the same steps as the original MPR heuristic, but it modifies the tie-breaking

procedure in order to generate QoS prioritized MPRs. Instead of a maximum node out-degree, a node with higher bandwidth is chosen when a tie happens during the MPR selection. In case of another tie in the above step, a node with minimum delay is then selected. This algorithm has a higher chance to select MPRs with larger bandwidth, but the improvement is only marginal, and thus, it cannot guarantee to find the optimal links. As shown in Fig. 2.6, path $\{g, f, c, a\}$ will be revealed based on the QMPR–1 algorithm because c has a larger bandwidth b. However, this result is still not the best one in terms of providing the highest bandwidth.

The second algorithm namely the QMPR–2, tries to improve the QoS performance of the QMPR–1. It also follows the same steps as the original MPR heuristic, but it selects nodes with higher bandwidth as MPRs, and the delay is used whenever there is a tie. In case of another tie in the above step, a node that covers the most number of uncovered two-hop neighbors is chosen. This algorithm enlarges the effect of the QoS criteria in the MPR selection. Hence, more MPRs can be chosen based on the QoS conditions, and consequently, a better chance can be achieved to find the optimal links between a given pair of source and destination. As can be seen in Fig. 2.6, the path with the highest bandwidth is found finally by using the QMPR–2 algorithm.

These two algorithms, especially for QMPR–2, can find MPRs with better QoS conditions thus providing suitable links for QoS requirements. However, both algorithms may increase the number of MPRs. This is due to the fact that MPRs which have higher bandwidth or lower delay might cover few uncovered two-hop nodes, and hence, more MPRs have to be selected to covered all two-hop nodes of the source. As shown in Fig. 2.6, nodes d and g are selected as MPRs of node f in the QMPR–2 algorithm, however, node b alone is sufficient to cover all two-hop neighbors of f and will be selected as an MPR in the original MPR heuristic. It

is also noted that not all MPRs are selected based on the QoS conditions. This is because that the initial phase of the original MPR heuristic is applied in both algorithms, and it can generate most of the MPRs. Therefore, both algorithms can only have effect on a small part of the MPR set.

In [37], Ge et al. tried to integrate the QoS feature into the OLSR routing protocol [15]. They also investigated the limitation of the original MPR heuristic and realized that good quality links can be hidden to other nodes in the network. Considering this limitation, three extended MPR selection algorithms are proposed to compute the MPR set based on QoS criteria. The first two algorithms are similar to Badis's but without considering the delay. The third one, referred to as the *QoS-based MPR–3* (QMPR–3) further improves the QoS performance. It points out a drawback in the QMPR–2 algorithm that not all two-hop neighbors have optimal links to reach the source node. Referring to Fig. 2.6, it is observed that node g will select node f as the MPR based on the QMPR–2 algorithm. Hence, node b will have the knowledge that it can reach g via f after f relays g's broadcast messages. Obviously, a larger bandwidth link $\{b, e\}$ is hidden from node b. The QMPR–3 algorithm solves this problem by using a heuristic similar to the in-degree MPR discussed previously.

The idea of the QMPR–3 algorithm is to let all two-hop nodes have an optimal bandwidth path through MPRs to the source node. Here, the optimal bandwidth path is the path with the highest bottleneck bandwidth. For each two-hop node x, source node S chooses a one-hop neighbor node as the MPR if it covers x, and the bottleneck of the path is the largest among all available paths from x to S. Each two-hop node has to go through this process until is finds an optimal path to the source node.

The QMPR–3 algorithm further increases the chance of finding a route with higher bandwidth since each two-hop node is linked to the source node using an optimal path. However, it generates more number of MPRs than the other two QoS-based MPR algorithms, which results in more retransmissions in a network. One can possibly think that in the worst case, every one-hop neighbor of the source node can be chosen as the MPR for different two-hop nodes. It is estimated that it will be too costly to ensure that every two-hop node has an optimal path to the source node. An alternative way is to consider a weighted value, which can be a ratio of the overall bandwidth of all links that a node has on the number of links. A one-hop node with the highest weight should be selected as an MPR. Therefore, two-hop nodes may have a higher chance to obtain larger bandwidth paths to the source node via fewer MPRs.

2.5.3 Summary of the MPR Based Algorithms

In this thesis, the MPR based algorithms are classified into two groups based on their objectives. Generally, algorithms in the Pure MPR group aim to find a small set of one-hop neighbor nodes based on the original MPR heuristic to forward broadcast messages, so that all nodes within two hops from the source node can receive the messages eventually. Whilst, algorithms in the QoS-based MPR group try to revise the original MPR selection heuristic to achieve QoS awareness. Among them, the QMPR–3 algorithm has a better performance of finding the optimal routes in the network. However, it also generates more MPRs compared with other QoS-based MPR algorithms thus increasing the overall retransmissions in the network. In order to provide a clear comparison of the MPR based algorithms, the costs of the algo-

rithms are shown in Table 2.2, where Δ represents the maximum node degree[1], Φ represents the maximum number of two-hop neighbors of a node in a given network, M represents the number of MPRs selected by a node, and n is the total number of nodes in the network. The above symbols will be used through this thesis.

Table 2.2: Cost comparison of the MPR based algorithms.

Algorithms	**Information range**	**Source dependent**	**Time complexity**	**Message complexity**	**Message size**	**Approximation ratio**
MPR [15]	2 hops	Yes	$O(3\Delta M + \Delta)$	$O(n)$	$O(\Delta)$	Δ
ID-MPR [31]	2 hops	Yes	$O(2\Delta^2 + \Delta)$	$O(n)$	$O(\Delta^2)$	Δ
MO-MPR [31]	2 hops	Yes	$O(3\Delta M + \Delta)$	$O(n)$	$O(\Delta)$	Δ
P-MPR [31]	2 hops	Yes	$O(3\Delta M + \Delta)$	$O(n)$	$O(\Delta)$	Δ
RP-MPR [31]	2 hops	Yes	$O(3\Delta M + \Delta)$	$O(n)$	$O(\Delta)$	Δ
UMPR [33]	2 hops	Yes	$O(3\Delta M + \Delta)$	$O(n)$	$O(\Delta)$	Δ
UBF [34]	2 hops	Yes	$O(3\Delta M)$	$O(n)$	$O(\Delta)$	Δ
QMPR–1 [35]	2 hops	Yes	$O(3\Delta M + \Delta)$	$O(n)$	$O(\Delta)$	Δ
QMPR–2 [35]	2 hops	Yes	$O(3\Delta M + \Delta)$	$O(n)$	$O(\Delta)$	Δ
QMPR–3 [37]	2 hops	Yes	$O(2\phi\Delta)$	$O(n)$	$O(\Delta)$	Δ

As can be seen from Table 2.2, all the reviewed MPR based algorithms have the information range of two hops, which means that nodes in these algorithms need knowledge of their one-hop and two-hop neighbors, and therefore, each node in the network has to include its one-hop neighborhood information in its signaling messages. It is also noted that, all the algorithms are source dependent, hence each node needs to check from where a packet was sent and whether the sender has selected it as an MPR, which which increases the complexity of the algorithm.

Due to the fact that all the reviewed algorithms are derived from the original MPR heuristic, they all have similar time and message complexities. In the original MPR heuristic, the time used for calculating an MPR set is dominated by the iterative steps (refer to Section 2.2). It is assumed that for the third step of the orig-

[1]*Node degree* is the number of one-hop neighbors of a node, and *maximum node degree* (Δ) is, in a given network, the node degree of the node which has the largest number of one-hop neighbors.

inal MPR heuristic, $O(\Delta)$ time is needed at most to find out all one-hop neighbors that solely cover some two-hop nodes. In step 4, the algorithm iteratively calculates the remaining one-hop neighbors until all two-hop nodes are covered. The two sub-steps in step 4 need $O(\Delta)$ and $O(2\Delta)$ time respectively for each round, and since the iteration process takes M rounds to complete, the step 4 needs at most $O(3\Delta M)$ time in total to finish. Therefore, the overall time complexity of the original MPR heuristic is $O(3\Delta M + \Delta)$. It is noted that the in-degree MPR algorithm has the highest time complexity in the pure MPR group. This is caused by applying a different iterative step, which may run Δ rounds and take $O(2\Delta^2)$ time in total to finish in the worst case. The two QoS-based MPR algorithms proposed by Badis et al. only modify the tie-breaking procedure and the MPR selection criterion of the original MPR heuristic, and therefore, they have the same time complexity as the original MPR heuristic. The QMPR–3 algorithm proposed by Ge et al. uses a different strategy to calculate an MPR set in the network. For each two-hop node of the source, the algorithm computes an MPR to setup a path to the source. This step can run as many as Φ times. However, due to the lack of details of the QMPR–3 algorithm, its time complexity cannot be explicitly deduced. It is assumed that the algorithm runs based on the following steps: first, for each two-hop node of the source, the bottleneck bandwidth of all available paths to the source node are calculated. This step takes $O(\Delta)$ time to complete in the worst case when a two-hop node is reachable by all one-hop nodes. Second, for each two-hop node, the algorithm picks up a node as the MPR that can provide the largest bottleneck bandwidth. This step can be finished in $O(\Delta)$ time. Since both steps have to be operated for all two-hop neighbors of the source node, the total time complexity of the algorithm can be $O(2\Phi\Delta)$.

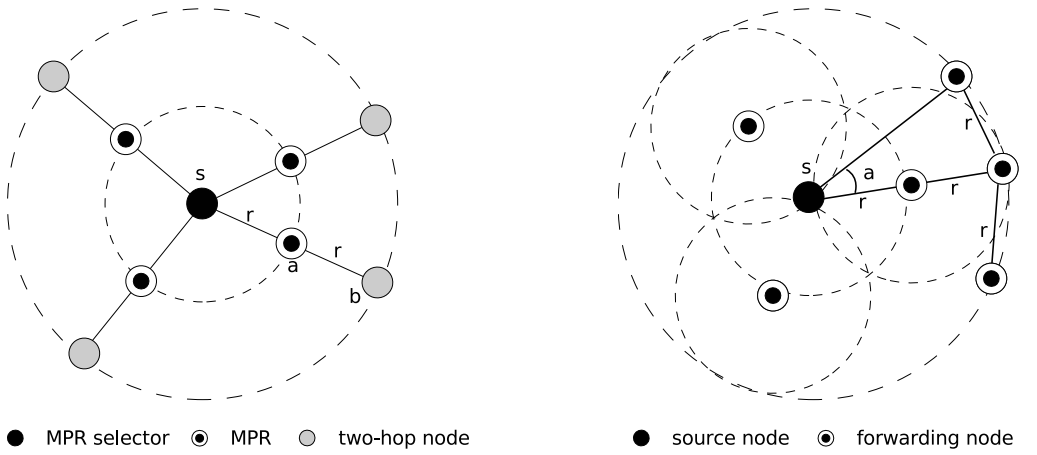
In the original MPR heuristic, each node needs to send out a signaling message

to its one-hop neighbors to inform its one-hop neighborhood information. After the MPR set calculation, the MPR selectors also send out a signaling message to inform one-hop nodes that have been selected as MPRs. Therefore, each node only sends a constant number of signaling messages during the MPR set calculation, and hence, the message complexity of the original MPR heuristic is $O(n)$. Since no extra signaling messages are required by the surveyed MPR based algorithms, they all have the same message complexity as the original MPR heuristic.

It is noticeable that almost all the surveyed MPR based algorithms have the same signaling message size of $O(\Delta)$. This is because the algorithms require to include the node IDs of one-hop neighbors in their signaling messages, where the number of one-hop neighbors of a node can be as many as Δ. The exceptional case is the in-degree MPR algorithm. whose message size is $O(\Delta^2)$. This is due to the fact that in order to calculate the in-degree value of a two-hop neighbor node, each node in the network has to include IDs of its one-hop neighbors and also IDs of their one-hop neighbors in the signaling messages, and thus dramatically increase the signaling message size.

Since the problem of finding a minimum MPR set for a given network is an NP-complete [25] problem, all the MPR based algorithms are approximation algorithms. It has been proven in [15] that the approximation ratio of the original MPR heuristic is $\log(n)$, where n is the total number of nodes in the network. The result indicates that the size of the MPR set generated by the original MPR heuristic is at most $\log(n)$ times larger than the optimal solution.

In fact, the approximation ratio of the original MPR heuristic should be Δ, which can be proven by referring to Fig. 2.7. In Fig. 2.7(a), all one-hop neighbors of source node S are located exactly at distance of r from S, and all two-hop neighbors



(a) Total Δ number of MPRs are selected by s using the original MPR heuristic.

(b) Only 15 forwarding nodes are selected by s in an optimal case.

Figure 2.7: An example of the worst scenario in the original MPR heuristic.

of S are located at the distance of $2r$ from S, where r is the transmission range of each node in the network. In such a case, each one-hop neighbor solely connects to a two-hop neighbor of S, so Δ MPRs are selected by S based on the original MPR heuristic. However, an optimal solution can be found in the same network as shown in Fig. 2.7(b). It can be seen that only three forwarding nodes are selected to cover the one-hop neighborhood of S. To cover two-hop neighbors of S, it is sufficient to select the two-hop nodes at distance of $2r$ from S as forwarding nodes whose distance is r between each other. Since the angle α shown in Fig. 2.7(b) is around 29 degree, there are at most 12 forwarding nodes selected in the two-hop neighborhood of S. Therefore, only 15 forwarding nodes are needed to cover all nodes within two-hop distance from S, so the approximation ratio of the original MPR heuristic is Δ. Considering that all the MPR based algorithms use a similar heuristic to generate MPR sets in the network, their approximation ratios should be the same as Δ.

2.6 CDS Based Algorithms

In the literature review of efficient broadcast algorithms for MANETs, many proposed algorithms were based on constructing a *connected dominating set* (CDS) in the network to minimize the redundant broadcasting. However, finding a minimum CDS (MCDS) in a given network is a well known NP-complete [25] problem, and therefore, only approximation algorithms can be used to achieve sub-optimal solutions. Generally, the existing CDS based broadcast algorithms can be divided into two groups. The first one constructs a CDS directly and then reduces its size. The algorithms in this group is referred to as the *direct-CDS based* algorithms. The second one constructs a *dominating set* (DS) (refer to Definition 2.1) first and then connects to nodes in the DS. In essence, the algorithms in this group first cluster the network by electing some *cluster-heads* (also known as *dominators*), which comprise the DS. Then, the *connectors* are generated to link to the dominators and form a CDS in the network. The algorithms in this group is referred to as the *cluster-CDS based* algorithms. This section reviews some selected leading CDS based algorithms in both groups. The classification of the CDS based algorithms is shown in Table 2.3.

Table 2.3: Classification of CDS based algorithms

Groups	**CDS generation procedure**
Direct-CDS algorithms	Construct a CDS in the network initially, and then reduce the size of it.
Cluster-CDS based algorithms	Cluster the network first by constructing a DS, and then link to nodes in the DS by generating connectors.

2.6.1 Direct-CDS Based Algorithms

Adjih et al. proposed an algorithm in [38] called the *MPR-based Connected Dominating Set* (MPR-CDS) to compute a CDS in a given network. It forms a CDS based on the existing MPR sets generated by using the original MPR heuristic. The algorithm points out that the idea of the original MPR technique is to compute some local CDSs, where each one of them is formed by a source node and its MPRs. Nodes in a local CDS only broadcast packets that come from the source node. This property is referred to as the *source dependent* broadcast, which requires a receiving node to check the sender of each packet, and thus increases the complexity of an algorithm. The MPR-CDS algorithm enhances the MPR technique by applying some strategies to the local CDSs and generating a global CDS in the network. Nodes in a global CDS rebroadcast every packet that is received once regardless its sender. This strategy is referred to as the *source independent* broadcast property. The algorithm requires the same node information as the original MPR heuristic, and hence, no extra message cost is introduced. Furthermore, since it is source independent, less time is used for each node in the CDS to forward a packet.

To generate a global CDS, the MPR-CDS algorithm applies two rules to the original MPR heuristic. A node x will be in the CDS if it meets either of the following rules:

Rule 1: x has the smallest node ID among its one-hop neighbors,

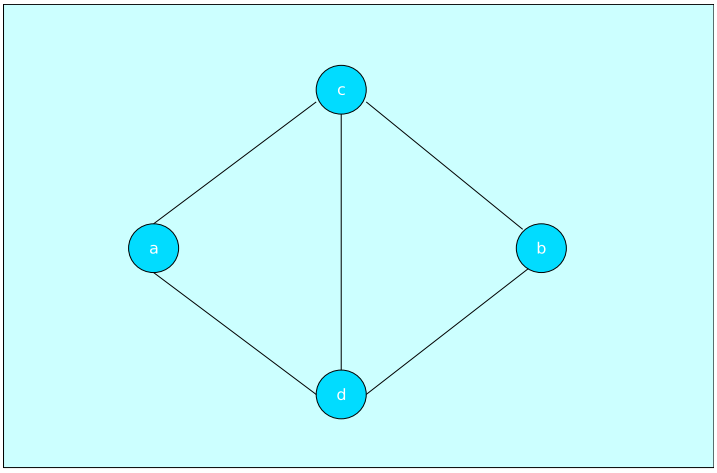
Rule 2: x has been selected as an MPR and the selector has the smallest node ID among its one-hop neighbors.

Specifically, the first rule is applied to all nodes in the network while the second one is used only by nodes inside MPR sets. In the MPR-CDS algorithm, the original

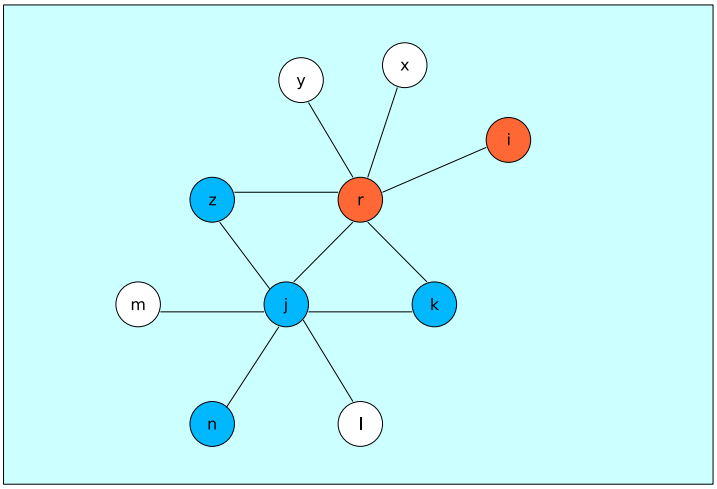
MPR heuristic is conducted first to generate MPR sets. Then the MPR selectors inform their one-hop neighbors about the MPRs they chose. Upon receiving this message, nodes that have been selected as MPRs apply the second rule to decide whether or not they are the dominating nodes (nodes inside the CDS). Furthermore, all nodes in the network also apply the first rule to evaluate themselves, and finally a CDS can be formed by combining all the dominating nodes in the network. It can be seen that the original MPR heuristic is a special case of the MPR-CDS where the only node elected by the first rule in a local CDS is the source node.

The advantage of the MPR-CDS algorithm is that it does not need any distributed knowledge of the global topology to generate a CDS in a network. This makes the algorithm very attractive for MANETs since it needs only local updates at each detected topology change. However, applying the two rules may increase the computation complexity of the algorithm since extra time is used for each node to evaluate itself.

In [39], Wu extends the MPR-CDS algorithm to construct a smaller CDS without additional cost. In the extended algorithm namely the *Enhanced MPR* (EMPR), two drawbacks of the MPR-CDS algorithm are pointed out. First, Rule 1 is unnecessary in many occasions, i.e. nodes selected based on Rule 1 are not essential for a CDS. Second, the original MPR heuristic does not take advantage of Rule 2. The first drawback can be explained by observing the diagram in Fig. 2.8(a) where nodes a and b are selected in the CDS based on Rule 1. However, It can be seen that node c alone is sufficient to cover all nodes in the network, and hence, Rule 1 is not suitable in this occasion. The second drawback puts forward that the MPR-CDS algorithm does not take the advantage of Rule 2 to achieve fault tolerance. Since only the MPRs whose selectors have the smallest node ID can be chosen as nodes in the CDS, and other selectors with larger node IDs will have no effect on the CDS



(a) An unnecessary occasion of Rule 1.



(b) The use of free neighbors.

Figure 2.8: Two drawbacks of the MPR-CDS algorithms.

calculation. Therefore, a node that does not have the smallest ID among its one-hop neighbors can choose its one-hop neighbors as MPRs without any extra cost. This strategy enhances the capability of fault tolerance in a MANET. As shown in Fig. 2.8(b), only node r is selected as an MPR by node i that has the smallest node ID among r's one-hop neighbors. If node i is switched off or leaves the network, based on the MPR-CDS algorithm, r will eliminate itself from the CDS, which may break the network connection. However, if node j also selects r as an MPR, j will

not affect r's decision to be in the CDS, and r will still be in the CDS without node i, and thus stabilizing the network. Such nodes like r are called *free neighbors* of j. The operation of the EMPR algorithm extends the MPR-CDS in two phases shown as follows:

Enhanced Rule 1: The node has the smallest ID among all its one-hop neighbors and it has two unconnected neighbors.

Enhanced original MPR heuristic: Initially, add all free neighbors of source node S to the MPR set and eliminate two-hop nodes that are covered by these free neighbors. Then apply the original MPR heuristic to the residual one-hop neighbors to cover all remaining two-hop nodes. Use the node ID to break a tie when two nodes cover the same number of uncovered two-hop nodes.

In essence, the Enhanced original MPR heuristic has already included the function of Rule 2, and thus the forwarding nodes selected by the heuristic are indeed in the CDS. Combining the both enhancements, the EMPR algorithm can generate a smaller CDS in a given network than the MPR-CDS algorithm.

Although the gains of two extensions are not explicitly given in [39], It is believed that both Enhanced Rule 1 and Enhanced original MPR heuristic contribute to the reduction of the size of the forwarding node set. For the Enhanced Rule 1, it adds more constrains to the original Rule 1, and thus reduces the chance of generating a forwarding node. For the Enhanced original MPR heuristic, the introduction of free neighbors can also reduces the number of forwarding nodes. This is due to the fact that free neighbors may have already covered a large number of two-hop nodes, and hence fewer number of forwarding nodes are needed to cover the residual two-hop nodes. An extreme case is that all two-hop nodes are covered by free neighbors, and therefore, no calculation is needed to generate any forwarding

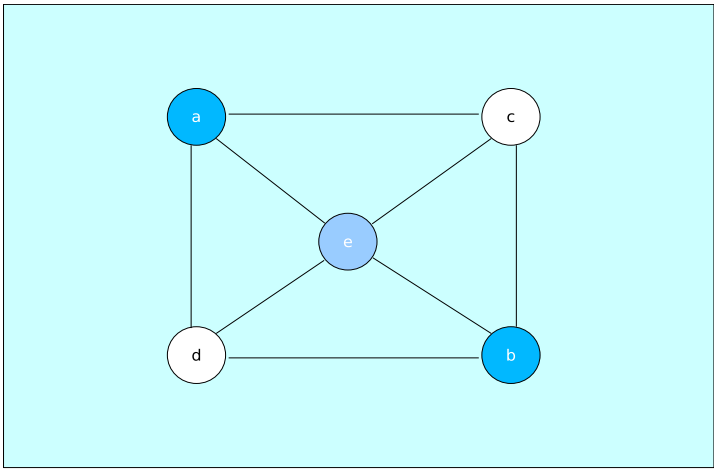


Figure 2.9: Extended Rule 1 in the DEMPR algorithm.

nodes. The gains of two extensions are also confirmed by simulation results presented in the paper, where the EMPR outperforms the MPR-CDS by producing less number of forwarding nodes in the network. In the paper, the author also points out that instead of the node ID, other criteria such as the node remaining power can be used to determine the forwarding nodes, so the resultant CDS can have some special properties like power awareness and mobility awareness.

In [40], Chen and Shen proposed a *Degree-Based Enhanced MPR* (DEMPR) algorithm that can further reduce the size of a CDS. They observed that the node degree value (the number of one-hop neighbors of a node) is more related to the size of a CDS than the node ID, and thus it should be given a higher priority when computing a forwarding node, and the node ID can be used whenever a tie happens. In their article, two improvements are put forward to enhance the EMPR algorithm:

Extended Rule 1: A node is in the CDS if it has the largest node degree among all its one-hop neighbors and it has two unconnected neighbors.

Extended Rule 2: A node is in the CDS if it has been selected as an MPR and its selector has the largest node degree among its one-hop neighbors.

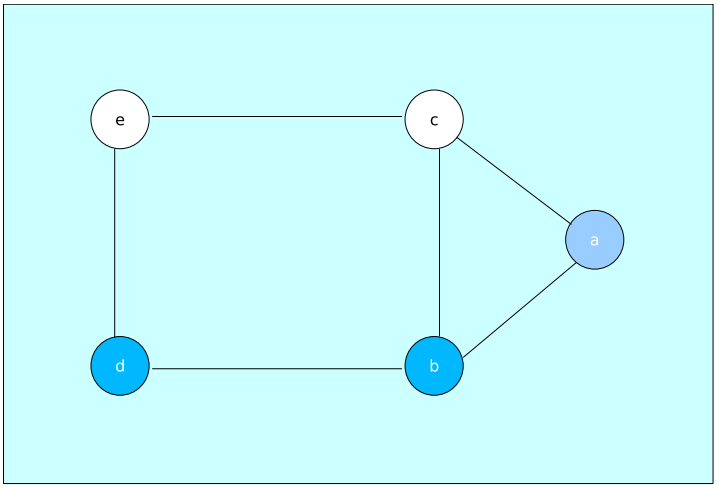


Figure 2.10: A pair of MPRs in the EEMPR algorithm.

Based on these two rules, the notion of free neighbors are also changed correspondingly. The free neighbors of a source node S are defined as its one-hop neighbors that have at least one neighbor node which has larger node degree than S.

Among the two extended rules, it is believed that the first one does most of the contribution on the reduction of the size of a CDS. This is because Rule 1 tends to choose a node as an MPR if it covers the largest number of nodes among its one-hop neighbors, so all its one-hop neighbors will not be elected as forwarding nodes by Rule 1, and therefore, fewer nodes are left in the network and fewer forwarding nodes are generated consequently. The result can be seen in Fig. 2.9. In the example network, node a and b are chosen as forwarding nodes based on rule 1 if node ID has a higher priority. However, only node e is selected when node degree is considered first.

In [41], Wu and Lou further extended the EMPR algorithm and proposed a new algorithm called *Extended Enhanced MPR* (EEMPR). The EEMPR algorithm uses three-hop neighbor information to cover all two-hop neighbors of each source node. The additional node information it requires is the one-hop neighborhood in-

formation of all two-hop neighbors of the source node. The basic idea behind this algorithm is that it tries to cover more two-hop neighbors with a pair of MPRs, which are two serially connected MPRs in one-hop neighborhood and two-hop neighborhood of the source node respectively. These two MPRs are either directly selected or indirectly selected by the source node. Fig. 2.10 can be used to help explain the operation of the EEMPR algorithm. In the original MPR heuristic, both node b and c are selected by the source node a since they both solely cover a two-hop node. However, it is worth noting that node d can also cover node e if d is chosen as an MPR of b. Therefore, all two-hop nodes of a can be covered eventually if node b and d are selected as a pair of MPRs. In such a case, node b is directly selected by node a while node d is indirectly selected by node a. Similar to the EMPR algorithm, free neighbors are also used in the EEMPR as they contribute additional coverage without introducing any cost. However, the definition of the free neighbor has been changed:

- **One-hop free neighbor:** A node y is a one-hop free neighbor of a source node S if y is in the one-hop neighborhood of S, and the node ID of S is not the smallest among the one-hop neighbors of y.
- **Two-hop free neighbor:** A node y is a two-hop free neighbor of a source node S if y is in the two-hop neighborhood of S, and the node ID of S is not the smallest among the one-hop neighbors of y.

As can be seen, two-hop free neighbors are also considered in the EEMPR algorithms, which help further reduce the number of MPRs generated in the two-hop neighborhood of a source node.

The EEMPR algorithm applies the Rule 1 used in the EMPR algorithm while an Enhanced Rule 2 is used to evaluate each pair of MPRs. The Enhanced Rule 2

operates as follows:

Enhanced Rule 2: A node x is in the CDS if:

1. It has been selected directly as an MPR and its selector has the smallest node ID among x's one-hop neighbors.
2. It has been selected indirectly as an MPR and its selector has the smallest node ID among x's one-hop neighbors.

The process of the MPR calculation is also extended in the EEMPR algorithm. Initially, all one-hop and two-hop free neighbors are added to the MPR set, and the two-hop nodes they covered are removed from the two-hop neighborhood of the source node. Then, among the one-hop and two-hop nodes that remain connected, pick up a pair of connected nodes as MPRs if they cover the most number of uncovered two-hop neighbors of the source node. Use the node ID to break a tie if multiple pairs exist.

The gain for using additional node information is that fewer MPRs are generated by each source node. The improvement can be explained by referring to the original MPR heuristic, where MPRs chosen by the initial phase tend to dominate the MPR set. The strategy used in the EEMPR algorithm avoids this problem, because a pair of MPRs may have a chance to cover those solely covered two-hop nodes, and thus fewer nodes are selected as MPRs to particularly cover those two-hop nodes. However, the drawback of the algorithm is that each source node has to reconsider all pairs of MPRs to cover its two-hop neighbors without taking the advantage of the forwarding nodes that have already been selected. This may prolong the calculation process and increase the number of the forwarding nodes in the network. Since for each source node, some forwarding nodes in its two-hop

neighborhood may have already been chosen into the CDS, these forwarding nodes can be excluded when the one-hop neighbors of the source begin to calculate their forwarding nodes. That is to say, one should treat these forwarding nodes as the free neighbors, even though they do not meet the free neighbor criteria. To further improve the EEMPR algorithm, the tie-break procedure also needs to be enhanced in order to optimize MPRs selected in the two-hop neighborhood of the source. Comparing to the node ID, It is believed the node degree of the second hop MPR is more suitable as the tie breaking criterion to reduce the size of the CDS. When multiple pairs of MPRs are available, the pair that has higher node degree of the second hop MPR should be given higher priority. This strategy is prone to choose a second hop MPR that has a larger coverage, and thus when the MPR is included as a free neighbors of a node, more two-hop nodes can potently be covered and a source generates fewer number of forwarding nodes.

2.6.2 Cluster-CDS Based Algorithms

In [42], Alzoubi et al. proposed a new algorithm called the *Geometric CDS* (GCDS) to approximate the minimum connected dominating set (MCDS) in a given network. In general, the algorithm operates in two steps:

1. **Clustering:** It first clusters the network by electing some *cluster-heads* (also known as *dominators*). The elected cluster-heads are able to reach all nodes in the network, but they are not connected yet. Nodes that can be reached by the dominators are called dominatees.

2. **Finding connectors:** It then generates *connectors* from the dominatees to connect to the dominators.

It is easy to see that the dominators generated in the first step form a *maximum independent set* MIS in the network since they are disjointed and sufficient to cover all nodes in the network. The definition of an MIS is described as follows:

Definition 2.2. *An* independent set *(IS) is a subset of nodes in a network, where any arbitrarily chosen pair of nodes in the IS is disconnected. A* maximum independent set *(MIS) is the largest IS in the network, where no other nodes can be added into this IS.*

The connectors generated in the second step connect the graph induced by the nodes in the MIS, and finally, a CDS can be form by combining the resultant dominators and connectors in the network.

In the GCDS algorithm, a node can be in one of the four states: dominator, dominatee, white_node, and connector. Initially, all nodes in the network are in the white_node state, and then they enter either dominator or dominatee state after the clustering process. The connector state can be only entered from the dominatee state. The rule of electing dominators can be described as follows:

- A white_node claims itself to be a dominator if it has the smallest node ID among all of its one-hop white_node neighbors.

The dominators generated by the above rule are two hops or three hops away from each other. After a white_node u changes its state to the dominator, it sends out a signaling message called $IamDominator(u)$ to its one-hop white_node neighbors to inform its new state. A white_node v that receives this message changes its state to the dominatee, and it stores the node ID of this dominator. Then v sends out an $IamDominatee(u, v)$ message to its neighbors to inform them that it can reach dominator u.

It takes two steps for a dominatee v to determine whether it is a connector for a pair of two-hop away dominators.

1. Dominatee v sends out a $TryConnector(u, v, w, 0)$ message to its one-hop neighbors for each dominator pair u and w located in v's one-hop neighborhood. The message indicates that v is capable to connect to dominators u and w, where the integer 0 represents that u and w are two hops away from each other.

2. Dominatee v announces itself as a connector to link to dominators u and w if it has a smaller node ID than all its neighbors that sent the message $TryConnector(u, *, w, 0)$. Then v broadcasts an $IamConnector(u, v, w, 0)$ to its neighbors to inform its new state.

It requires four steps for a dominatee v to become a connector to link to a pair of three hops away dominators.

1. For each pair of dominators u and w, which are one hop and two hops away from dominatee v respectively, v broadcasts a $TryConnector(u, v, w, 1)$ message to its one-hop neighbors, where the integer 1 indicates that u and w is three hops apart and v is the first connector on the path to connect to u and w.

2. Dominatee v changes its state to the connector if it has a smaller node ID than all its neighbors that sent the message $TryConnector(u, *, w, 1)$. It then broadcasts an $IamConnector(u, v, w, 1)$ message to inform its new state.

3. Connector v selects a dominatee x from its one-hop neighborhood to connect to two hops away dominator w, if x has the smallest node ID among all v's one-hop dominatees that can reach w. Then v sends a signaling message to x asking it to be a connector to link to dominator w.

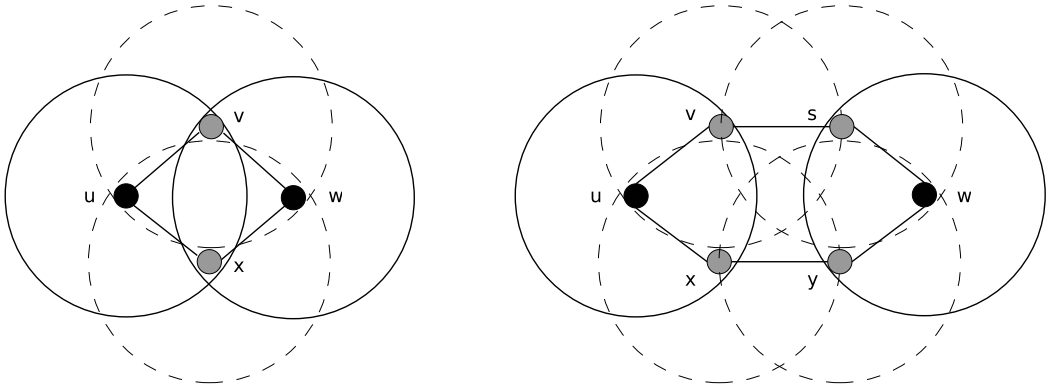


Figure 2.11: Examples of redundant connectors generated in the GCDS algorithm.

4. After receiving such a message from v, dominatee x changes its state to the connector and broadcasts an $IamConnector(u, x, w, 2)$ message.

After the above procedures, the algorithm guarantees that each pair of two-hop and three-hop away dominators can be connected, and therefore, a CDS is formed in the network.

The drawback of the GCDS algorithm is that it may generate redundant connectors for a pair of dominators. This is due to the fact that each dominatee determines whether it is a connector based on the $TryConnector$ messages sent by its dominatee neighbors, so two dominatees may both decide to connect to the same dominator if they are not within the transmission range of each other. Fig. 2.11 demonstrates this problem. In Fig. 2.11(a), dominators u and w are two hops apart, and dominatees v and x are not located in the one-hop neighborhood of each other. Therefore, v and x both announce to be connectors to link to the dominator pair u and w. Similar situation also happens in Fig. 2.11(b), where connector pairs v, s and x, y are generate to connect to three-hop away dominators u and w. It is proven in [42] that for each pair of two-hop away dominators, there are at most two nodes claiming to be connectors for them, and for each pair of three-hop away dominators,

at most ten connectors are generated to link to them.

In [43], Gao et al. presented an efficient approximation algorithm called the *efficient CDS* (ECDS) algorithm for constructing a minimum connected dominating set (MCDS) in MANETs. The algorithm also forms an MIS first in the network, and then it connects to the nodes in the MIS to construct a CDS. However, the algorithm ensures that there is a unique path generated between a pair of dominators whose distance is within three hops.

Similar to the GCDS algorithm, nodes in the ECDS algorithm can also be in one of the four states: *dominator*, *dominatee*, *connector*, and *candidate*, where all nodes initially are in the candidate state. The algorithm follows the same steps as the GCDS algorithm to construct an MIS in the network, but unlike the GCDS algorithm where dominatees evaluate themselves to become connectors, connectors in the ECDS algorithm are selected by the dominators. A dominator u selects connectors to connect to a two-hop or three-hop away dominator w based on the following steps:

1. Dominator u broadcasts a $REQUEST\_DOMI$ message to find all other dominators within three-hop distance from it. The message can only be relayed by dominatees at most two times before it reaches a dominator. This can be done by setting the time-to-live (TTL) value in the message header to two, and the message will be dropped by a node if the TTL value reaches zero.

2. A dominatee that receives the $REQUEST\_DOMI$ message from u first checks the TTL value in the message. It drops the message if the TTL value is zero, otherwise, it appends its node ID in the message and decreases the TTL value by one. Then it broadcasts the message to its one-hop neighbors.

3. After a dominator w receives the $REQUEST\_DOMI$ message for the first time, it checks the dominatee IDs in the message, so it knows the path that the message went through. Then it generates a $REPLY\_DOMI$ message including a path that this message should visit and sends it. The path is a reverse order of the one in the $REQUEST\_DOMI$ message. If another $REQUEST\_DOMI$ message that contains a different path to the same dominator u is received by w, it compares two paths and stores the shorter one. Then a new $REPLY\_DOMI$ is created and sent.

4. When the dominatee that is included in the path receives the $REPLY\_DOMI$ message, it changes its state to connector and sends the message to the next node according to the path in the message.

Above steps guarantee the connectivity of each pair of dominators that are within a distance of three hops, and they also ensure that for each pair of dominators, there is only one path connecting them.

In [29], Alzoubi et al. proposed an *message-optimal CDS* algorithm (MOCDS) that intends to minimize the communication complexity of constructing a CDS in MANETs. Similar to the previous algorithms, the MOCDS algorithm also operates in two phases: constructing an MIS and generating connectors. It first applies the same method as used in the GCDS algorithm to cluster the network, and then it uses a different approach to find connectors and ensures a unique path between dominator pairs.

In the MOCDS algorithm, each dominatee maintains two lists: $list_1$ and $list_2$, where $list_1$ stores node IDs of one-hop nearby dominators, and $list_2$ stores IDs of two-hop away dominators and IDs of the dominatees that can reach them. Each dominator also maintains two lists: $LIST_2$ and $LIST_3$, which contain IDs of domi-

nators that are two (three) hops away from it and IDs of dominatees that can connect to these dominators. To find connectors, each dominatee node broadcasts two signaling messages containing $list_1$ and $list_2$ respectively to its one-hop neighbors. Upon receiving the $list_1$ message, a dominator u checks the dominator IDs included in the message and it inserts a dominator ID and the message sender ID into its $LIST_2$ list if this dominator does not appear in the $LIST_2$ and $LIST_3$ lists and u has a smaller node ID than this dominator. Similarly, u selects a dominator from the $list_2$ message and inserts the dominator's ID and the message sender ID into the $LIST_3$ list, if this dominator does not appear in u's $LIST_2$ and $LIST_3$ list and u has a smaller node ID than it.

To find connectors, a dominator u broadcasts a message that contains all entries in its $LIST_2$ and $LIST_3$ lists to its one-hop neighbors. A node v (other than a dominator) that receives this message checks whether its ID appears in any of the entries in the message and if so it proceeds as follows:

1. It changes its state to the connector and creates a $Rlist$ list that contains two fields: a pair of IDs of two dominators that can be connected through v, and the ID of another connector if the two dominators are three hops apart from each other.

2. For each entry in the message that contains its ID, v inserts the message sender ID and the ID of the dominator (target dominator) that is contained in the entry into the first field of the $Rlist$.

3. When the target dominator is two hops away from it, v checks its $list_2$ list and inserts node w's ID into the second field of the $Rlist$, if w can reach the target dominator. when the target dominator is one hop away from it, v sets the second field of the $Rlils$ to $null$.

Since only a dominator with the smaller node ID broadcasts signaling messages to select connectors, at most one connector is selected for a pair of dominators that are two or three hops apart from each other.

2.6.3 Summary of CDS Based Algorithms

In this section, the existing CDS based algorithms are surveyed and classified into two groups based on the methods they used to form a CDS in the network. The summary of the classification can be referred to Table 2.3. In order to compare the performance of the CDS based algorithms, the costs of the algorithms are evaluated here. A comparison of the costs of different algorithms is shown in Table 2.4.

As the Table shows all the CDS based algorithms are not source dependent, since they generate a CDS in the network, where nodes in the CDS retransmit broadcast packets regardless the senders. Among the algorithms, the EEMPR, ECDS and MOCDS algorithms require three-hop away node information. In the EEMPR algorithm, each node in the network includes IDs of its one-hop neighbors and IDs of their one-hop neighbors in the signaling messages, and therefore, a node in the network has knowledge of other nodes that are within three hops distance from it. In the ECDS algorithm, each dominator collects information of other dominators that are at most three hops away from it by sending the $REQUEST\_DOMI$ messages, whilst dominators in the MOCDS algorithm finds three-hop away dominators via the $list_2$ messages sent by dominatees. The extra node information gives a node more details of the network topology and consequently facilitates the CDS calculation. However, collecting and storing such information require more time and memory space, which can result in inaccuracy of the information in a mobile environment since nodes move frequently and the information collected from a node

Table 2.4: The Cost comparison of the CDS based algorithms.

Algorithms	**Information range**	**Source dependent**	**Time complexity**	**Message complexity**	**Message size**	**Approximation ratio**
MPR-CDS [38]	2 hops	No	$O(3\Delta M + 3\Delta)$	$O(n)$	$O(\Delta)$	Δ
EMPR [39]	2 hops	No	$> O(3\Delta M + \Delta^2)$	$O(n)$	$O(\Delta)$	Δ
DEMPR [40]	2 hops	No	$> O(3\Delta M + \Delta^2)$	$O(n)$	$O(\Delta)$	Δ
EEMPR [41]	3 hops	No	$> O(3\Delta M + \Phi\Delta + \Delta^2)$	$O(n)$	$O(\Delta^2)$	Δ
GCDS [42]	2 hops	No	$O(n + \Delta)$	$O(n)$	$O(1)$	*constant*
ECDS [43]	3 hops	No	$O(n)$	$O(n)$	$O(1)$	*constant*
MOCDS [29]	3 hops	No	$O(n + \Delta)$	$O(n)$	$O(1)$	*constant*

may be "stale" already. Therefore, balancing the hop information range is a non-trivial issue in designing broadcast protocols in MANETs.

The time complexity of the MPR-CDS algorithm is determined by the time used for each node to complete the CDS calculation. Compared with the original MPR algorithm (refer to Section 2.2), the extra computational cost of the MPR-CDS is the two rules that determine the forwarding state of a node. It is estimated that both rules need $O(\Delta)$ time to finish, so the overall time complexity of the MPR-CDS algorithm is $O(3\Delta M + 3\Delta)$. In the EMPR algorithm, two extensions are introduced to the Rule 1 and the heuristic of the MPR-CDS algorithm, both of them add extra computational cost. Since the detail of the algorithm is not presented in the article, it is difficult to deduce the explicit time complicity of these extra costs. However, it is estimated that the process to add all free neighbors to an MPR set in the extension may take $O(\Delta^2)$ time to run in the worst case, and it might be the dominant part of the extra costs. Therefore, the time complexity of the EMPR algorithm should be larger than $O(3\Delta M + \Delta^2)$. Similar analysis can also be applied to the DEMPR algorithm, and the algorithm discriminates the EMPR algorithm only in the criterion of two rules, it has the same time complexity as the EMPR algorithm. In the EEMPR algorithm, a more complex process is used to

evaluate nodes in order to form a smaller size CDS. The process that adds both one-hop and two-hop free neighbors in an MPR set may take $O(\Delta^2 + |N_2|\Delta)$ time to complete. The MPR set selection process in the EEMPR algorithm is also extended, which chooses a pair of MPRs at each round. However, how to choose such a pair of MPRs is not specified in the article, it can be estimated that the process may consume more time than the one used the EMPR algorithm, and hence, the overall time complexity of the EEMPR is larger than $O(3\Delta M + \Phi\Delta + \Delta^2)$. For the cluster-CDS based algorithms, it has already been proven in [29] that the time used to construct an MIS in the network is no larger than $O(n)$. In the GCDS algorithm, each dominator requires $O(\Delta)$ time to select dominatee to link to other two-hop away dominators, and each dominatee also uses $O(\Delta)$ time to decide whether it is a connector. Hence, the total time complexity of the GCDS algorithm can be $O(n + \Delta)$. In the ECDS algorithm, since each dominator and dominatee node use a constant time to process the $REQUEST\_DOMI$ and $REPLY\_DOMI$ messages, the overall time complexity of the algorithm is $O(n)$. For the MOCDS algorithm, $O(\Delta)$ time is needed for both a dominatee and a dominator to build their lists, so the overall the time complexity of the algorithm is $O(n + \Delta)$.

In the direct-CDS based algorithms, since all nodes in the network send out constant number of signaling messages during the CDS construction, the message complexity of them are $O(n)$. For the cluster-CDS based algorithms, each node in these algorithms sends exactly one signaling message to its neighbors indicating its state, and after forming an MIS in the network, dominators and dominatees in these algorithms also send out a constant number of signaling messages to select connectors, and therefore, the message complexities of the cluster-CDS based algorithms are also $O(n)$. However, the number of signaling messages sent by each algorithm varies in practical, which will be shown in the simulation study presented in this

thesis.

As can be seen from Table 2.4, most of the direct-CDS based algorithms have the same signaling message size. This is due to the fact that nodes in these algorithms need to include all IDs of their one-hop neighbors in signaling messages, which can be as large as $O(\Delta)$. But for the EEMPR algorithm, since each node in the network needs to know all nodes within a three-hop range from it, the signaling message size of the algorithm can be $O(\Delta^2)$. In the GCDS algorithm, the $IamDominatee$, $IamDominator$, and $IamConnector$ messages contain fixed size fields. The $TryConnector$ messages sent by dominatees store pairs of dominators that are within three hops away from each other. Since it has been proven in [42] that the number of dominators within k hops from a node is bounded by a constant, the size of the $TryConnector$ message is also bounded. Therefore, the GCDS algorithm has a constant bounded message size. In the ECDS algorithm, both $REQUEST\_DOMI$ and $REPLY\_DOMI$ messages only contain fixed length fields, so the message size of the algorithm is bounded. For the MOCDS algorithm, the signaling messages sent by dominatees contain entries in the $list_1$ and $list_2$ lists, and the signaling messages sent by dominators contain entries in the $LIST_2$ and $LIST_3$ lists. Based on the similar proof in [42], sizes of the above lists are all bounded by constants, so the signaling message size of the MOCDS is also bounded.

Since all the CDS based algorithms are approximated algorithms, their performances can be evaluated using the approximation ratio (refer to Section 2.4). For the MPR-CDS, EMPR and DEMPR algorithms, because they apply the original MPR heuristic for calculating forwarding nodes, their approximation ratios should be the same as the original MPR algorithm. For the EEMPR algorithm, it has been proven in the article that for each node in the network, the number of forwarding

node it selected is bounded by a constant, which indicates that the algorithm has a local approximation ratio. However, it also points out that considering the entire network, the approximation of the algorithm is still Δ since the worst case can be found when nodes are placed in a line, and the node IDs increase monotonously along the line. For the cluster-CDS based algorithms, it has been proven in [44] that the size of the MIS created by using the method described in the algorithms is bounded by a constant. Furthermore, for each pair of dominators that are within three-hop distance from each other, only a constant number of connectors are generated by the algorithms to link to them, and therefore, the total number of nodes in the CDS is also bounded, so the algorithms all have constant approximation ratios. However, dominators in the cluster-CDS based algorithms tend to connect to all the three-hop away dominators, which might result in unnecessary connectors in some occasions, since some three-hop away dominators may be already connected by other two-hop away dominators. Therefore, the algorithms can further improved by considering whether a three-hop away dominator is needed to be connected.

2.7 Summary

In this chapter, a thorough survey is presented to discuss some leading MPR and CDS based broadcast algorithms for MANETs. The algorithms are categorized into groups and detailed information on the operation process of each algorithm is reviewed. Comprehensive performance comparisons between different algorithms are conducted. Also, problems and possible solutions are provided in order to help readers further improve the algorithms.

From the survey it can be seen that, the original MPR heuristic is simple in both computation and communication load, and is easy to implement. However, the

original MPR method is source dependent, which may increase the complexity of the algorithm since nodes in the network need to check the sender of each packet. A good way to avoid the source dependency is to construct a CDS in the network to relay broadcast packets. As described in the survey, a CDS can be formed by either generating forwarding nodes directly or clustering the network first then connecting the cluster-heads. Between these two approaches, the second one is thought to be more suitable for MANETs, since it can create a hierarchy over the network. Creation of a network hierarchy can be extremely useful as people can change the cluster-head election criteria to choose cluster-heads with special properties, such as more remaining power, higher computing capability, and larger memory capacity.

Enlightened by the MPR and CDS approaches, three new efficient broadcast algorithms called Gateway Multipoint Relay (GMPR), Enhanced Gateway Multipoint Relay (EGMPR) and Low-Cost Flooding (LCF) algorithms have been invented in this research project. The algorithms are simple and their computational and communication loads imposed on the nodes are low. Especially the LCF algorithm, has linear time and message complexity, and its signaling message size and the approximation ratio are bounded by constants. Details of the proposed algorithms are presented in the following chapters.

Chapter 3

Proposed Efficient Broadcast Algorithms

3.1 Introduction

In this research project, based on the MPR and CDS approaches, three new efficient broadcast algorithms for MANETs are proposed. The algorithms aim to minimize the redundant transmissions in the network by limiting the number of nodes that forward the broadcast packets. To achieve this, the algorithms generate a CDS of the network, where only the nodes in the CDS retransmit broadcast packets. This chapter discusses the proposed algorithms in detail.

3.2 Gateway MPR Algorithm

The *Gateway Multipoint Relay* (GMPR) algorithm [45] has been proposed for constructing a CDS in a network. The process of finding a CDS in the algorithm can

be described in two phases. In the first phase, an MIS is constructed in the network where nodes in the MIS are called *dominators*, and nodes covered by the dominators are referred to as *dominatees*. The dominators form a dominating set (DS) (refer to Definition 2.1) and they operate as gateways in the network. In the second phase, each gateway calculates an MPR set to cover all its two-hop neighbors by using a heuristic similar to the one that was proposed in the original MPR algorithm. However, not all the MPR nodes need to forward packets in the network. An MPR node further determines whether it needs to forward broadcast packets based on the node degree value of its MPR selectors. An MPR is a *forwarding* MPR only if it is selected by a dominator whose node degree is the largest among all this MPR's one-hop neighbors. The forwarding MPRs are then put in the DS and they perform as connectors of the gateways. It is proven in the next chapter that the gateways and connectors generated in the GMPR algorithm can form a CDS in the network. After the CDS construction, a self-pruning procedure that aims to remove redundancy in the network is applied to each gateway to further evaluate whether a gateway is necessary in the CDS.

In the GMPR algorithm, a node can be in one of four states: dominator, dominatee, connector and white_node. Initially, all nodes are in the white_node state, and they subsequently change to either the dominator or dominatee state. Only nodes in the dominatee state can change to the connector state. Fig. 3.1 shows the state transition process of a node, where arrows indicate the possible transition directions.

Similar to the original MPR algorithm, the GMPR algorithm also requires each node in the network to periodically exchange hello messages between its one-hop neighbors. The hello message contains neighbor node information that is necessary for the operation of an algorithm. For nodes in different states, the contents of the

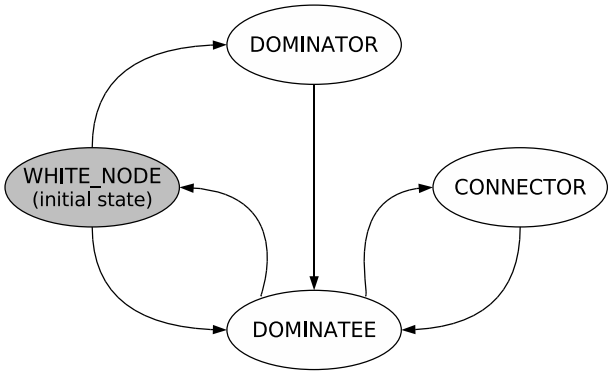


Figure 3.1: The state transition diagram of a node in the GMPR algorithm.

hello messages vary. Fig. 3.2 shows the format of the hello messages of a node in different states. Fields in the messages are similar to the ones in the original MPR algorithm. The only extra information required by the GMPR algorithm is the state of a node, since each node in the network needs to indicate its state to the neighbors. Since there are only four states in the GMPR algorithm, they can be easily represented using only two bits in a hello message, and therefore, this extra cost is really marginal and can be ignored. Furthermore, it is noted that in the GMPR algorithm, only the hello message sent by a dominator carries the field of MPRs, while in the original MPR algorithm, all hello messages carry this field. This property can largely reduce the overall signaling message size of the GMPR algorithm, and thus shorten the time to transmit and check the signaling messages for each node in the network. For the operation of the algorithm, it is assumed that each node has a unique ID and it knows the IDs of all its one-hop neighbors, which can be achieved by letting each node broadcast its ID to its one-hop neighbors initially. Also, each node in the network has a omni-directional antenna with fixed transmission range, and a node can communicate with another node if the distance between them is shorter than the transmission range.

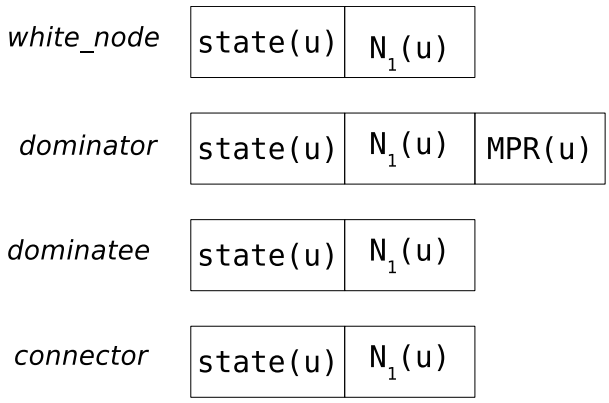


Figure 3.2: The hello message format of the GMPR algorithm.

3.2.1 Generating Gateways in Networks

In the first phase of the GMPR algorithm, the network is clustered and the clusterheads or dominators are referred to as the gateways of the network. The gateways form an MIS in the network (is proven in the next chapter), where each node in the MIS is at most three hops apart from others. Different criteria can be used to elect gateways, but in this algorithm, the node degree value is used as the metric since it can reduce the number of resultant gateways in the network thus limiting the total number nodes in the CDS. This is due to the fact that a gateway with larger node degree may have larger coverage of nodes, and these covered nodes will not be elected as gateways, fewer nodes are left for the gateway election, and thus fewer gateways will be generated in the network. The process of generating gateways can be described as follows:

- Initially, all nodes in the network are in the white_node state, and a white_node u announces itself as a dominator if its node degree $\deg(u)$ is the largest among its one-hop white_node neighbors (neighbors in the white_node state) or it has no white_node neighbors and dominators in its one-hop neighborhood.

- A white_node u changes to the dominatee state if it receives a hello message from a dominator v, and v has a larger node degree $\deg(u)$ than u.
- A dominator u becomes a dominatee if it receives a hello message from another dominator v, and v has a larger node degree $\deg(v)$ than u.
- A dominatee u changes back to the white_node state if it loses all dominators around its one-hop neighborhood.

If a tie occurs in the above steps, the node with the smaller node ID will be given the priority. Nodes in the dominator state formulate an MIS in the network since no two adjacent nodes will be marked as dominators, and these dominators operate as gateways to relay packets throughout the network. It is possible to change the property of the generated gateways by using different metrics during the gateway election process. A possible example is to use a node's remaining power as the criterion for choosing gateways in order to extend the life span of the network.

3.2.2 Generating a CDS

In this phase, each gateway calculates an MPR set based on a greedy algorithm [26] similar to the original MPR heuristic discussed in Section 2.2. For a given node u that needs to calculate an MPR set, assume that v is a one-hop neighbor of u. Let $P_v = |\{w \in N_1(v) | v \in N_1(u) \text{ and } w \in N_2(u)\}|$ denote the *preference* of node v, which is the number of two-hop neighbors of u that node v can cover. The greedy algorithm operates as follows:

- Add nodes in $N_1(u)$ to $\mathrm{MPR}(u)$ if they are the only neighbors of some nodes in $N_2(u)$. Then update u's neighborhood information by removing MPRs and the two-hop neighbors covered the MPRs from $N_1(u)$ and $N_2(u)$ respectively.

- If there are still some uncovered two-hop neighbors, add a one-hop neighbor to $\mathrm{MPR}(u)$ if it covers the largest number of uncovered two-hop neighbors. If there is a tie, choose the node with larger P as the MPR. In case of another tie, select the node with a smaller node ID.

After the first phase, each dominatee has some dominators in its one-hop neighborhood. It is defined that the *largest dominator* of a dominatee u is the one that has the largest node degree value $\deg$, or in case of a tie, the one that has the largest node ID value among all the dominators in u's one-hop neighborhood. The node IDs of selected MPRs are included in the hello messages of gateways. Upon receiving these hello messages, a dominatee determines its state based on the following steps:

- A dominatee u changes to the connector state if it is selected as an MPR by a dominator v, and $\deg(v)$ is the largest among all u's one-hop away dominators.
- A connector u returns to the dominatee state if its largest dominator does not choose it as the MPR.

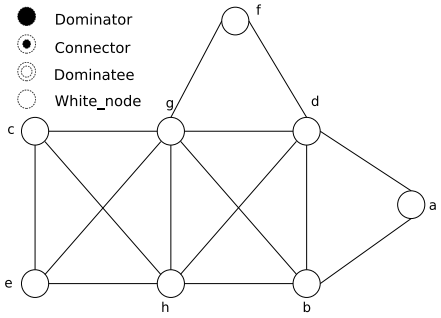
Only the nodes which are in the connector state retransmit broadcast packets, and they ensure the connectivity of the gateways. By combining all the gateways and connectors generated in the network, a CDS can be constructed.

3.2.3 Self-Pruning Procedure

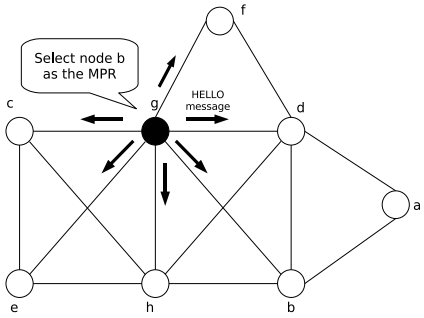
After generating connectors, each dominator applies a *self-pruning procedure* (SP) to determine whether it is redundant in the CDS, and if so it removes itself from the CDS. The self-pruning procedure can be described as follows:

- A dominator u eliminates itself from the CDS if it has a connector v that is in $N_1(u)$, and v can cover all the one-hop neighbors of u.

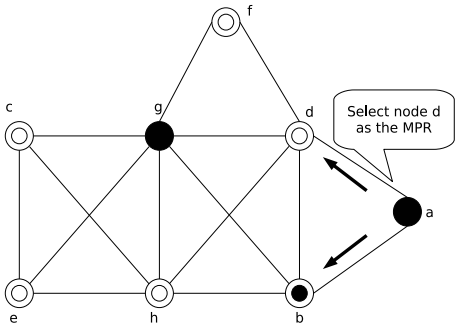
The dominator that is eliminated by the self-pruning procedure is referred to as a *silent-dominator*, which still has the dominator state and calculates the MPR set, but it will no longer retransmit packets. The self-pruning procedure can effectively reduce the number of gateways in the network, thus further limiting the size of the CDS generated by the algorithm.



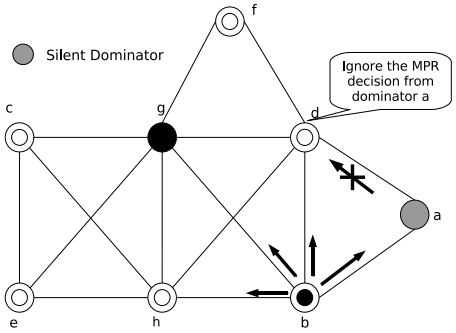
(a) Initially, all nodes are in the white_node state.



(b) g broadcasts a hello message to announce itself as a dominator, and it selects b as its MPR.



(c) a announces itself as a dominator and selects d as its MPR.



(d) d ignores a's decision, and a changes to a silent-dominator after it receives the hello message from b.

Figure 3.3: An example of constructing a CDS by using the GMPR algorithm.

Fig. 3.3 demonstrates an example of the processes of constructing a CDS using the GMPR algorithm. The letter near each node represents the node ID, the arrows represent hello messages and their sending directions, and the cross on an arrow

indicates that the message has been ignored. A CDS is constructed through the following steps:

1. Initially, all nodes in this network are in the white_node state as shown in Fig. 3.3(a). After knowing its neighborhood information, node g announces itself as a dominator since it has a larger node degree $deg(g)$ than all its white_node neighbors. Then g calculates an MPR set to cover its two-hop neighbors. Based on the greedy algorithm described in Section 3.2.2, node b is selected as an MPR because it covers g's only two-hop neighbor node a, and b has a smaller node ID than node d. After selecting all the MPRs, node g immediately broadcasts a hello message to inform its one-hop neighbors about its new state and the MPR nodes it has selected. The above processes are shown in Fig. 3.3(b).

2. Upon receiving g's hello message, g's one-hop neighbors change their states to the dominatees, and node b further changes to the connector state since dominator g, which currently is the only dominator around b, has chosen it as an MPR. Then all dominatees immediately broadcast hello messages to indicate their new states to their one-hop neighbors. After receiving the hello messages from dominatees b and d, node a declares itself as the dominator because it has no white_node neighbors around. Then a also calculates an MPR set to cover all its two-hop neighbors, which are nodes f, g and h in this case. Consequently, a selects node d as the MPR because it can cover more two-hop neighbors than node b. Then a sends out a hello message to all its one-hop neighbors indicating its new state and the MPRs it has chosen. Fig. 3.3(c) illustrates above processes.

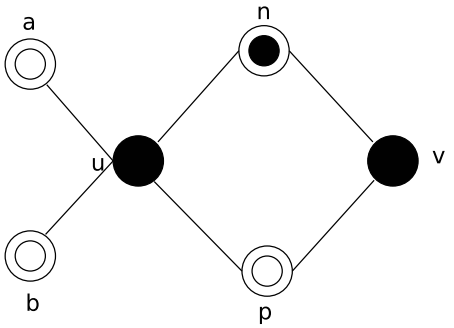
3. Upon receiving the hello message from a, nodes b and d have the knowledge

of this new dominator, and d further notices that it is chosen as an MPR by a. However, d ignores a's MPR decision due to the fact that a is not the largest dominator around d. Therefore, only node b operates as the connector to connect to dominators a and g. At this time, b sends out a hello message to its one-hop neighbors with its state set as the connector. After receiving this hello message, both dominators a and g run the self-pruning procedure to evaluate themselves, and a further removes itself from the CDS and becomes a *silent-dominator* since all its one-hop neighbors can be covered by connector b.

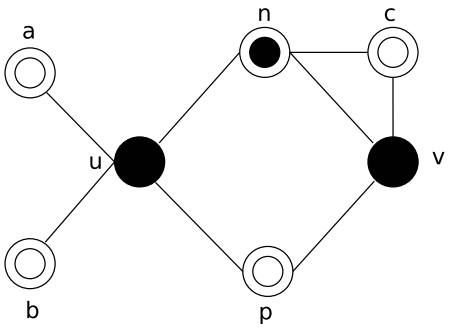
In general, the GMPR algorithm combines the MPR and clustering methods to form a CDS in a network. The algorithm limits the number of forwarding nodes in two ways. First, it applies a condition to the MPRs to determine whether they can be put into the CDS. Second, the self-pruning procedure applied to the dominators can further reduce the number of nodes in the CDS. Moreover, the algorithm also reduces the overall workload of computing MPRs because only the dominators in the network conduct the MPR calculation. Also, since IDs of the selected MPRs are only included in the hello messages of dominators, the overall signaling message size of the GMPR algorithm is also reduced.

3.3 Enhanced Gateway MPR Algorithm

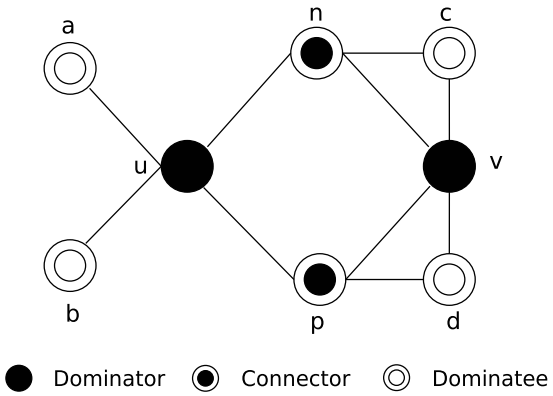
Based on the GMPR algorithm, an *Enhanced Gateway Multipoint Relay* (EGMPR) algorithm [46] is proposed. It aims to extend the GMPR algorithm to further reduce the size of a CDS generated in a given network while still keeping low computation and communication complexities and small overhead of the signaling message. The EGMPR algorithm achieves better performance by enhancing the self-pruning



(a) All dominatees of v can be covered by dominator u.



(b) All dominatees of v can be covered by dominator u and connector n.



(c) All dominatees of v can be covered by dominator u and connectors m and n.

Figure 3.4: Three example networks to show the drawbacks of the GMPR algorithm.

procedure and the hello message format of the GMPR algorithm. The detail of the enhancements is described in this section.

3.3.1 Drawbacks of GMPR Algorithm

It is observed that the self-pruning procedure used in the GMPR algorithm is insufficient on many occasions. In particular, when there is a sparse network topology, the chance that a single connector covers all one-hop neighbors of a dominator is rather low. To clearly illustrate this, three examples are shown in Fig. 3.4.

In Fig. 3.4(a), nodes u and v are dominators and node n is selected as a connector by u since it covers u's two-hop neighbor v, and it has a smaller ID than p (node IDs are ranked in alphabetical order). It can be seen that in this network, n cannot cover all the one-hop neighbors of dominator v, but dominator u alone can sufficiently cover them. In Fig. 3.4(b), a node c is added in the one-hop neighborhood of dominator v, and thus, u cannot solely cover all the one-hop neighbors of v, but it is still possible to cover them by using both u and n. The final example network in Fig. 3.4(c) shows that dominator u selects two connectors, n and p, to cover its two-hop neighbor nodes, and when combining u, n and p, all the one-hop dominatees of v can be covered. From these three examples, it can be seen that dominator v is redundant and can be removed from the CDS if dominator u and its connectors n and p are considered in the self-pruning procedure. Based on this concept, the GMPR algorithm can be enhanced to further reduce a CDS size in a network.

3.3.2 Extended Hello Message Format

In order to enhance the self-pruning procedure, it is necessary to extend the format of the hello messages sent by each dominatee and connector in the GMPR. A new variable called LARGEST_DOMINATOR, which contains the node ID of the largest dominator[1] of a dominatee (or a connector), is appended in the hello message. It is said that a connector u "belongs" to a dominator v, if v is the largest dominator of u. Since it has been proven in [42] that based on the clustering method used in the GMPR algorithm, at most 5 dominators can connect to a given dominatee. Therefore, only constant time is needed for each dominatee

[1]The largest dominator of a dominatee u is defined as a dominator whose node degree $\deg$ is the largest among all the dominators that are in the one-hop neighborhood of u.

to determine its largest dominator, and since the LARGEST_DOMINATOR value only adds one node ID in the hello message, the increase of the message overhead is marginal. After receiving the extended hello messages from dominatees and connectors, a dominator u knows the nodes in its one-hop neighborhood that belong to other two-hop away dominators, and consequently, it knows the two-hop away dominators that have larger node degree than itself.

The above processes can be illustrated by referring to Fig. 3.4(a). In this network, u's ID is assigned to the LARGEST_DOMINATOR variable in m and n's hello messages since u has a larger node degree than v. Upon receiving these extended hello messages, dominator v knows that both its one-hop neighbors m and n can be covered by dominator u, and connector n also belongs to u.

Based on the extended hello message, a dominator may further determine whether all its one-hop neighbors can be covered by another dominator and the connectors that belongs to it. Therefore, the self-pruning procedure of the GMPR algorithm can be enhanced as follows:

Enhanced self-pruning procedure

- A dominator v is eliminated from a CDS if all its one-hop neighbors can be covered by a two-hop away dominator u and the connectors belong to u, where the connectors are in the one-hop neighborhood of v.

The enhanced self-pruning procedure can largely increase the probability of covering all the dominatees of a dominator and thus more dominators can be marked as silent-dominators and the size of a CDS is reduced.

Since no extra signaling messages are needed in the EGMPR algorithm, and only one node ID is appended in the signaling messages, the communication cost of the algorithm is the same as that of the GMPR algorithm. Moreover, since each

dominatee and connector only need constant time to decide which dominator they belong to, their overall workload of computation is still low. For dominators in the network, the extra computation costs induced by the enhanced self-pruning procedure are the processes of finding two-hop away dominators and their connectors, and deciding whether they are capable of covering all its dominatees. However, it will be proven in the next chapter that these processes has the same time complexity as the one in the GMPR algorithm.

3.4 Low-Cost Flooding Algorithm

The literature survey conducted in this research project shows that, most of the proposed broadcast algorithms do not have a linear time complexity and bounded approximation ratio, which are two important factors affecting the performance of an algorithm. These factors are crucial when an algorithm operates in dense networks, where the number of nodes in the network can be up to thousands. A broadcast algorithm must run fast and be scalable in these networks, and should guarantee its output result to be close to the optimal one in order to achieve efficiency in the worst case scenario. Considering the above issues, a *Low-Cost Flooding* (LCF) algorithm [47, 48] has been proposed as part of this research project. The term "cost" in the name of the algorithm represents its superior attributes of time and message complexity, signaling message size, and approximation ratio in comparison to the well known algorithms surveyed in Chapter 2. The LCF algorithm is especially useful for wireless sensor networks (WSNs) [49], which are ramifications of wireless ad hoc networks.

A WSN consists of a large number of multi-functional devices called sensor nodes that can be very small, low-cost and operate under very modest power re-

quirements. These sensor nodes integrate sensing, processing and transmission capabilities, and are deployed in a geographical area to collect, process and transmit data using wireless communications. Unlike the nodes in wireless ad hoc networks, sensor nodes normally have limited processing and memory capacity. However, when combining information collected by a large number of sensor nodes, they are capable of measuring a given physical environment in great detail. Due to their distributed property, WSNs have a wide range of application potential for both military and civilian purposes, such as intrusion detection, tactical surveillance, weather monitoring or sensing ambient conditions [50, 51].

Similar to the GMPR algorithm, the LCF algorithm also progresses in two phases. In the first phase, a network is clustered and cluster heads are elected. The cluster heads are referred to as *dominators* while the nodes they connect to are called *dominatees*. In the second phase, two types of connectors, *passive connectors* and *active connectors*, are selected from dominatees by each dominator to connect to its two-hop and three-hop away dominators respectively. A passive connector is selected by dominators or active connectors, and in contrast, an active connector can only be selected by dominators. So, during the progression of the LCF algorithm, a node initially starts in *white_node* state, and settles into one of the following three states: *dominator*, *dominatee* or *connector* until a topology change occurs. The nodes periodically broadcast hello (signaling) messages to notify their neighbours about their existence, or whenever their state changes. The information carried in these messages depends on the different states a node enters.

The remaining part of this section only discusses the process of connecting dominators in the network. During the discussion, it is assumed that each node in the network has a unique ID, and it learns the IDs of all its one-hop neighbors through their initial broadcast messages. Also, each wireless node has an omni-

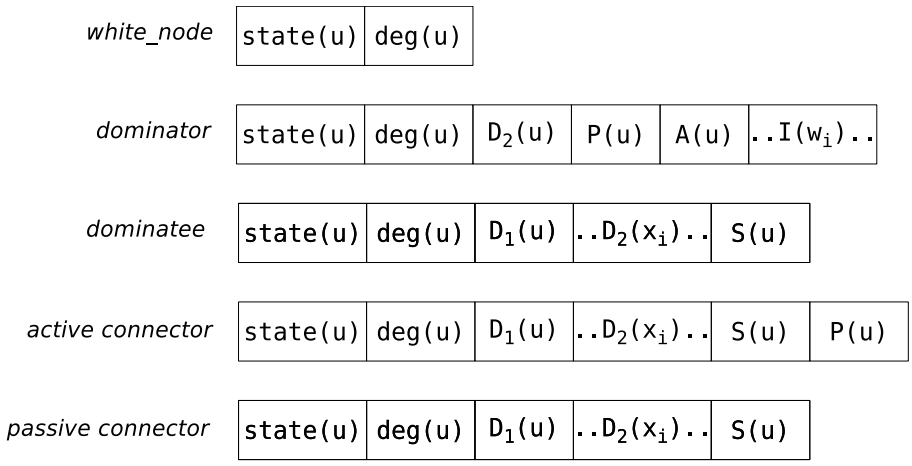


Figure 3.5: hello message formats of the LCF algorithm (fixed length source and destination address, and sequence number fields are not shown).

directional antenna with a maximum transmission range of one unit. Two nodes can communicate if the distance between them is within the maximum transmission range. Under these assumptions, a network topology can be modeled as a unit disk graph $G = (V, E)$ where V and E are the set of vertices and edges respectively [52].

3.4.1 Hello Message Formats of LCF Algorithm

In the LCF algorithm, the format of the hello message sent by a node may vary depending on different states that the node is in. Fig. 3.5 demonstrates different formats of the hello messages used in the LCF algorithm.

It can be seen that all hello messages contain two fields: the node state and the node degree, which occupy fixed length in the hello messages. For a dominator u, its hello messages contain the following additional variable length fields:

1. $D_2(u)$: the set of node IDs of u's two-hop dominators,

2. $P(u)$: the set of node IDs of u's passive connectors,

3. $A(u)$: the set of node IDs of u's active connectors, and

4. $I(w_i)$: for each node w_i in $A(u)$, the set of node IDs of isolated dominators that w_i needs to connect to.

For a dominatee u, its hello messages contain the following variable length fields:

1. $D_1(u)$: the set of node IDs of u's one-hop dominators,

2. $D_2(x_i)$: for each node x_i in the set $D_1(u)$, the set of node IDs of two-hop dominators of x_i, and

3. $S(u)$: the set of node IDs of special dominators of u.

The hello messages of an active connector u have all the fields of a passive connector's hello message plus an extra field, $P(u)$, which is the set of node IDs of passive connectors of u. A passive connector u has the same hello message format as a dominatee.

It can be seen that the LCF algorithm differs from the GMPR and EGMPR algorithms in its two types of connectors. A dominator selects a passive connector to link to other dominators that are two hops away from it, whilst a dominator selects an active connector and informs it the three-hop away dominators that it needs to connect to. Then the active connector also selects some passive connectors to finally connect to these three-hop away dominators. An example is shown in Fig. 3.6 to clearly explain the functions of these two types of connectors. In this network, in order to connect to two-hop dominators m and n, dominator p selects node c as a passive connector. To connect to three-hop away dominator k, dominator n selects node d as an active connector and informs it to connect to k. Then active connector d further chooses node e as a passive connector to link to k. Details of connecting dominators are described in the following sections.

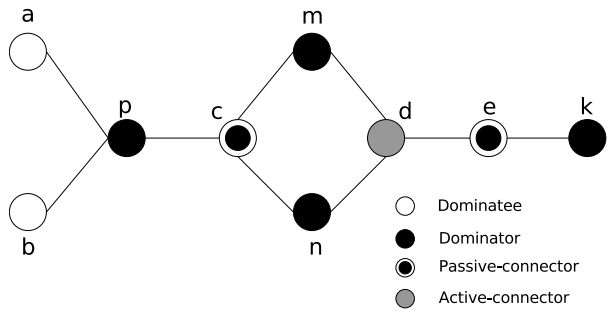


Figure 3.6: An example of utilization of two types of connectors in the LCF algorithm (the letters next to the nodes represent node IDs).

3.4.2 Connecting Two-Hop Away Dominators

After the dominator election process, nodes in the network enter either the dominator or dominatee states. A node in the dominatee state includes node IDs of its one-hop dominators in its hello messages (refer to the hello message formats in Fig. 3.5). Each dominator u uses this information to complete its set of two-hop dominators $D_2(u)$ and learns through which dominatees these dominators can be connected. A dominator u then includes node IDs of all dominators in $D_2(u)$ in its hello messages, and therefore, a nearby dominatee also has the knowledge of the two-hop dominators of u. For each dominatee v, dominator u also keeps a list of dominators in $D_2(u)$ that can be covered by v. After knowing its $D_2(u)$, a dominator u selects some passive connectors from its nearby dominatees to connect to other dominators that are two hops away. However, u excludes a dominatee v from the passive connector selection if u doest not have the largest node ID among v's one-hop dominators, and u removes v's one-hop dominators from its two-hop dominator set $D_2(u)$. Dominator u then chooses passive connectors from the remaining dominatees based on the following rule:

Passive connector selection (by dominators)

- Remove a selected passive connector and the two-hop dominators it covers

from the dominatee list and $D_2(u)$ respectively. If the $D_2(u)$ is not empty, for each remaining dominatee, calculate the number of dominators in $D_2(u)$ that it can reach. Choose a dominatee as a passive connector if it connects to the largest number of dominators in $D_2(u)$.

Each dominator includes node IDs of the selected passive connectors in the hello messages to inform its dominatees, and the selected dominatees change their states upon receiving these hello messages. Referring to Fig. 3.6, where dominators p, m and n have the knowledge of each other from the hello message sent by dominatee c. Also, dominators m and n know that neither of them is the largest dominator of c, and therefore, only dominator p may select passive connectors to link to its two-hop away dominators, and it eventually chooses dominatee c as the passive connector to connect to m and n.

3.4.3 Connecting Three-Hop Away Dominators

Since each dominator u includes IDs of nodes in $D_2(u)$ in the hello messages, a dominatee v has the knowledge of all of the two-hop dominators of u after it receives the messages from u. Dominatee v then includes node IDs of all its one-hop dominators in the hello messages, and for each included one-hop dominator w, dominatee v also includes the node IDs of the two-hop dominators in $D_2(w)$ in the hello messages. Upon receiving hello messages from other dominatees, a dominatee v checks the one-hop dominators of these dominatees, and it marks a dominatee x as a *special dominatee* if x does not share any dominator with v, and all the one-hop dominators of x are also marked as *special dominators*. For each special dominatee x, dominatee v keeps a list of x's special dominators, and v includes all special dominators in its hello messages.

After receiving the hello messages from dominatee v, a dominator u keeps a list of the two-hop dominators of each node in $D_2(u)$. It then checks the special dominators contained in the message and marks a special dominator as an *isolated dominator* based on the following rules:

1. it is not a dominator in $D_2(u)$,
2. it is not a two-hop dominator of any node in $D_2(u)$, and
3. it is not a previously marked isolated dominator.

Dominator u keeps node IDs of all the marked isolated dominators, and then it selects a dominatee v as an active connector based on the following criteria:

1. v has isolated dominators marked by u, and
2. u has the largest node ID among those isolated dominators and v's one-hop dominators.

Dominator u inserts the node IDs of the selected active connectors in the hello messages, and for each active connector w, u also inserts in the hello messages the node IDs of the isolated dominators that w needs to connect. A dominatee v changes its state to the active connector if it has been chosen by a dominator, and then v chooses some passive connectors to connect to these isolated dominators based on the following steps:

Passive connector selection (by active connectors)

- For each special dominatee stored in v, calculate the number of isolated dominators it has.
- Select a special dominatee as a passive connector if it covers the most number of isolated dominators.

Node IDs of the passive connectors are included in the hello messages sent by active connectors, and so dominatees change their states to passive connectors once they have been selected by the active connectors.

An example of the connection of three-hop dominators is shown in Fig. 3.7. In Fig. 3.7(a), dominatees d and e broadcast their hello messages first, and upon receiving these messages, dominatee c compares the node IDs of the one-hop dominators listed in them with its own list of one-hop dominators ($D_1(c)$). The comparison reveals that c shares the dominator m with e (both c and e can reach dominator m) but does not have any shared dominators with d. This leads c to mark d as a special dominatee and to keep a list of d's special dominators (k in this example). These messages are also received by dominators m and n.

Fig. 3.7(b) shows that after n receives e's hello message, it selects e as a passive connector by using the process described in Section 3.4.2. In the meantime, c also sends out a hello message to its neighbours.

In Fig. 3.7(c), p, upon receiving c's hello message, checks the list of the special dominators in the message and compares them with its own list of two-hop dominators ($D_2(p)$) and all the two-hop dominators of nodes in $D_2(p)$. In this particular case p compares special dominator k with p's two-hop dominator m and m's two-hop dominator n, and since there is no match, p marks k as an isolated dominator.

Since p has the largest node ID (IDs are ranked in alphabetical order for this example) among c's one-hop and isolated dominators (which are m and k respectively), p selects c as an active connector and informs c to connect to isolated dominator k by sending a hello message to c. Although m could also mark k as an isolated dominator, as it is not being the largest dominator around c and thus, does not select c as an active connector. In the mean time, d also broadcasts a hello message.

In Fig. 3.7(d), after being selected as an active connector by p, c checks each of its special dominatees and calculates the number of isolated dominators they cover. In this case, c knows that its only special dominatee d covers the isolated dominator k, and thus selects d as a passive connector. It should be noted that, k does not select d as an active connector since its node ID is smaller than d's isolated dominators p and m.

In the LCF algorithm, each dominator and active connector store a list of the node IDs of the connectors they have selected, and they send out hello messages immediately to inform other nodes only when their connector lists change. This approach reduces the number of signaling messages being exchanged in the network, but, increases the calculation time of the algorithm. However, it is shown in the next chapter that the size of the connector list is bounded by a constant, and consequently, so is the extra calculation time.

In general, the LCF has been proposed to improve the efficiency of broadcast algorithms in dense wireless ad hoc networks. The contributions of the LCF algorithm can be summarized as follows:

- Compared with other related algorithms, the LCF algorithm works more efficiently in dense networks in terms of reducing the number of unnecessary retransmissions. Simulation results presented in Chapter 5 confirm its supe-

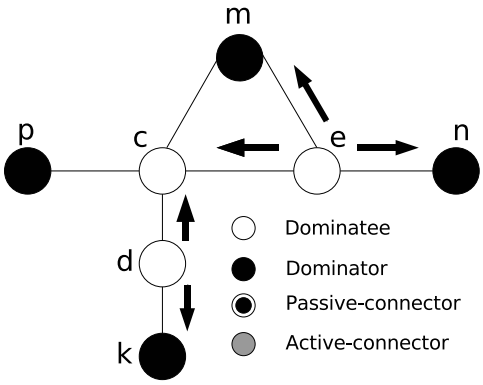
rior performance.

- Its communication and computational processes are lightweight.
- Its performance bounds are guaranteed since it has a constant approximation ratio.
- It has a signaling message size bounded by a constant (i.e. maximum message size is known), which helps on reducing the probability of collisions and transmission errors.

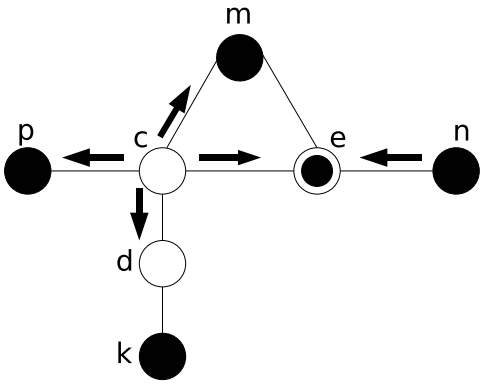
It is proven in the next chapter that the LCF algorithm has $O(n+\Delta)$ time complexity and $O(n)$ message complexity, and for a given network topology, the number of forwarding nodes generated by the algorithm is always smaller than the size of this network's smallest possible CDS multiplied by a real constant.

3.5 Summary

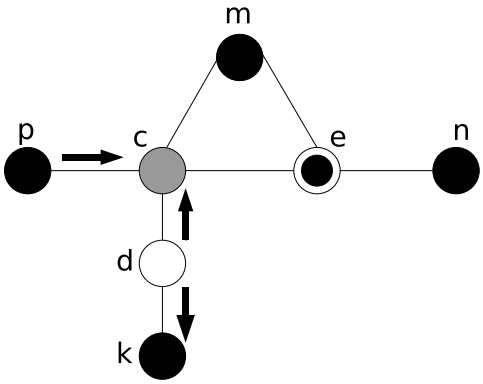
In this chapter, three new broadcast algorithms are presented that aim to minimize the redundant flooding of broadcast in wireless ad hoc networks. To achieve broadcast efficiency, the algorithms first form an MIS in a network where nodes in the MIS are able to reach all the other nodes in the network, but they are disconnected from each other. The algorithms then generate connectors to link to the nodes in the MIS in order to construct a CDS in the network. Only the nodes in the CDS retransmit broadcast packets thus significantly reducing the redundant transmissions. In the next chapter, the algorithms are validated, and their costs, which are defined in Section 2.4, are also analyzed in order to provide a clear comparison with the surveyed algorithms.



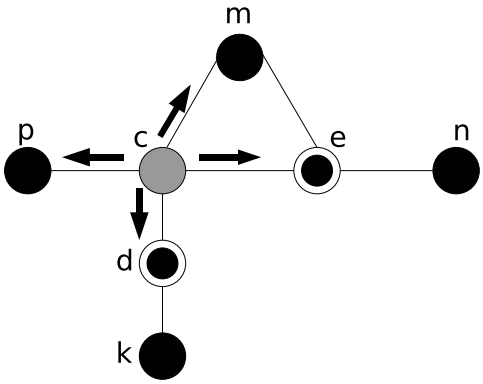
(a) c receives and processes the hello messages of d and e.



(b) n selects e as a passive connector.



(c) p selects c as an active connector.



(d) c selects d as a passive connector.

Figure 3.7: An example of connecting three-hop dominators. Arrows represent hello messages and their sending directions.

Chapter 4

Theoretical Analysis of Proposed Algorithms

4.1 Introduction

This chapter presents the validation of the three proposed algorithms. It is shown that the dominators and connectors generated by the algorithms in a network can form a CDS. In addition, the chapter presents an analysis of the performance of the algorithms based on the costs defined in Section 2.4. From the analysis, it can be seen that the the computational and communication load of the proposed algorithms on the network are light, and their signaling message size and approximation ratio are small. Especially the LCF algorithm has the best performance of all.

4.2 Validation of Proposed Algorithms

In this section, the proposed algorithms are validated. It is shown that the algorithms are able to generate a CDS in a given network of arbitrary topology. For the GMPR and EGMPR algorithm, it is also proven that the self-pruning procedure and the enhanced self-pruning procedure will not affect the CDS property of the generated set.

4.2.1 Verification of GMPR Algorithm

This section presents the validation of the GMPR algorithm. The correctness of the algorithm can be proven in two steps. First, it is shown that the gateways generated in the first phase of the algorithm form an MIS in the network, where nodes in the MIS can cover all other nodes in the network. Second, it is shown that the connectors generated in the second phase of the algorithm link the gateways thus forming a CDS in the network.

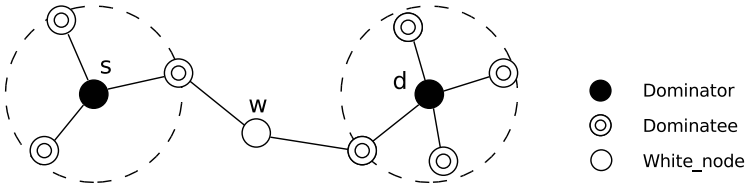


Figure 4.1: Illustration of the validation of IS.

Lemma 4.1. *Let H be the set of gateways generated in the GMPR algorithm for a given connected network $G = (V, E)$, where V is the set of nodes in G and E is the set of edges in G. Then H is an maximum independent set (MIS) of G.*

Proof. This lemma is proven in two steps. First, it is proven that H is an independent set (IS) in G. Then it is proven that no other nodes in G can be added to H,

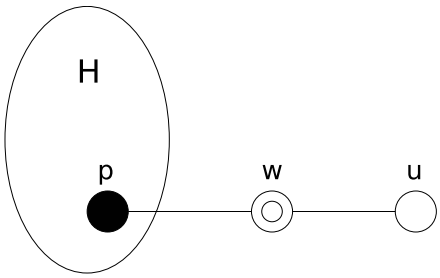


Figure 4.2: Illustration of the proof of MIS.

which establishes that H is an MIS in G.

The first part of the lemma can be proven using contradiction. Assume that there is a node w, which is not in H or does not have any neighbor in H. Then w must be at least two hops away from any given gateway in the network. This situation is shown in Fig. 4.1, where node s and d are gateways in H, and the dash circles represent the transmission range of the nodes. In this network, node w will eventually announce itself as a dominator based on the steps described in Section 3.2.1, since it does not have any dominator or white_node nearby. Therefore, node w is actually a gateway and it should be in H. This result contradicts the assumption, and thus H is an IS in G.

The second part of the lemma can also be proven by contradiction. Assume that H is not an MIS of G. There must then be a node u in G that is not a gateway and it can be added to H. This is illustrated in Fig. 4.2 where node p is the closest gateway to u in H. As can be seen, if u is in H, u must be at least one hop away from p since nodes in an IS are disconnected. Assume that node w connects to p and u, based on the gateway election procedure described in Section 3.2.1, node u will eventually be elected as a gateway, which contradicts the assumption. Therefore, by combining the above two parts, the lemma is proven. □

Lemma 4.2. *For any given gateway s in the MIS generated by the GMPR algo-*

rithm in a given network $G = (V, E)$, it is at most three hops away from the other gateways in the MIS.

Proof. This can be easily demonstrated based on Lemma 4.1. Assume that s is four hops away from d, which is the nearest gateway to s in the MIS. Referred to Fig. 4.1, in such a case, node w will eventually elect itself as a gateway, and then it becomes the nearest gateway of s, where the distance between them are three hops. Therefore, the largest distance between any given gateway s and other gateways in the MIS generated by the GMPR algorithm is at most three hops. □

Lemma 4.3. *All gateways are connected through the connectors generated in our algorithm.*

Proof. This lemma can be proven by contradiction. Let u denote the gateway that has no connector to other gateways. Based on Lemma 4.2, u is at most three hops away from its nearest gateways. If u has some gateways that are two hops away as shown in Fig. 4.3(a), let W denote the set that contains u and u's two-hop away gateways, there must be a node in W that has the highest priority (the largest node degree value $\deg$, or if there is a tie, the smallest node ID value), and this node will choose at least one node in $N_1(u)$ as the MPR to cover its two-hop neighbors. Based on the GMPR algorithm, the selected MPR becomes a connector, and thus, u is connected to at least one gateway which is two hops away.

If u has only gateways three hops away as shown in Fig. 4.3(b). u selects some nodes in $N_1(u)$ as MPRs to cover all nodes in $N_2(u)$, and based on our algorithm, these selected MPRs become connectors. Similarly, the three-hop away gateways also select some nodes in $N_2(u)$ as MPRs to cover their two-hop neighbors, and at least one selected MPR becomes a connector. Therefore, connectors in $N_2(u)$ can be reached by at least one connector in $N_1(u)$, and thus u is connected to at least

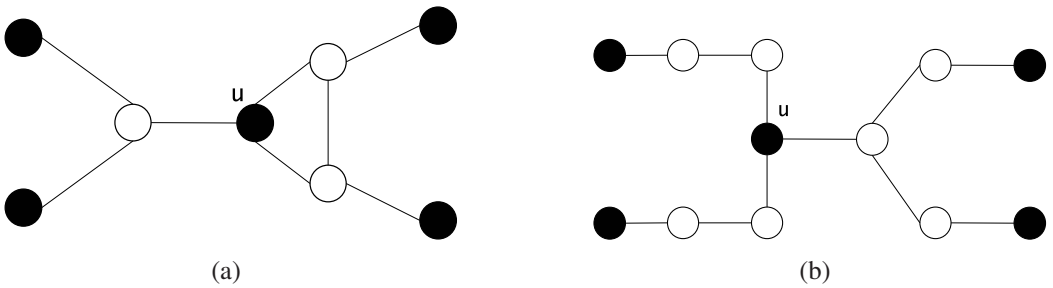


Figure 4.3: Illustration of the proof of Lemma 4.3.

one three-hop away gateway through two connectors. When combining the above two situations, it can be seen that there is no isolated gateway in V'. Therefore, the sub-graph induced by V' is connected. □

Theorem 4.1. *Given a connected network G, a node set V', which consists of gateways and connectors generated in the GMPR algorithm, is a CDS of G.*

Proof. The theorem is proven by combining Lemma 4.1 and Lemma 4.3. □

Theorem 4.2. *After the self-pruning procedure that is applied to each dominator generated by the GMPR algorithm, the remaining dominators and connectors still form a CDS.*

Proof. Since a silent-dominator generated by the self-pruning procedure still announces itself as a dominator, and it calculates an MPR set to cover its two-hop neighbors, its elimination from the CDS does not affect the construction of connectors. Furthermore, the connector that covers all one-hop neighbors of a silent-dominator can still ensure the connectivity of other connectors of that silent-dominator. Therefore, the connectivity of the CDS is maintained after the self-pruning procedure. □

From Theorem 4.1 and 4.2, it can be seen that, the GMPR algorithm can sufficiently construct a CDS in a given network.

4.2.2 Verification of EGMPR Algorithm

Due to the fact that the EGMPR algorithm is based on the same heuristic as the GMPR algorithm for constructing a CDS in a network, its validation can be proven in the same way as the GMPR algorithm. This section only validates the enhanced self-pruning procedure used in the EGMPR algorithm.

Theorem 4.3. *After running the enhanced self-pruning procedure, the remaining dominators and the connectors generated in the EGMPR algorithm still form a CDS in the network.*

Proof. For a given silent dominator v, since all one-hop neighbors of v are covered by another dominator and its connectors, the connectors in v's one-hop neighborhood are still connected thus ensuring the connectivity of gateways in the network. Furthermore, a silent dominator v still announces itself as a dominator and calculates MPRs to cover its two-hop neighbors, its elimination will not affect the connectivity of the CDS. □

4.2.3 Verification of LCF Algorithm

The correctness of the LCF algorithm can be also proven in two steps. First, it is proven that the dominators elected in the LCF algorithm form an MIS. Afterwards, it is proven that the selected connectors (passive and active connectors) can connect to all dominators, and thus a CDS can be constructed in the network. Since it the same strategy is applied as the GMPR algorithm to form a MIS in the network, the first step of the proof can be done by using the same procedure as described in Section 4.2.1.

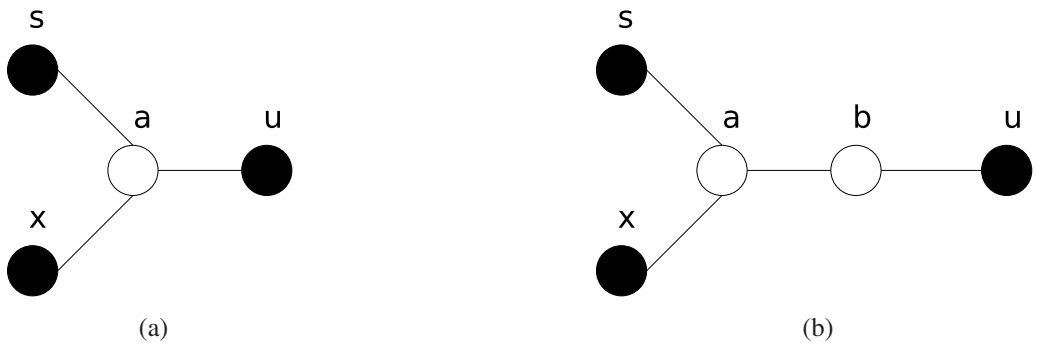


Figure 4.4: Illustration of the proof of Lemma 4.4.

Lemma 4.4. *Let V' be the subset of V that contains all dominators and connectors generated by the LCF algorithm. Then, the sub-graph induced by V' is connected.*

Proof. This lemma can be proven by contradiction. Suppose that there is a dominator u that is disconnected from other nodes in V', based on Lemma 4.2, u must be either two hops or three hops away from other dominators in V' as shown in Fig. 4.4. In Fig. 4.4(a), u is two hops away from s and x, and a is not in V', however, based on the passive connector selection rule described in Section 3.4.2, x will choose a as a passive connector to connect to s and u since x is the largest dominator of a, therefore, a is actually in V' and u is connected through a. In Fig. 4.4(b), u is three hops away from s and x, and both a and b are not in V'. Based on the active connector selection rule described in Section 3.4.3, a will be selected as an active connector by x since it has the isolated dominator u, and a further selects b as a passive connector to cover u, and hence, both a and b are actually in V' and u is connected. □

4.3 Performance Evaluation of Proposed Algorithms

This section investigates the performance of proposed algorithms. The costs of the algorithms defined in Section 2.4 are used to evaluate the performance, so a clear comparison can be provided against the surveyed algorithms presented in this thesis. Since proposed algorithms construct a CDS in the network, they are all source independent. The following sections describe the costs for each proposed algorithm.

4.3.1 Performance of GMPR Algorithm

For the GMPR algorithm, due to the fact that each dominator needs to know IDs of neighbor nodes within a two-hop range in order to calculate an MPR set, the information range of the GMPR is two hops. Since each dominator in the GMPR algorithm includes IDs of its one-hop neighbors and the MPRs it selected in the hello messages, its message size should be the same as the original MPR algorithm of $O(\Delta)$. However, since only dominators in the algorithm include IDs of MPRs in hello messages, the average message size of the algorithm is still smaller that the original MPR algorithm.

Theorem 4.4. *The GMPR algorithm has* $O(3\Delta M + \Delta + n)$ *time complexity and* $O(n)$ *message complexity.*

Proof. The time complexity of the GMPR algorithm is the time used to construct a CDS in the network. The time for constructing an MIS by the GMPR algorithm is $O(n)$, which can be proven using the similar method in [29]. For each gateway in the network, the time complexity for calculating an MPR set is the same as the original MPR heuristic, which is $O(3\Delta M + \Delta)$ (refer to Section 2.5.3). For each

dominatee in the network, since it has been proven in [42] that at most 5 dominators are in its one-hop neighborhood, only a constant time is need for the dominatee to decide whether its largest dominator has selected it as an MPR.

In the self-pruning procedure, each dominator first needs to find all connectors in its one-hop neighborhood, which may use $O(\Delta)$ time, and for each connector, the dominator needs at most $O(\Delta)$ time to check whether this connector can cover all its one-hop neighbors. Therefore, the overall time complexity of the self-pruning procedure is $O(\Delta^2)$. However, since this procedure is used after constructing a CDS in the network, it can be excluded from the overall time complexity of the GMPR algorithm. Therefore, the total time complexity of the GMPR algorithm is $O(3\Delta M + \Delta + n)$.

During the CDS construction of the GMPR algorithm, each gateway sends exactly one hello message to its one-hop neighbors to inform its state and its MPR decision. Upon receiving these hello messages, each dominatee and connector also send one hello message to their one-hop neighbors. Since there are at most 5 dominators connecting to a dominatee or a connector, at most five hello messages are sent for each dominatee or connector during the CDS construction. Therefore, the total message complexity of the GMPR algorithm is $O(n)$. □

Since each gateway in the GMPR algorithm selects forwarding nodes based on the original MPR heuristic, the approximation ratio of the GMPR algorithm should be the same as the original MPR heuristic as proven in Section 2.5.3.

4.3.2 Performance of EGMPR Algorithm

Since the EGMPR algorithm only enhances the self-pruning procedure of the GMPR algorithm, it should have the same performance as the GMPR algorithm. For the enhanced self-pruning procedure proposed in the EGMPR algorithm, it takes $O(\Delta)$ time for a dominator u to find all two-hop away dominators and their connectors, and at most $O(\Delta)$ time is spent on each two-hop away dominator in order to check whether it and its connectors can cover all dominatees of u. Therefore, total $O(\Delta^2)$ time is used for u to decide whether it is redundant in the CDS. Compared with the self-pruning procedure used in the GMPR algorithm, it can be seen that the enhanced self-pruning procedure does not introduce more cost to the algorithm. However, it is shown in the next chapter that the enhanced self-pruning procedure can effectively eliminate more redundant dominators than the one used in the GMPR algorithm.

4.3.3 Performance of LCF Algorithm

In the LCF algorithm, each dominatee node needs to include IDs of its one-hop dominators and also IDs of their two-hop dominators in the hello messages (refer to Fig. 3.5 in Section 3.4.1), which indicates that the algorithm requires a information range of three hops. This property may degrade the performance of the algorithm, because the information contained in the hello messages may be outdated when mobile nodes are moving fast in the network. However, since the initial aim of the LCF algorithm is to improve the broadcasting performance in dense sensor networks, where nodes are fixed or not moving rapidly, the information range property might not have significant impact on the performance of the algorithm.

In order to evaluate the other costs of the LCF algorithm, some lemmas are

needed to be present first.

Lemma 4.5. *For a given connected graph $G = (V, E)$, the size of any MIS of G is at most $4s_{opt} + 1$, where s_{opt} is the size of any minimum CDS of G.*

Lemma 4.6. *For every node u of a given connected graph $G = (V, E)$, the number of dominators inside the disk centered at u and of radius k-*units *is bounded by a constant l_k.*

Proofs of Lemmas 4.5 and 4.6 are presented in [44] and [42] respectively. From Lemma 4.6, it can be seen that for each node (dominator, dominatee or connector) u in the network, the number of dominators within k hops ($k \geq 2$) from u is bounded by a constant l_k. Based on the proof of Lemma 4.6 in [42], it can be deduced that for a node u,

$$l_k < \begin{cases} \frac{(k+0.5)^2 - 0.25}{0.25} & \text{if } u \text{ is a dominator} \\ \frac{(k+0.5)^2}{0.25} & \text{if } u \text{ is a dominatee or connector} \end{cases} \tag{4.1}$$

Theorem 4.5. *The LCF algorithm has $O(n+\Delta)$ time complexity and $O(n)$ message complexity.*

Proof. The time complexity of the LCF algorithm is composite of two parts: the time used to construct an MIS in the network, and the time used to generate connectors to link nodes in the MIS. For constructing an MIS in the LCF algorithm, it has been proven in [29] that at most $O(n)$ time is needed, where n is the total number of nodes in the network.

Refer to the procedures of generating connectors as described in Sections 3.4.2 and 3.4.3, since there is a maximum of 5 dominators connected to a dominatee as

proven in [42], for each dominator u, it takes at most $O(5\Delta)$ time to check whether it is the largest dominator of all its dominatees, and at most $O(|D_2(u)|\Delta)$ time is used to calculate passive connectors to cover all nodes in $D_2(u)$. To calculate active connectors, a dominator u spends $O(|D_2(v)|)$ time checking whether it is the largest dominator of its each dominatee v, and it takes $O(|D_2(v)|\Delta)$ time in the worst case to finish checking all u's dominatees. Then u needs $O(|D_3(u)|\Delta)$ time to calculate all active connectors in the worst case. For each dominatee v, it takes a constant time to find special dominators from a hello message of another dominatee, and for each active connector x, there are at most $|D_2(x)|$ isolated dominators that need to be connected, and it takes $O(2|D_2(x)|\Delta)$ time for x to determine the passive connectors to cover those isolated dominators. From Lemma 4.6, it is known that for each node m in the network, $|D_2(m)|$ and $|D_3(m)|$ are bounded by constant values, and therefore, the time complexity of the LCF algorithm should be $O(n + \Delta)$.

The message complexity for constructing an MIS in the LCF algorithm is $O(n)$ since each node only sends a constant number of signaling messages. During the process of connecting two-hop and three-hop dominators, each dominator sends out one hello message to select its passive and active connectors. Furthermore, an active connector also sends out one hello message to inform its passive connectors. Since each node sends a constant number of signaling messages, the total message complexity of the LCF algorithm is $O(n)$. □

Theorem 4.6. *The lengths of hello (signaling) messages used in the LCF algorithm are bounded by constants.*

Proof. In the LCF algorithm, all hello messages share a number of fixed length fields such as source and destination addresses, sequence number, node state and node degree, which all have fixed size. Assume that the total length of these fields

is h. A white_node's hello messages are clearly fixed length (refer to Fig. 3.5). Let l_d be the length of a dominator u's hello messages, the upper bound of the length of a dominator's hello messages can be presented as:

$$l_d = h + |D_2(u)| + |P(u)| + |A(u)| + |I(w_1)| + \cdots + |I(w_n)| \tag{4.2}$$

where $w_1, w_2, \cdots w_n \in A(u)$ and $n = |A(u)|$. Based on the inequality 4.1, the upper bounds of the lengths for $D_2(u)$, $P(u)$ and $A(u)$ can be determined as $|P(u)| \leq |D_2(u)| \leq 23$ and $|A(u) \leq |D_3(u)| \leq 47$ respectively. For the isolated dominator set $I(w_i)$ of each node w_i in $A(u)$, it has

$$\sum_{i=1}^{|A(u)|} |I(w_i)| = |I_s(u)|,$$

where $I_s(u)$ is the set of the total number of isolated dominators marked by u. It is easy to see that $|I_s(u)| \leq |A(u)| \leq 47$ (the total number of isolated dominators marked by u cannot exceed the number of u's three-hop away dominators), and therefore, by inserting these upper bounds into Equation 4.2, the following inequality can be deduced:

$$l_d \leq h + 23 + 23 + 47 + 47 \leq h + 140$$

The length (l_p) of hello messages of a passive connector or dominatee u can be written as follows

$$l_p \leq h + |D_1(u)| + |S(u)| + |D_2(x_1)| + \cdots + |D_2(x_n)| \tag{4.3}$$

where $x_1, x_2, \cdots, x_n \in D_1(u)$ and $n = |D_1(u)|$. Since only 5 dominators can

be located in the one-hop neighborhood of a passive connector, so $|D_1(u)| \leq 5$. Special dominators of u are limited by the size of $D_2(u)$, and from Inequality 4.1, it is known that $|S(u)| \leq |D_2(u)| \leq 24$. For each node x_i in $D_1(u)$, it has $|D_2(x_i)| \leq 24$. By inserting these upper bounds into Equation 4.3, it is easy to arrive at the following inequality for the length l_p of the hello messages of a passive connector or dominatee

$$l_p \leq h + 5 + 24 + 5 \times 24 \leq h + 149$$

An active connector's hello messages contain the $P(u)$ field in addition to the fields of a passive connector's ones, and $|P(u)| \leq 24$ (refer to Inequality 4.1). Therefore, the length (l_a) of an active connector's hello messages will be bounded by the following inequality

$$l_a \leq l_p + 24.$$

From above analysis, it can be concluded that signaling messages of the LCF algorithm are bounded by constants. □

Theorem 4.7. *For a given connected graph $G = (V, E)$, the LCF algorithm has a constant approximation ratio.*

Proof. Based on Lemma 4.6 it can be seen that, for each dominator u in the MIS, there are at most l_3 number of dominators that are within three hops distance from u. If it is assumed that each dominator u in the MIS is able to reach all dominators within three hops, a new graph $G' = (V', E')$ is created by the dominators, and the node degree of each node in G' is l_3. The total number of edges in G' is at most

$$|E'| = \frac{l_3 \times |V'|}{2},$$

where $|V'|$ is the number of nodes in G' [53]. Since one pair of connected nodes

Table 4.1: Cost comparison of proposed algorithms.

Algorithms	**Information range**	**Source dependent**	**Time complexity**	**Message Complexity**	**Message size**	**Approximation ratio**
GMPR	2 hops	No	$O(3\Delta M + 3\Delta + n)$	$O(n)$	$O(\Delta)$	Δ
EGMPR	2 hops	No	$O(3\Delta M + 3\Delta + n)$	$O(n)$	$O(\Delta)$	Δ
LCF	3 hops	No	$O(n + \Delta)$	$O(n)$	$O(1)$	$constant$

in G' contributes one edge, it is easy to see that the total number of pairs of dominators that within three hops away from each other in the MIS is at most $l_3|E'|/2$. The LCF algorithm generates at most one passive connector for a two hops away dominator pair, and one active, one passive connector to connect to a three-hop away dominator pair. Therefore, at most two connectors are generated for each pair of dominators that are within three hops distance from each other, and the total number of connectors generated by the LCF algorithm in G is $l_3|E'|$.

From Inequality 4.1 it is known that $l_3 < 48$. Also, Lemma 4.5 shows that the size of the dominator set H in graph G is at most $4s_{opt} + 1$. Therefore, the total number of connectors generated in G is no larger than $188s_{opt} + 47$, and for the total number of nodes (s_{lcf}) in the CDS generated by the LCF algorithm, the following inequality holds:

$$s_{lcf} \leq (188s_{opt} + 47) + (4s_{opt} + 1) = 192s_{opt} + 48$$

s_{lcf} is a constant multiple of s_{opt} (number of nodes in the minimum CDS). In other words, the LCF algorithm has a constant approximation ratio. □

4.4 Summary

In this chapter, the proposed algorithms have been verified, and their performances in terms of the costs defined in Section 2.4 have also been evaluated.

Table 4.1 summarizes the theoretical performance of the algorithms. It can be seen that all the proposed algorithms are source independent, which means that nodes in these algorithms do not need to check the sender of each packet which leads to saving of the processing time. The information range of the algorithms is within three hops. For the LCF algorithm, it has information range of three hops. However, since the LCF algorithm is proposed to operate in a low-mobility environment, its larger information range requirement will not significantly affect its performance.

It can be seen that the proposed algorithms have lightweight computation and communication complexity, and especially the LCF algorithm has linear time and message complexities with respect to the growth of density of a network. Also, the LCF algorithm has bounded signaling message size and approximation ratio, which guarantee its performance under any given network topology. In the next chapter, simulation based experiments are conducted to compare the proposed algorithms with some leading related algorithms to establish their efficiency.

Chapter 5

Simulation Studies of Proposed Algorithms

5.1 Introduction

In order to carry out a practical evaluation of the performance of proposed algorithms, simulation based experiments were conducted in this research project. Nowadays, computer based discrete-event simulation has widely accepted to be a valuable tool in many areas where analytical methods are not applicable and experimentation is not feasible [54]. The mainstream approach in the MANET research community usually follows the development, simulation, and publish process, and MANET publications normally include performance simulations that compare different protocols. In this research project also, simulations were used to test the proposed broadcast algorithms against some selected leading broadcast algorithms surveyed in Chapter 2. Several aspects of the performance of the algorithms are tested including (a) the number of forwarding nodes generated, (b) the average number of

signaling messages sent by each node and (c) the average signaling message size. These attributes affect the performance of broadcast algorithms thus influencing the lifespan of the network. Details of the simulation environment and the analysis of results are presented in this chapter.

5.2 Simulation Framework

Simulation studies in this research were conducted based on the OMNeT++ [55] simulator, which is a discrete-event simulation platform written in C++ programming language [56]. The main advantage of the OMNeT++ is that it is highly modular. An OMNeT++ model consists of hierarchically nested modules. A simple module that is written using C++ resides at the lowest level of the hierarchy. Its functionality is determined by the module designer. A compound module then is formed by combining an arbitrary number of simple or other compound modules. Different modules can communicate with each other via passing messages, which can be used to simulate communication links in the network. Currently, many models have been published for OMNeT++ to simulate fields of Internet, mobility and ad hoc networks. Details of the simulator can be found in [55].

The simulation model used in this research is the Mobility Framework (MF) [57], which is one of the existing add-on model suites for simulating wireless, ad hoc and sensor networks within the OMNeT++. The core framework of the MF implements support for node mobility, dynamic connection management and a wireless channel model. Moreover, it provides a *basic module*, which can be further enhanced by users to implement their own modules for different purposes. This concept enables designers to develop protocols for the MF without having to deal with the necessary interface and interoperability issues. Additional information on

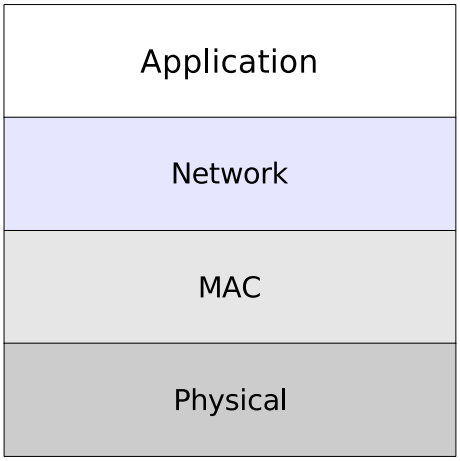


Figure 5.1: The protocol stack used in Mobility Framework.

the MF model suite can be found in Appendix A.

In the MANET domain, researchers usually apply an ISO/OSI[1] like protocol stack in their simulations. Fig. 5.1 illustrates the MANET protocol stack used in the MF. In this stack, the levels above the network layer are combined into a single application layer, which has the duties of application traffic generation and transmission rate control. In this experiment, new modules implementing efficient broadcast algorithms were created as application layer protocols that run on top of the MANET protocol stack. A broadcast algorithm in the application layer uses messages sent from the lower layers to carry out forwarding node calculations. Results of the calculation are then sent down to the network layer.

In this study, a modified network layer module was also created, which did not perform any routing scheme since it only needed to broadcast the application packets sent down from the upper layer module. The main function of the network layer was to check the sequence number and Time-To-Live (TTL) fields in the header of each received packet to decide whether to discard a packet. An application packet was encapsulated into a network layer packet, whose destination

[1] http://www.ussg.iu.edu/usail/network/nfs/network_layers.html

address in the header is set to the broadcast address, and the source address was set to the its own address. It also set the TTL field in the packet header to one, so the broadcast packet could not be relayed by one-hop neighbors of the host. The packet is then send down to the MAC layer.

In the MF, an IEEE 802.11b model is implemented to work in the MAC layer. In the early stage of this research, this model was used in the simulation. However, it can largely increase the run time of the simulation of dense network topologies. Since the aim of this research was to evaluate the efficiency of broadcast algorithms, which does not consider packet transmission rate, end-to-end delay or packet loss, a MAC layer was then considered to be perfect in the experiment, i. e. there are no contentions and collisions in the network. However, in order to avoid hosts sending packets at the same time, each host waited a random time between 0 and 10 seconds before it sent out the packet. The packet sent from the network layer was packed into the MAC layer packet, and in the packet header, the destination MAC address was set to the broadcast address and the source MAC address was set to the sender's own address. The packet was then sent down to the physical layer.

The physical layer in the MF carried out a signal strength evaluation. The calculation of a received signal strength was based on the global parameters like *transmitterPower*, *carrierFrequency* and *pathLostAlpha*. Details of the signal strength calculation can be found in Appendix A. A packet was discarded by the physical layer module if the signal strength is lower than a predefined threshold. In this project, it was assumed that the network had a perfect physical channel, so there is no packet loss due to the signal attenuation. Dynamic connections between hosts were done by a centralized module in the MF. Detailed information on connecting hosts in MF is presented in Appendix A.

5.3 Simulation Scenario

The simulation experiment was conduced with the input parameters set as stated in Appendix B. The simulation network used in the experiment was defined as a 100m $\times$ 100m two-dimensional area, where nodes were randomly deployed in this area. In this research, it was assumed that the nodes in the network were static, and they had the same maximum transmission range. Two nodes were able to communicate with each other if the distance between them is smaller than the maximum transmission range. To ensure connectivity of the network, first, a random node that was previously located in the network was selected, and then a new node was randomly placed within the transmission range of this node. The C++ code used to generate connected topologies is presented in Appendix B. Fig. 5.2 demonstrates an example of the simulation network area and a typical network topology that consists of 100 nodes. As can be seen, the area in white color is the simulation ground. The dark nodes in the network represent individual hosts, where the lines indicates wireless communications between the hosts.

The simulation experiment tested two types of networks: sparse and dense networks. Different densities of a network can be achieved by increasing the total number of nodes N in the network. In the experiment, the value of N ranged from 100 to 500 with an interval of 100. Moreover, three transmission ranges (R = 15m, 25m and 50m) were tested separately for each network topology in order to further change the density of the network. For each N in the network, a sufficient number of runs were conducted, and for each run, an unique seed was used to generate a different network topology.

In the experiment, the determination for the number of simulation runs was based on the objective of 95% confidence interval (CI) [58] and an approximate

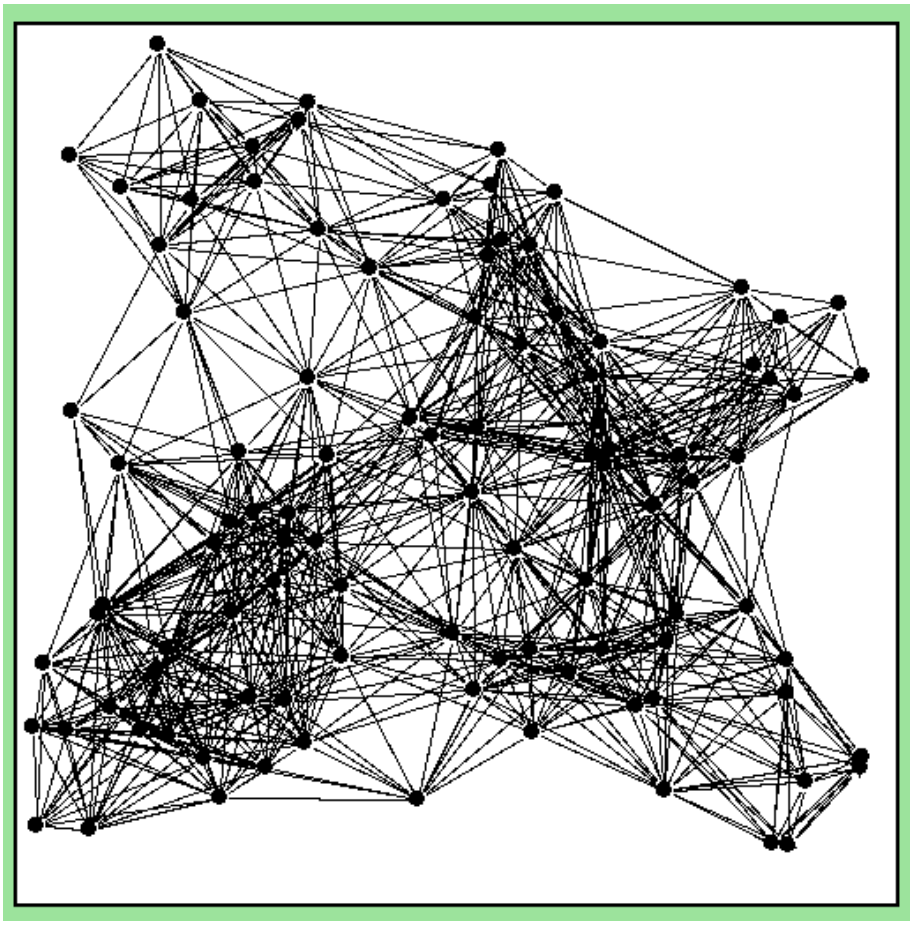


Figure 5.2: An illustration of the simulation network area used in this research project and a typical network topology generated by randomly placing 100 nodes in this area.

precision of $\pm 5\%$ of the sample mean[2]. Equation 5.1 [59] shows the formula to determine the number of sufficient runs for a given CI and an estimated error bounds E.

$$n = (\frac{z_{a/2}\sigma}{E})^2 \tag{5.1}$$

The value of $z_{a/2}$ is approximately 1.96 for a normal distribution, and σ is the standard deviation of sample means, which is unknown. However, it can be estimated

[2]95% of the chance that actual mean (population mean) of the parameter being measured is within the interval defined by the sample mean plus or minus an $\pm 5\%$ of the sample mean.

from the samples as

$$\sigma^2 = \frac{1}{n-1}\sum_{i=0}^{n-1}(\overline{X} - X_i)^2, \tag{5.2}$$

where n is the total number of samples, $\overline{X}$ is the sample mean and X_i is each sample. Based on the preliminary try, it was shown that 21 simulation runs were sufficient to achieve the goal of 95% confidence level for the experiment.

In the experiment, three performance metrics were tested:

Number of generated forwarding nodes This metric indicates the broadcasting efficiency of an algorithm. Generally, it is desirable for a broadcast algorithm to generate a small number of forwarding nodes in the network. In the experiment, this metric was measured by counting nodes in the forwarding state in the network.

Average number of signaling messages This metric reflects the message complexity of an algorithm, which may significantly affect the lifespan of a MANET since a large number of signaling message exchanges can consume a considerable amount of node's power. In the experiment, this metric was measured according to the average number of signaling messages sent by each node during the calculation of forwarding nodes. The periodical hello messages were not counted in.

Average signaling message size This metric also affects the overall performance of a broadcast algorithm. Large messages usually need more time to be processed and transmitted, which increase the end-to-end delay and the probability of collisions in the network. This metric was measured by counting the fields contained in signaling messages, where the size of a field was in the unit of node ID.

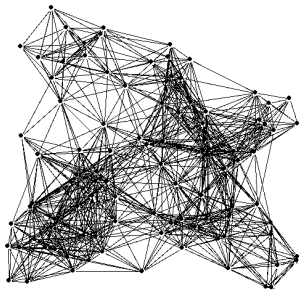
5.4 Analysis of the Results

In the simulation experiment, the original MPR algorithm (MPR) [15], MPR-CDS algorithm (MPR-CDS) [38], Enhanced MPR algorithm (EMPR) [39], Extended Enhanced MPR algorithm (EEMPR) [41], and the message-optimal CDS (MOCDS) [29] algorithm were used to compare with the three proposed algorithms namely the Gateway MPR (GMPR) algorithm [45], Enhanced Gateway MPR algorithm (EGMPR) [46], and the Low-Cost Flooding algorithm (LCF) [47, 48]. For the GMPR algorithm, it was tested separately with and without the self-pruning procedure in order to show the effectiveness of the self-pruning procedure. The rest of this section discusses the simulation results.

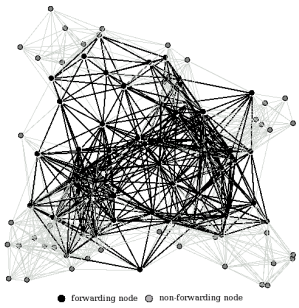
5.4.1 Comparison of the Number of Forwarding Nodes

The first test was the number of the forwarding nodes generated by each algorithm. Since the number of forwarding nodes strongly influences the number of retransmissions, it can significantly affect the overall energy consumption. In general, it is desirable to reduce the number of forwarding nodes in order to limit unnecessary retransmissions in the network. Fig. 5.3 demonstrates the differences of the generated forwarding node set by the algorithms tested in this experiment. The network topology used in this example is the same as the one shown in Fig. 5.2. The network contains 100 nodes and the transmission range of each node is 25m.

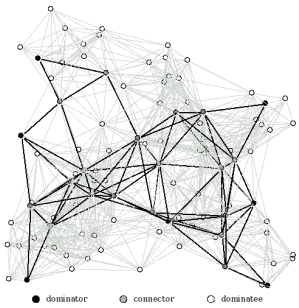
As can been seen from the figures, each algorithm constructs forwarding node sets of different sizes in the network, the number of forwarding nodes generated in Fig. 5.3(b) to Fig. 5.3(j) are 56, 25, 26, 22, 21, 16, 14, 17 and 17 respectively. It can be seen that in this relatively sparse network topology, all proposed algorithms



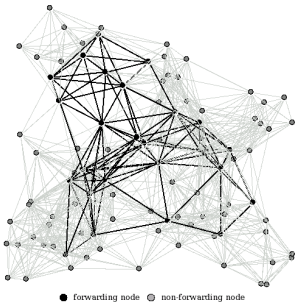
(a) Original network topology.



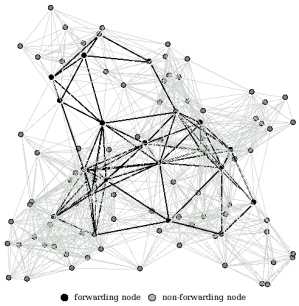
(b) The original MPR algorithm.



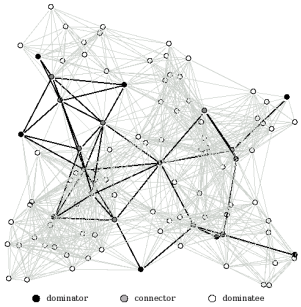
(c) The MOCDS algorithm.



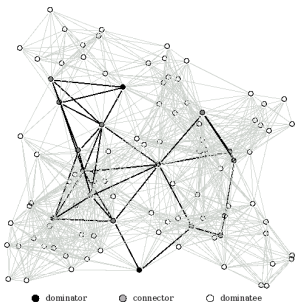
(d) The MPR-CDS algorithm.



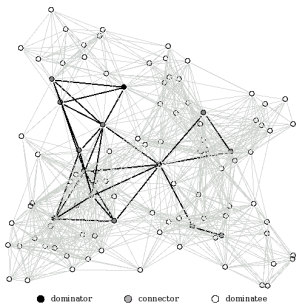
(e) The EMPR algorithm.



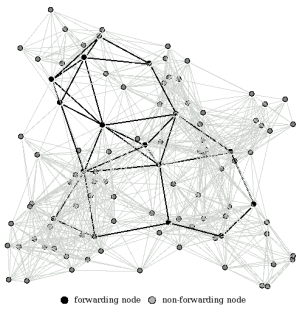
(f) The GMPR algorithm (without self-pruning procedure).



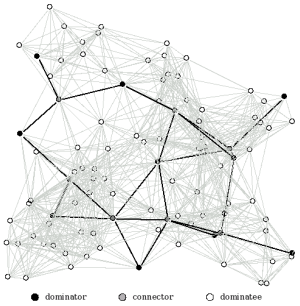
(g) The GMPR algorithm (with self-pruning procedure).



(h) The EGMPR algorithm.



(i) The EEMPR algorithm.



(j) The LCF algorithm.

Figure 5.3: A comparison of the number of forwarding nodes generated by different algorithms for a given 100-node network topology.

have better performance than the other ones. It is also worth noting that the LCF algorithm generates forwarding nodes that their coverage overlap with each other less. Therefore, we can say that the algorithm can reduce the overall transmission interference in the network. This property could be due to the fact that the LCF algorithm utilises the special dominators in the process of selecting the passive connectors, and thus fewer connectors are generated in the one-hop neighborhood of a dominator.

Figures 5.4 to 5.6 show the complete results of the forwarding nodes generated by the algorithms in three transmission ranges. In general, all the algorithms generate fewer number of forwarding nodes when the transmission range increases. This can be easily explained since the node coverage of a forwarding node increases along with the node transmission range, and thus fewer forwarding nodes are needed to cover all the nodes in a network. As can be seen from the figures, the original MPR algorithm does not work well in all three scenarios. The number of forwarding nodes it generated is around 45% of the total number of nodes in the network. This is due to the fact that each node in the original MPR algorithm calculates MPRs to cover its two-hop neighbors without considering the MPRs selected by other nodes in the network. Therefore, MPRs cover largely overlapping fields and contribute limited coverage of nodes in the network. This drawback has been pointed out and improved by the MPR-CDS, EMPR and EEMPR algorithms, which reevaluate each MPR and decide whether it is necessary to be a forwarding node. As shown in the figures, the performance of these algorithms are much better than the original MPR algorithm. Especially for the EEMPR algorithm, it largely reduces the number of generated forwarding nodes in the network. However, this algorithm has high computation complexity, and it requires a significant amount of neighbor node information, which is proven to be as large as $O(\Delta^2)$ in Section 2.6.3. Since it took

too much time and memory space to run this algorithm for the transmission range of 50m, only results up to 300 nodes are shown in this thesis.

The MOCDS algorithm also performs badly in all transmission ranges. This is due to the reason that each dominator u generated by MOCDS algorithm chooses a dominatee v as a connector without considering v's coverage of the dominators. A dominatee v is selected by the dominator u to connect to a two-hop or a three-hop away dominator if v comes first to notice u that it can reach these dominators. This strategy may cause a dominator to ignore some better placed dominatees that can cover bigger number of two-hop or three-hop away dominators thus increasing the total number of connectors generated in the network.

From the figures, it can be seen that the proposed algorithms have better performance than the other ones in all transmission ranges. It is noted that the self-

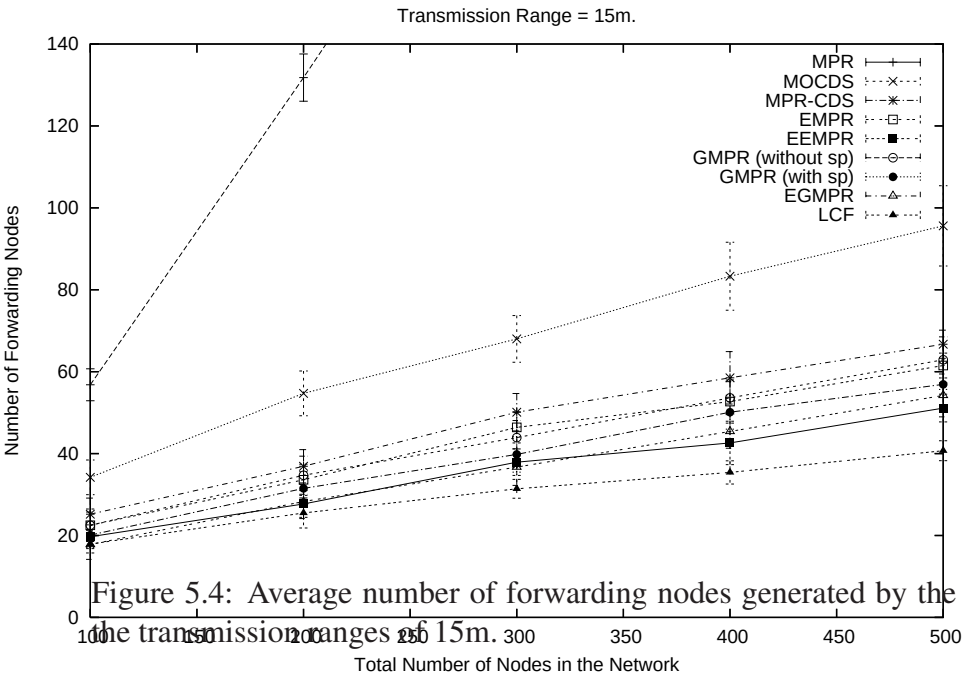


Figure 5.4: Average number of forwarding nodes generated by the algorithms for the transmission ranges of 15m.

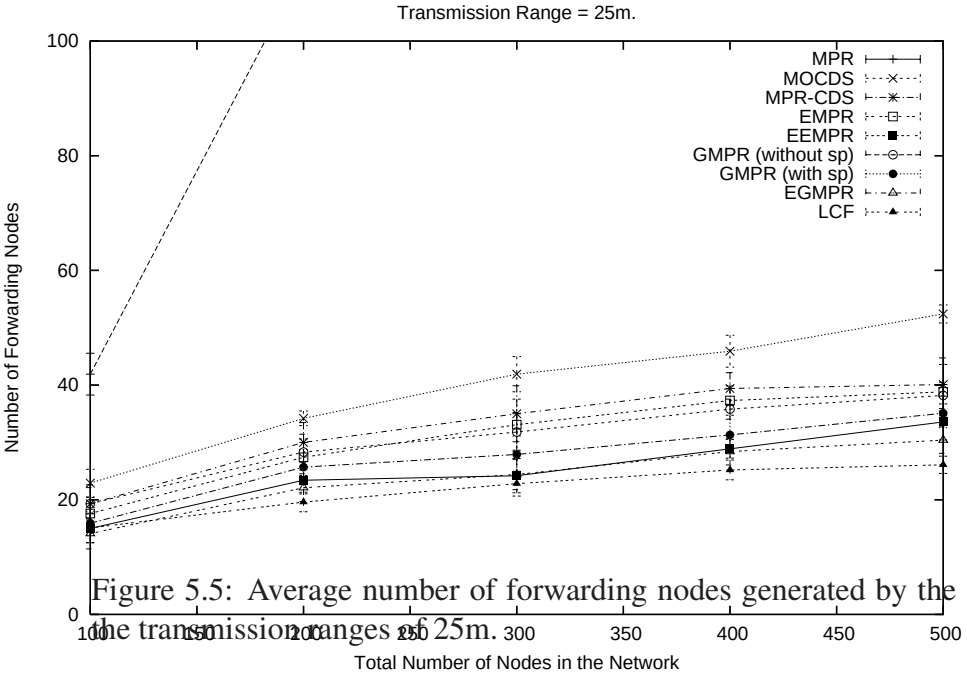


Figure 5.5: Average number of forwarding nodes generated by the algorithms for the transmission ranges of 25m.

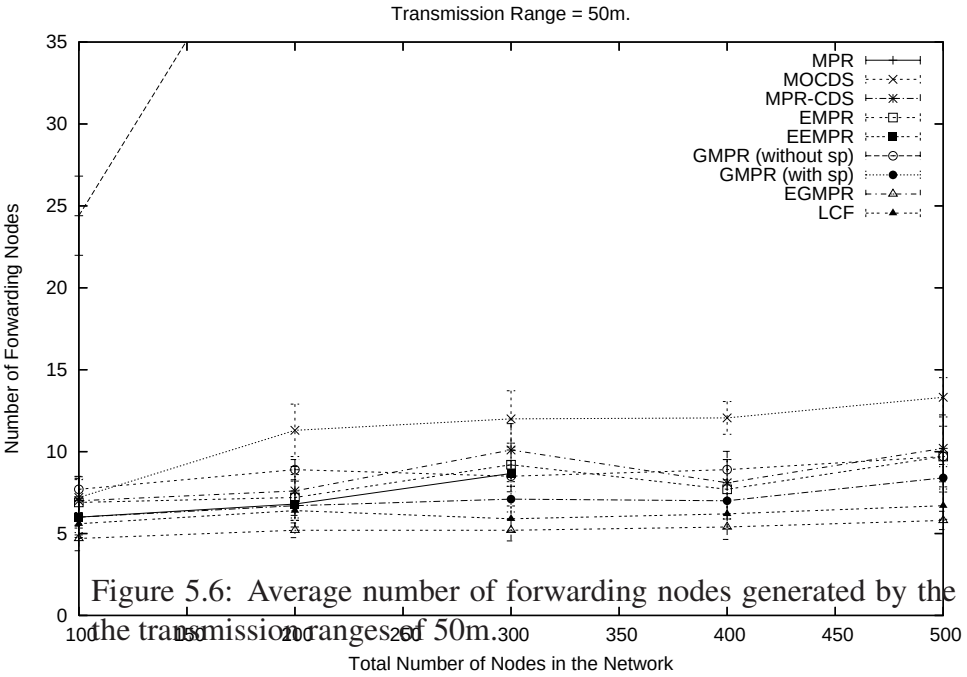


Figure 5.6: Average number of forwarding nodes generated by the algorithms for the transmission ranges of 50m.

pruning procedure used in the GMPR algorithm can largely minimize redundant dominators in the generated CDS. Moreover, the procedure works more efficiently when the transmission range is large. This is because that a connector generated by the GMPR algorithm may have more chance to cover all one-hop neighbors of a dominator in a larger transmission range. Therefore, the probability of finding a redundant dominator in the CDS is also increased. It is worth noting that the EGMPR algorithm can successfully improve the GMPR algorithm by generating fewer number of forwarding nodes in the network. This improvement indicates that the enhanced self-pruning procedure proposed in the EGMPR algorithm can increase the probability of finding redundant dominators in the generated CDS. As shown in Fig. 5.6, the EGMPR algorithm has the best performance among all the algorithms, where the number of forwarding nodes it generated is almost constant regardless of the increase of total nodes in the network.

Among the tested algorithms, the LCF algorithm has the best performance in general. It generates the fewest number of forwarding nodes in both 15 and 20m transmission ranges, and it has nearly the same results as the EGMPR algorithm in 50m transmission range. It is noted that the LCF algorithm works more efficiently in dense topologies such as N = 400 or 500. In these topologies, the number of forwarding nodes generated by the LCF algorithm is around 20% lower than the one of the EEMPR algorithm, and 50% lower than the one of the MOCDS algorithm. This remarkable achievement is mainly due to two reasons. First, only the dominator that has the largest node degree value among its two-hop or three-hop away dominators selects connectors. Second, the algorithm uses the concepts of special dominators and isolated dominators to evaluate the connector candidates, and therefore, only a small number of connectors are generated in the one-hop neighborhood of a dominator. The simulation results of the LCF algorithm are consistent with its

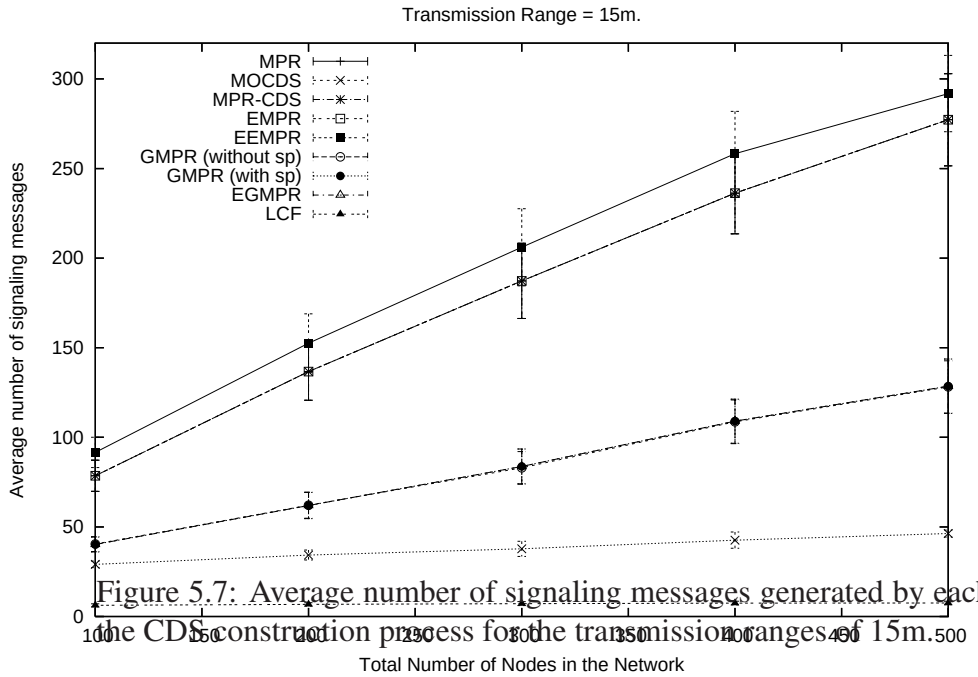


Figure 5.7: Average number of signaling messages generated by each node during the CDS construction process for the transmission ranges of 15m.

aim, which is to achieve broadcasting efficiency in dense networks. The results also confirm the theoretical analysis of the algorithm conducted in Chapter 4, where it is proven to be scalable against the density of the network.

5.4.2 Comparison of the Number of Signaling Messages

Figures 5.7 to 5.9 compare the average number of signaling messages sent by each node in different algorithms. As illustrated, the EEMPR algorithm generates the highest number of signaling messages of all, followed by the MPR-CDS, EMPR and original MPR algorithms, which have almost the same results. For these algorithms, the reason for a high number of signaling messages is that each node in the network recalculates MPRs whenever the conditions of its neighborhood have changed. Each node then sends out a signaling message to its neighbors to indicate

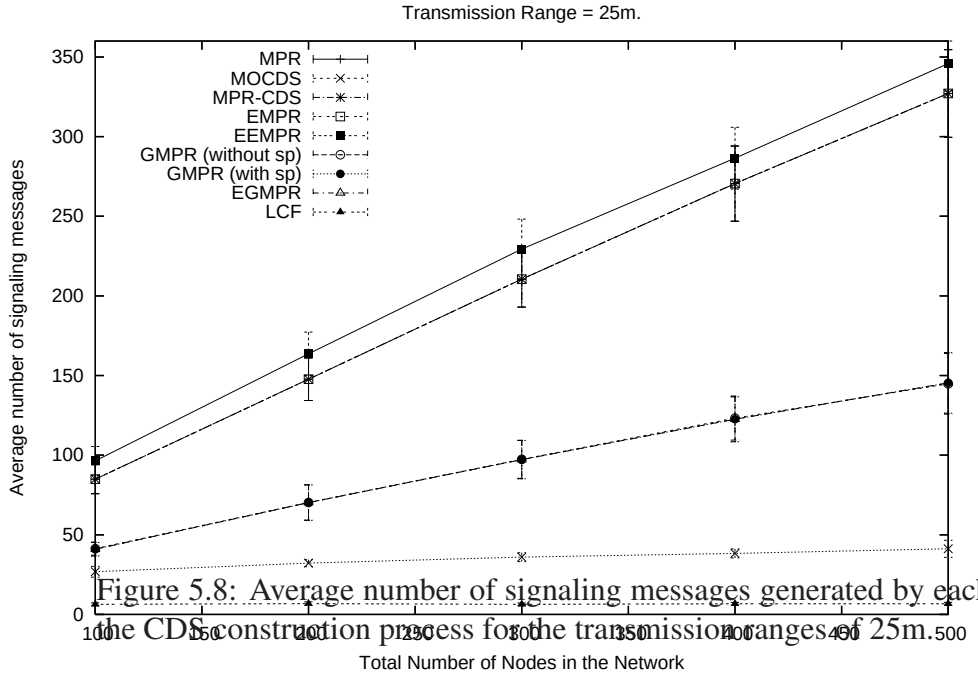


Figure 5.8: Average number of signaling messages generated by each node during the CDS construction process for the transmission ranges of 25m.

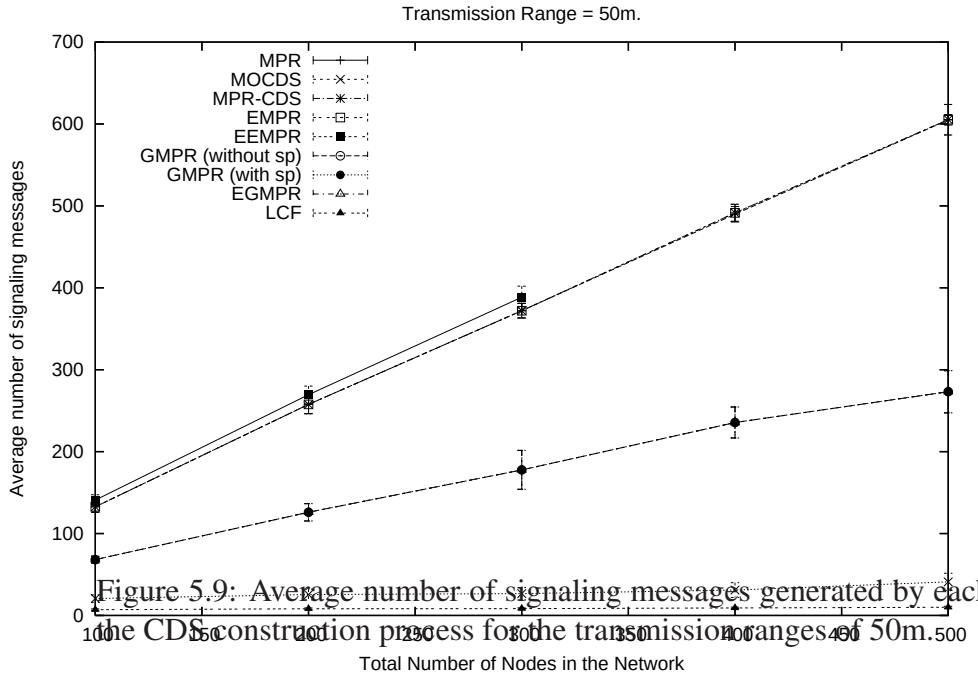


Figure 5.9: Average number of signaling messages generated by each node during the CDS construction process for the transmission ranges of 50m.

its new MPR decisions. For the EEMPR algorithm, the one-hop MPRs also need to calculate some two-hop MPRs, and then they send out signaling messages to inform the selected two-hop MPRs. Therefore, the EEMPR algorithm generates more signaling messages than the MPR-CDS, EMPR and original MPR algorithms.

From the figures, it can be seen that the number of signaling messages generated by the proposed GMPR and EGMPR algorithms is around half of the ones generated by the MPR-CDS and EMPR algorithms. This is because that only gateway nodes in the proposed algorithms calculate MPRs and send signaling messages to inform their selected MPRs. Therefore, the average number of signaling messages sent by each node in these algorithm is largely reduced. For the EGMPR algorithm, since it only differs in the self-pruning procedure with the GMPR algorithm, it generates the same number of signaling messages as the GMPR algorithm.

It is worth noting that the LCF and MOCDS algorithms generate a much smaller number of signaling messages than the others, and particularly for the LCF algorithm, it generates the fewest number of signaling messages in all transmission ranges. This is mainly due to two reasons. First, not all dominators in the LCF algorithm need to select passive and active connectors, and only the dominators that have chosen some passive or active connectors need to send signaling messages. Second, each dominator and active connector keep a list of connectors they have selected, and they send signaling messages only if the list changes. This procedure increases the time complexity of the algorithm since it needs to check the connector list every time after the forwarding node calculation. However, since the number of connectors selected by each dominator and connector in the LCF algorithm is always bounded by constants as proven in Section 4.3.3, the time for the procedure to complete has to be bounded.

5.4.3 Comparison of the Average Signaling Message Size

Results of average signaling message sizes of different algorithms are presented in Tables 5.1 and 5.2. The message sizes are in units of node IDs which can be 16-bit or 48-bit MAC addresses. Algorithms in the tables are ranked based their average signaling message sizes. As can be seen from the tables, the average sizes of signaling messages of all the algorithms increase along with the network density and the transmission range. This can be easily explained since the number of neighbors of a node increases along with the density of the network. Therefore, more neighbor information is included in signaling messages.

Among the tested algorithms, the proposed LCF algorithm has the smallest signaling message sizes in all transmission ranges. Its signaling message size is more than 50% lower than the EGMPR algorithm. This remarkable achievement can be explained by referring to Section 4.3.3, where its message size is proven to be bounded by constants. This property ensures that the message size of the

Table 5.1: Average message size (in units of node IDs) for $R = 15m$.

	Total number of nodes				
	100	**200**	**300**	**400**	**500**
LCF	15.79	25.55	34.03	43.13	51.48
MOCDS	17.50	26.15	31.92	41.10	51.49
GMPR (without SP)	28.92	48.69	65.24	80.91	94.01
GMPR (with SP)	28.92	48.69	65.24	80.91	94.01
EGMPR	29.93	49.68	66.24	81.86	95.11
MPR	32.02	52.40	70.06	86.91	100.86
MPR-CDS	32.02	52.40	70.06	86.91	100.86
EMPR	53.58	91.53	125.08	157.25	183.95
EEMPR	911.24	2653.06	4946.21	6809.10	8136.01

Table 5.2: Average message size (in units of node IDs) for $R = 25m$.

	Total number of nodes				
	100	**200**	**300**	**400**	**500**
LCF	16.36	27.85	35.45	46.51	55.52
MOCDS	17.94	28.25	39.14	50.21	53.65
GMPR without SP	31.42	52.47	74.11	92.74	112.11
GMPR with SP	31.42	52.47	74.11	92.74	112.11
EGMPR	32.42	53.46	76.08	93.71	113.87
MPR	34.41	56.9	78.51	98.93	118.62
MPR-CDS	34.41	56.9	78.51	98.93	118.62
EMPR	58.57	101.06	142.74	182.43	220.92
EEMPR	1056.63	3096.98	5888.12	7464.24	8945.48

LCF algorithm do not depend on the number of nodes in the network, and thus it is more scalable in dense networks. It is noted that the proposed GMPR and EGMPR algorithms also have relatively small signaling message sizes. This is due to the fact that in these algorithms, only the signaling messages sent by gateways need to carry IDs of selected MPRs and one-hop neighbors, while other signaling messages only carry IDs of one-hop neighbors.

The MOCDS algorithm has the second best performance among the tested algorithms, where its signaling message size is slightly larger than the LCF algorithm. This result confirms the theoretical analysis of the MOCDS algorithm conducted in Section 2.6.3, where the algorithm is proven to have bounded signal message size. For the original MPR and the MPR-CDS algorithms, they have the same signaling message size since they require the identical node information. For the EMPR algorithm, it has a larger average signaling message size than the original MPR and the MPR-CDS algorithms. This is because in the EMPR algorithm, each node in the network adds IDs of its free neighbors in the signaling messages thus increasing

Table 5.3: Average message size (in units of node IDs) for $R = 50m$.

	Total number of nodes				
	100	**200**	**300**	**400**	**500**
LCF	16.02	27.46	37.40	47.35	58.59
MOCDS	17.47	29.38	34.20	53.39	70.18
GMPR without SP	48.14	92.76	135.87	173.71	202.29
GMPR with SP	48.05	92.69	135.80	175.40	207.90
EGMPR	48.90	93.68	136.81	176.97	208.79
MPR	50.33	94.97	137.84	180.46	221.13
MPR-CDS	50.93	96.11	139.51	181.05	221.25
EMPR	94.094	183.61	269.81	352.27	432.27
EEMPR	2511.23	9280.35	18667.90		

the average signaling message size.

It is worth noting that the EEMPR algorithm has a much larger signaling message size, which is 20 times larger than the other ones in average. This is due to the fact that the algorithm requires each node in the network to include IDs of its one-hop neighbors and also IDs of their one-hop neighbors in signaling messages. Therefore, the total messages size can reach $O(\Delta^2)$ as shown in Section 2.6.3. Although the EEMPR algorithm is able to generate a small number of forwarding nodes in the network, it requires much more time and memory space to check and store the information contained in its signaling messages than other algorithms. Therefore, it is estimated that the EEMPR algorithm may face severe scalability problem in dense networks.

5.5 Summary

This chapter discussed the details of the simulation based experiments conducted in this research project. The chapter first introduced the simulation environment and simulation models used in the experiments, it then showed the complete simulation results. Finally, it presented a comprehensive analysis of the results which verified the efficiency of the proposed algorithms.

In the experiment, the proposed algorithms were tested against some related leading algorithms, and for each tested algorithm, three performance metrics including: the number of forwarding nodes, the average number of signaling messages, and the average signaling message size were evaluated in order to compare the performance of the algorithms. Three transmission ranges were also used in the experiment to generate different network densities. The results established that the proposed algorithms outperform the other ones in all tests. Especially for the LCF algorithm, it had the best performance among all. The results also showed that the LCF algorithm works more efficiently than other tested algorithms in a dense network environment, where it can largely reduce the number of forwarding nodes with only a limited number of signaling message exchanges. It also has a very small signaling message size even in a dense network topology. From the simulation results, it can be seen that the GMPR and the EGMPR algorithms exhibit satisfactory performance even though they do not have bounded approximation ratios. The results also indicate that the self-pruning procedures used in the GMPR and the EGMPR algorithms are effective since they can successfully reduce the CDS size in different network densities without increasing the number of signaling messages or average signaling message size.

Chapter 6

Conclusions and Recommendations for Future Work

6.1 Conclusions

Wireless ad hoc networks (MANETs) have gained much attention in recent years due to their self-organising and infrastructure-free characteristics. Each node in a MANET can act as a router to receive and forward packets, allowing seamless communications between people and devices. Hence, MANETs have great application potential in various scenarios.

Broadcast is an important data transmission method used in MANETs. The goal of a broadcast algorithm is to maximize the node coverage in the network while keeping the computation and communication overheads to a minimum. However, the wireless nature of MANETs makes it difficult to directly adopt broadcast methods used in wired networks. Furthermore, since mobile nodes in certain kinds of MANETs (for example wireless sensor networks) normally are powered

by batteries, reducing energy consumption becomes critical. Therefore, broadcast algorithms designed for MANETs have to be energy efficient in most cases.

This thesis presented an investigation into the existing efficient broadcast algorithms proposed for MANETs. The research focused on the Multipoint Relay (MPR) and connected dominating set (CDS) based algorithms since they are two popular methods that are frequently used in this domain. In the first part of this research, a thorough literature survey was conducted on the MPR and CDS based algorithms. The algorithms were categorized into groups based on their objectives and operational principles. Details of each algorithm were discussed. Drawbacks and possible improvements of the algorithms were also given. Furthermore, in the literature survey, a theoretical analysis was conducted for each algorithm to evaluate their performance, and a group by group comparison of their performance was also presented.

The survey results show that the source dependent property increases the implementation complexity of a broadcast algorithm, because forwarding nodes in a network have to check the sender of each received broadcast packet in order to decide whether they need to retransmit it. It was also indicated in the survey that an effective way to achieve source independence property is to construct a CDS, where nodes in the CDS retransmit all broadcast packets received for the first time. The survey results further pointed out that most of the existing algorithms do not have time and message complexities which increase linearly as the network size grows, and their approximation ratios are not bounded. These aspects may significantly influence the scalability of a broadcast algorithm. Therefore, algorithms that have an unbounded approximation ratio and high computation and communication complexities may not work well in dense or large-scale networks.

In the second part of this thesis, three efficient broadcast algorithms, Gateway Multipoint Relay (GMPR) algorithm, Enhanced Gateway Multipoint Relay (EGMPR) algorithm, and the Low-Cost Flooding (LCF) algorithm were proposed. As the other algorithms published in the research literature, the proposed algorithms do not take rapid changes in the topology, or mobility of the nodes into consideration. The three algorithms achieve broadcasting efficiency by using the CDS construction method in a network. The GMPR and EGMPR algorithms were based on the maximum independent set (MIS) and the MPR algorithm concepts. In the first phase of these two algorithms, an MIS is formed in the network where nodes in the MIS are referred as dominators. Then, the MPR calculation is performed by each dominator to generate MPRs to cover all their two-hop neighbors. Finally, an MPR becomes a connector if its selector has the largest node degree in its one-hop neighborhood. A self-pruning procedure was also proposed in the GMPR algorithm to further reduce the size of the CDS. A dominator in the CDS is eliminated by the self-pruning procedure if there is a connector in its one-hop neighborhood, and the connector can sufficiently cover all one-hop neighbors of this dominator. In the EGMPR algorithm, the self-pruning procedure was further enhanced to achieve higher effectiveness of reducing the size of a CDS. Among the proposed algorithms, the LCF aims to achieve high broadcasting efficiency in dense networks. The LCF algorithm also forms an MIS in the first place, and then it generates connectors to link nodes in the MIS. However, the connectors are selected through a more effective method, which ensures that at most one connector is chosen for a pair of two-hop apart dominators, and at most two connectors are chosen for a pair of three-hop apart dominators.

The algorithms were verified through a series of proofs presented in this thesis. For each proposed algorithm, theoretical analysis was conducted to evaluate

its performance. It was shown that the computational and communication loads imposed by the algorithms on the network element are significantly lower than the leading algorithms published in the research literature. The LCF algorithm in particular, has linear time and message complexities, and its signaling message size and approximation ratio are bounded by constants.

To further evaluate the practical performance of the proposed algorithms, simulation based experiments were conducted. The simulation framework was based on the OMNeT++ simulator and the Mobility Framework (MF) model suite. Modules implementing different broadcast algorithms were designed that ran on top of the models provided by the MF. In the experiment, the proposed algorithms were tested against some selected leading algorithms surveyed in this thesis. For each of the algorithms, three following performance metrics were tested.

- Number of forwarding nodes generated by the algorithm,
- Average number of signaling messages sent by a node, and
- Average signaling message size of the algorithm.

The simulation results showed that the proposed algorithms perform better. Especially for the LCF algorithm, it has the best performance with respect to all metrics tested.

In summary, the contributions of this research can be outlined as follows:

- A thorough survey of the existing MPR and CDS based algorithms, which presents operational details of the algorithms.
- Theoretical analysis of each surveyed algorithm, and comparisons of their operational costs.

- Three new efficient broadcast algorithms that have better performance than the existing ones.
- New modules that implement many existing broadcast algorithms.
- A C++ program that can generate connected random topology for the MF.
- Simulation evaluations of the proposed algorithms and some surveyed algorithms using different transmission ranges and network topologies.

6.2 Recommendations for Future Work

One of the main goals of efficient broadcast algorithms in MANETs is to save energy of nodes. This study does not directly measure the energy cost of each broadcast algorithm. It is believed that further experiments are necessary to compare the energy consumption of different broadcast algorithms. Moreover, the mobility issue is not considered in this study, which may also affect the performance of a broadcast algorithm. Therefore, simulations must be conducted in order to test the algorithms under the mobile environment.

In MANETs, link failures frequently occur, which may be caused by many factors such as exhaustion of a node's power, attenuation of signals or mobility of nodes. A broadcast algorithm for MANETs has to cope with topology changes possibly frequent in order to successfully deliver packets to all nodes in the network. However, most of the existing broadcast algorithms proposed for MANETs are not fault tolerant, that is, a local link failure can trigger a global recalculation of forwarding nodes. Hence, it is recommended that further research on fault tolerant mechanisms should be conducted for the existing broadcast algorithms.

Furthermore, since broadcasting is frequently used by routing protocols in MANETs, it would be of interest to conduct a study on how the proposed broadcast algorithms affect the performance of some standardized routing protocols such as the AODV [12], DSR [13], and OLSR [14] protocols in MANETs.

Appendix A

Architecture of Mobility Framework Model

A.1 Overview of Mobility Framework Model

The Mobility Framework (MF) is a simulation model that intends to support wireless and mobile simulations within OMNeT++ simulator. Details of the OMNeT++ simulator can be referred to [55]. The core framework implements the support for node mobility, dynamic connection management and a wireless channel model. Additionally it provides *basic* modules that can be derived in order to implement own modules. The framework can be used for simulating:

- mobile wireless networks
- distributed (ad-hoc) and centralized networks
- sensor networks
- multichannel wireless networks

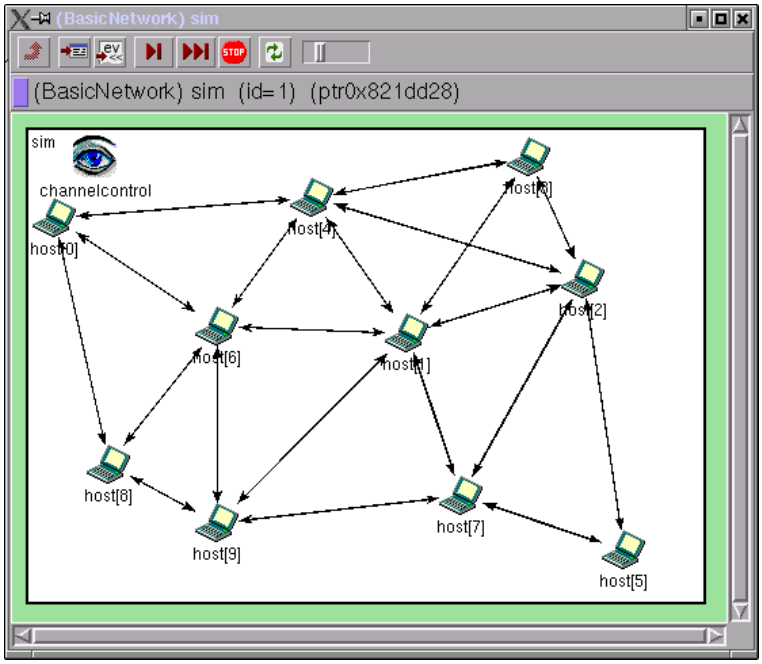


Figure A.1: An example simulation network setup with 10 hosts by using Mobility Framework model.

- many other simulations that need mobility support and / or a wireless interface

Fig. A.1 shows a screen shot of an example of MF simulation network setup with 10 hosts.

The two core components of the MF are: a model of the mobile host in OMNeT++, and the architecture for mobility support and dynamic connection management. This section provides an in-depth tour of the major modules in these two components.

A.1.1 Mobile Host Model

A mobile host model in the MF consists of different modules that implement the functionality similar to the standard ISO/OSI layers. The internal structure of a mobile host model is shown in Fig. A.2. As can be seen, apart from the standard

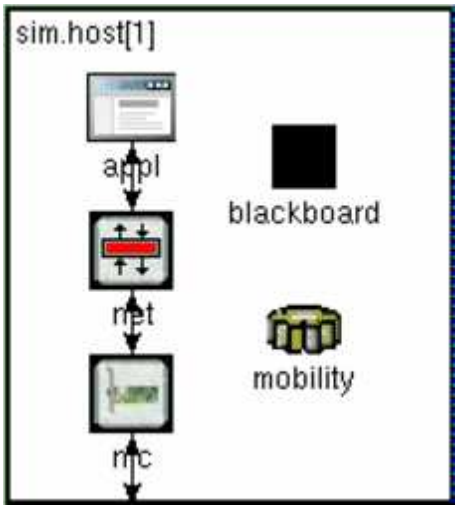


Figure A.2: Structure of a mobile host model in the MF.

ISO/OSI layers there are also a *mobility* module and a module called *blackboard* in the mobile host model. The *mobility* module provides the geographical position of the host and handles its movement, whist the *blackboard* module is used for cross layer communications. It provides an easy way to exchange information between layers.

For the *ic* module, it acts like a wireless network interface card that includes physical layer functions like transmitting, receiving, modulation as well as medium access mechanisms. The *nic* module is divided into two parts as shown in Fig. A.3. The *snreval* and *decider* modules create a physical layer part, and the *mac* module creates the a MAC layer part. The *snreval* module can be used to simulate a transmission delay for a received message, and it calculates the received signal strength. The *decider* module can only process the signal information sent from the lower layer, and it decides whether this message got lost, has bit errors or is correctly received based on some predefined thresholds. Messages sent from the *mac* module bypass it and directly goes to the *snreval* module.

The *net* and *appl* modules process the information sent from the lower layer

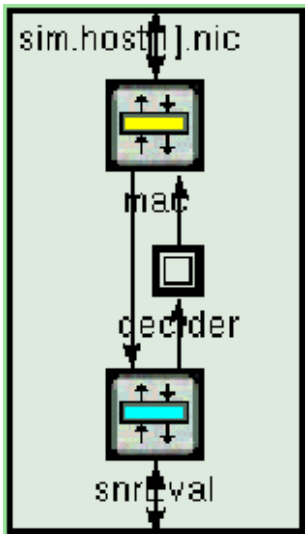


Figure A.3: The structure of the nic module.

and perform calculations of algorithms implemented by protocol designers. Normally, routing protocols are implemented in the *net* module, and the *appl* module deals with application layer protocols. In this simulation study, the efficient broadcast algorithms are created as *appl* modules to run on top of the *net* module, where the *net* module only takes care of checking the sequence number and TTL fields in each packet. A broadcast module calculates forwarding nodes based on the message sent from the lower layer, and it appends IDs of forwarding nodes in the application layer message and sends it down to the *net* module. Then the message goes through each layer in sequence, and is finally sent to the communication channel that connects to the destination node by the *snreval* module.

A.1.2 Mobility Architecture and Dynamic Connection Management

The mobility architecture contains two core modules: a global *channelcontrol* module, and an independent *mobility* module in each host. The *channelcontrol* module is

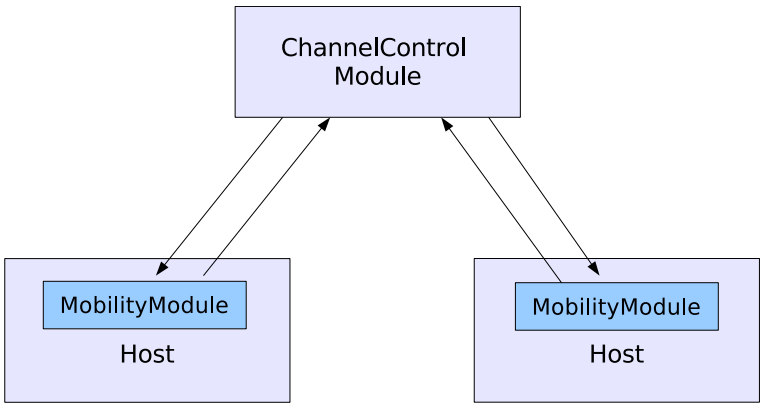


Figure A.4: Mobility architecture.

responsible for establishing communication channels between hosts that are within communication distance and tearing down these connections once they loose connectivity again. The *Mobility* module has two main tasks. The first task is to handle the movements of the host, which can be done using various different mobility models implemented by users. The second task is to inform the location changes of the host to the *channelcontrol* module. Then, the *channelcontrol* module updates all connections for this host. Fig. A.4 shows the relation between the two modules. Based on this architecture, the MF can handle mobility in a distributed manner locally in every host module. Decisions on where to move a host neither require global knowledge nor do they affect other hosts.

Connection management is handled centrally by the *channelcontrol* module. In order to set up and tear down connections, the distances between hosts have to be calculated for which the global knowledge of the positions of all hosts is needed. The *channelcontrol* module determines connections of hosts based the value of the maximum interference distance (MID), which is a conservative bound on the maximal distance at which a host can still possibly disturb the communication of a neighbor, i.e. all hosts further away will not recognize the sending signal at all. The value of the MID is calculated based on the free space propagation model [60]. The

following Equation gives the formula to calculate the MID.

$$MID = \frac{P_{tmax} G_t G r \lambda^2}{(4\pi)^2 P_{rmin} L} \tag{A.1}$$

In the equation, P_{tmax} is the maximum transmission power of a host. P_{rmin} is the minimum receiving power threshold for a packet. G_t and $G_(r)$ are the antenna gains of the transmitter and the receiver respectively, which are set to one in the *channelcontrol* module. λ is the wavelength, and L is the system loss that is also set to one in the module. The value of P_{tmax}, P_{rmin} and λ can be specified by users in the initial state of the simulation. This formula is also used by the *snreval* module to calculate the received signal strength, where the distance value used in the equation is the geographical distance between the sender and itself.

Appendix B

Configuration of Simulation

B.1 OMNeT++ Configuration File

Configuration and global parameters of the OMNeT++ simulator and the Mobility Framework model are set through a configuration file. The default name of the file is *omnetpp.ini*. Detailed information on parameters available for configuration of the simulator environment can be found in [55]. The configuration file for this simulation is shown in Listing B.1.

Listing B.1: The *omnetpp.ini* file used in this study.

```
include runs.ini # define how many runs for a topology
include para.ini # define the total number of nodes in the network
include topo.ini # define geographical position of each host

[General]
ini-warnings = true
network = sim
#random-seed = 13
rng-class = ''cMersenneTwister''
num-rngs = 1
sim-time-limit = 100
debug-on-errors = yes

[Tkenv]
bitmap-path = ''~/oppsim/MF/mobility-fw1_0a6/bitmaps''
```

```
default-run=1
use-mainwindow = yes
print-banners = yes
slowexec-delay = 300ms
update-freq-fast = 10
update-freq-express = 100
breakpoints-enabled = yes

[Cmdenv]
express-mode= yes
event-banners = yes
module-messages = yes
verbose-simulation = yes

[Parameters]

##########################################################
#        Parameters for the entire simulation            #
##########################################################

# network area size
sim.playgroundSizeX = 100
sim.playgroundSizeY = 100

#sim.numHosts = 20 # replaced by the para.ini file

# uncomment to enable debug messages for all modules
**.debug=true
**.coreDebug=true

##########################################################
#        Parameters for the ChannelControl               #
##########################################################

# debug switch
#sim.channelcontrol.coreDegug = true
sim.channelcontrol.carrierFrequency = 2.4e+9

# max transmission power [mW]
#sim.channelcontrol.pMax = 0.252662 # 50m
#sim.channelcontrol.pMax = 0.063165 # 25m
sim.channelcontrol.pMax = 0.022739 # 15 m

# signal attenuation threshold [dBm] // control the transmission
    distance
sim.channelcontrol.sat = -80

# path loss coefficient alpha
sim.channelcontrol.alpha = 2

##########################################################
#        Parameters for the Mobility Module              #
```

```
##########################################################

# debug switch
sim.host[*].mobility.debug = true

# if set to 0 the MN does not move
sim.host[*].mobility.vHost = 0
sim.host[*].mobility.updateInterval = 0.5

# starting position for the hosts "-1" means random staring point
#sim.host[*].mobility.x=-1 # replaced by the topo.ini file
#sim.host[*].mobility.y=-1

##########################################################
#        Parameters for the Host                         #
##########################################################
sim.host[*].color = "cyan"
sim.host[*].appendDisplay = "b=20,20,oval;o=blue,yellow,2"
sim.host[*].defonehopNtimer = 6
sim.host[*].applLayer = "MprApplLayer" # use own application layer

##########################################################
#        Parameters for the Application Layer            #
##########################################################

# debug switch
sim.host[*].appl.debug = true

##########################################################
#        Parameters for the Network Layer                #
##########################################################

# debug switch
sim.host[*].net.debug = true
sim.host[*].net.headerLength=32 # in bits, fixed length
sim.host[*].net.bcMaxEntries = 50 # Max size of the list storing
    sent messages
sim.host[*].net.bcDelTime = 5.2 # timer for deleting entries in
    the list
sim.host[*].net.defaultTtl = 1 # packets can only reach one hop
    distance


##########################################################
#        Parameters for the Mac Layer                    #
##########################################################

# debug switch
sim.host[*].nic.mac.debug = true
sim.host[*].nic.mac.headerLength = 144 #mac + channalID = 3*48
    bits
#sim.host[*].nic.mac.maxQueueSize = 14i #only (for NIC802.11)
```

```
#sim.host[*].nic.mac.bitrate = 2E+6  #in bits/second  (only for
    NIC802.11)
#sim.host[*].nic.mac.rtsCts = false #only for NIC802.11
#sim.host[*].nic.mac.broadcastBackoff = 31 #(only for NIC802.11)

#########################################################
#       Parameters for the Decider                      #
#########################################################

# debug switch
sim.host[*].nic.decider.debug = true
sim.host[*].nic.decider.snirThreshold = -100 #in dB (only for
    NIC802.11)

#########################################################
#       Parameters for the SnrEval                      #
#########################################################

# debug switch
sim.host[*].nic.snrEval.debug = true

sim.host[*].nic.snrEval.headerLength=192 # fix length
sim.host[*].nic.snrEval.bitrate = 2E+6 # in bits/second
#sim.host[*].nic.decider.bitrate = 2E+6 # in bits/second #only for
     NIC802.11

# max transmission power [mW], should be the same as sim.
    channelcontrol.pMax
#sim.host[*].nic.snrEval.transmitterPower = 0.252662#  [mW] # 50
#sim.host[*].nic.snrEval.transmitterPower = 0.063165#  [mW] # 25m
#sim.host[*].nic.snrEval.transmitterPower = 0.022739#  [mW] # 25

#sim.host[*].nic.snrEval.carrierFrequency=2.4E+9
#sim.host[*].nic.snrEval.thermalNoise=-1000000000 # for
    calculating SNR
#sim.host[*].nic.snrEval.sensitivity=-85 # for detecting whether a
#packet transmitting in the channel
#sim.host[*].nic.snrEval.pathLossAlpha=2 # for calculating
    received power
```

The **para.ini** file stores the parameter indicating the total number of nodes in the network, and the **runs.ini** file specifies the number of runs for a particular topology. The **topo.ini** file contains all geographical positions of the nodes in the network. The position is calculated by a C++ program created in this study. All other parameters of different modules in the MF are set in the Parameters part of the configuration file.

B.2 C++ Program to Calculate Positions of Hosts

In order to generate connected random topologies, a C++ program is created in this study. The program reads four arguments which are the transmission range of the node, the seed for generating random values, the total number of nodes in the network, and the network size. It then randomly deploys a node within the transmission range of another node, which is also randomly selected from the list of previously placed nodes. The source code of the program is shown in Listing B.2.

Listing B.2: The C++ program to generate a connected random topology.

```
#####################################################################
# syntax:                                                           #
# ./randomtopology <transmission_range> <seed> <node_number>        #
#  <network_size>                                                   #
#                                                                   #
#####################################################################

#include <cstdlib>
#include <cstdio>
#include <cmath>
#include <iostream>
#include <fstream>
#include <map>

using namespace std;
typedef map<int, struct pos, greater<int> > Map_int_struct;

# the structure storing the position of a host
struct pos
{
  float x;
  float y;
};

int main(int argc, char *argv[])
{
  int transmission_range=0; // transmission range of a node
  int rangesq;
  int seed=1; // default seed
  int node_number=0; // default number of nodes in the network
  int network_size=0; // default network size

  transmission_range=atoi((const char*)argv[1]);
  seed=atoi((const char*)argv[2]);
```

```
node_number=atoi((const char*)argv[3]);
network_size=atoi((const char*)argv[4]);

rangesq=transmission_range*transmission_range;
Map_int_struct nodemap; //  stores the nodes that have been
 placed
Map_int_struct::iterator nodemapmit;
srand48(seed);

pos possion; // assign the 0 node possion
possion.x=network_size/5; // [0,100), first node's X position
possion.y=network_size/5; // [0,100)  first node's Y position
nodemap.insert(make_pair(0,possion)); // insert the 0 node

int prt; // node id of a previous located node
float px;
float py;
float posx=0;
float posy=0;
float w=360.00;
float angle=0.0; // turn angle from the previous point
float ppx=15;

for(int i=1; i<node_number; i++)
{ // start from the second node
  prt=(int)(drand48()*nodemap.size()); //randomly pick a number
  cout<<"prt is: "<<prt<<endl;

  nodemapmit=nodemap.find(prt); //find a previously placed node
 in the map
  px=nodemapmit->second.x; // initial x possition
  py=nodemapmit->second.y; // initial y possition

  //decide distance
  ppx = drand48()*transmission_range; // the diviation of x

  //decide angle
  angle= w * drand48(); // decide an angle
  posy= sinf(angle)*ppx;
  posx= cosf(angle)*ppx;
  posy=py-posy;
  posx=px+posx;

  //don't want a node to be placed too close to the border
  while(posx<10||posx>(network_size-10)||posy<10||posy>(
 network_size-10))
  {
    angle= w * drand48(); // decide an angle
    posy= sinf(angle)*ppx;
    posx= cosf(angle)*ppx;
    posy=py-posy;
    posx=px+posx;
```

```
    }

    //insert positions of hosts
    pos *possion = new pos();
    possion->x=posx;
    possion->y=posy;
    nodemap.insert(make_pair(i, *possion));
    delete possion;
    possion=NULL;

  } // end start from the second node

  //create the topo.ini
  ofstream tfile("topo.ini", ios::out);
  if(!tfile)
  {
    cerr<<"can not create file\n";
  }
  tfile<<"[Parameters]"<<endl;

  for(nodemapmit=nodemap.begin(); nodemapmit!=nodemap.end(); ++
   nodemapmit)
  {
     tfile<<"sim.host["<<nodemapmit->first<<"].mobility.x ="<<
   nodemapmit->second.x<<endl;
     tfile<<"sim.host["<<nodemapmit->first<<"].mobility.y ="<<
   nodemapmit->second.y<<endl;
     tfile<<endl;

  }

  return 0;
  }
```

For each previously placed host, the program stores its geographical position in a list. It then randomly selects a host from the list and places a new host within the transmission range of this host. In order to randomly place a node within the transmission range of a previously deployed node, the program first randomly chooses a value between zero and the node transmission range. This value is treated as the direct distance between the two nodes. Then, the program randomly selects an angle between 0 and 360, and it calculates the X and Y positions of the host based on simple triangular formulas. This program is executed for each run indicated in the **runs.ini** file, and a different seed is assigned to the program for each run in order to

generate a different topology.

Appendix C

Publications

Conference Proceedings

- O. Liang, Y. A. Şekercioğlu, and N. Mani, Gateway multipoint relays – an MPR-based broadcast algorithm for ad hoc networks, in 10th IEEE International Conference on Communications Systems (ICCS'06), Oct. 2006.
- O. Liang, Y. A. Şekercioğlu, and N. Mani, Enhanced gateway multipoint relays for constructing a small connected dominating set in wireless ad hoc networks, in 10th IEEE International Conference on Communications Systems (ICCS'06), Oct. 2006.
- O. Liang, Y. A. Şekercioğlu, and N. Mani, A low-cost flooding algorithm for wireless sensor networks, in IEEE Wireless Communications and Networking Conference (WCNC'07), March 2007.

Journal Papers

- O. Liang, Y. A. Şekercioğlu, and N. Mani, A survey of multipoint relay based broadcast schemes in wireless ad hoc networks, IEEE Communications Surveys & Tutorials, vol. 8, pp. 30-46, Nov. 2006.

- O. Liang, Y. A. Şekercioğlu, and N. Mani, Low-cost flooding: an energy-efficient broadcast algorithm for wireless sensor networks, IEEE Transactions on Mobile Computing, under review.

Bibliography

[1] "IETF mobile ad hoc network working group." [Online]. Available: http://www.ietf.org/html.charters/manet-charter.html

[2] S. Corson and J. Macker, "Mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations," RFC 2501, Jan. 1999. [Online]. Available: http://www.faqs.org/rfcs/rfc2501.html

[3] I. Chlamtac, M. Conti, and J. J. N. Liu, "Mobile ad hoc networking: imperatives and challenges," *Ad Hoc Networks*, vol. 1, pp. 13–64, 2003.

[4] L. Yang, S. Conner, X. Guo, M. Hazra, and J. Zhu, "Common wireless ad hoc network usage scenarios," Internet Draft, Oct. 2003. [Online]. Available: http://www.flarion.com/ans-research/Drafts/draft-irtf-yang-ans-scenarios-00.txt

[5] R. Ramanathan and J. Redi, "A brief overview of ad hoc networks: challenges and directions," *IEEE Communications Magazine*, vol. 50, pp. 20–22, May 2002.

[6] J. A. Freebersyser and B. Leiner, *A DoD perspective on mobile ad hoc networks*, C. E. Perkins, Ed. Reading, MA: Ad Hoc Networking, Addison-Wesley, 2001.

[7] B. Leiner, R. Ruth, and A. R. Sastry, "Goals and challenges of the DARPA GloMo program," *IEEE Personal Communications*, vol. 3, no. 6, pp. 34–43, 1996.

[8] R. Bruno, M. Conti, and E. Gregori, "Mesh networks: commodity multihop ad hoc networks," *IEEE Communications Magazine*, vol. 43, pp. 123–131, Mar. 2005.

[9] H. Alshaer and E. Horlait, "Emerging client-server and ad-hoc approach in inter-vehicle communication platform," in *Proceedings of IEEE Vehicular Technology Conference VTC 2004*, Italy, Sept. 2004, pp. 3955–3959.

[10] J. G. Jetcheva, Y. C. Hu, S. PalChaudhuri, A. K. Saha, and D. B. Johnson, "Design and evaluation of a metropolitan area multitier wireless ad hoc network architecture," in *Proceedings of Fifth Workshop on Mobile Computing Systems and Applications*, Monterey, CA, USA, Oct. 2003, pp. 32–43.

[11] C. R. Dow, P. J. Lin, S. C. Chen, J. H. Lin, and S. F. Hwang, "A study of recent research trends and experimental guidelines in mobile ad-hoc networks," in *Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA'05)*, Taipei, Taiwan, 2005.

[12] C. Perkins and E. Royer, "Ad-hoc on-demand distance vector routing," in *Proceedings of IEEE International Workshop on Mobile Commerce and Services (WMCS'99)*, Feb. 1999, pp. 90–100.

[13] D. Johnson and D. Maltz, "Dynamic source routing in ad-hoc wireless networks," *Mobile Comput.*, 1996, pp. 153–181.

[14] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," RFC 3626, Oct. 2003. [Online]. Available: http://www.faqs.org/rfcs/rfc3626.html

[15] A. Laouiti, A. Qayyum, and L. Viennot, "Multipoint relaying: an efficient technique for flooding in mobile wireless networks," in *35th Annual Hawaii International Conference on System Sciences HICSS'2001*.

[16] C. Ho, K. Obraczka, G. Tsudik, and K. Viswanath, "Flooding for reliable multicast in multi-hop ad hoc networks," in *Proceedings of the International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communication (DIALM)*, 1999, pp. 64–71.

[17] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Proc. 5th Annu. ACM/IEEE Int. Conf. Mobile Computing and Network*, 1999, pp. 151–162.

[18] B. Williams and T. Camp, "Comparison of broadcasting techniques for mobile ad hoc networks," in *Proceedings of Third ACM Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC 2002)*, Switzerland, 2002, pp. 194–205.

[19] J. Wu and F. Dai, "A generic distributed broadcast scheme in ad hoc wireless networks," *IEEE Trans. Computers*, vol. 53, Oct. 2004, pp. 1343–1354.

[20] O. Liang, Y. A. Şekercioğlu, and N. Mani, "A survey of multipoint relay based broadcast schemes in wireless ad hoc networks," *IEEE Communications Surveys & Tutorials*, vol. 8, pp. 30–46, Nov. 2006.

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

[22] E. S.-R. Committee, *Radio equipment and systems: HYPERLAN type 1*, functional specifications ETS 300–652, ETSI, Jun. 1996.

[23] G. Allard, P. Jacquet, and L. Viennot, "Ad hoc routing protocols with multipoint relaying," 5eme Rencontres Francophones sur les aspects Algorithmiques des Telecommunications, (AlgoTel'2003). [Online]. Available: gyroweb.inria.fr/~viennot/postscripts/algotel2003ajv.pdf

[24] I. Joe and S. Batsell, "Mpr-based hybrid routing for mobile ad-hoc networks," in *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN'02)*, Nov. 6–8 2002, pp. 7–12.

[25] M. Garey and D. Johnson, *Computers and intractability, a guide to the theory of NP-completeness*. W.H. Freeman, 1979.

[26] V. Chvátal, *A greedy heuristic for the set-covering problem*. Mathematics of Operation Research, 1979, ch. 4, pp. 233–235.

[27] B. Das and V. Bharghavan, "Routing in ad-hoc networks using minimum connected dominating sets," in *Proceedings of International Conference on Communications (ICC'97)*, vol. 3, 1997, pp. 376–380.

[28] J. Wu and H. L. Li, "On calculating connected dominating set for efficient routing in ad hoc wireless networks," in *Proceedings of 3rd ACM International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 1999, pp. 7–14.

[29] K. Alzoubi, P. J. Wan, and O. Frieder, "Message optimal connected dominating set construction for routing in mobile ad hoc networks," in *Proceedings of Third ACM int'l Symp. Mobile Ad Hoc Networks and Computing (MobiHoc'02)*, 2002.

[30] E. Hansen and S. Zilberstein, "Monitoring and control of anytime algorithms: a dynamic programming approach," *Artificial Intelligence*, no. 126, pp. 139–157, 2001.

[31] B. Mans and N. Shrestha, "Performance evaluation of approximation algorithms for multipoint relay selection," in *Med-Hoc-Net 2004, The Third Annual Mediterranean Ad Hoc Networking Workshop*, Bodrum, Turkey, Jun. 27–30 2004.

[32] N. Shrestha, "Performance evaluation of multipoint relays: Collision and energy efficiency issues," Macquarie University, Tech. Rep., 6 Apr. 2003.

[33] J. Lipman, P. Boustead, and J. Judge, "Utility-based multipoint relay flooding in heterogeneous mobile ad hoc networks," in *Proceedings of the Workshop on the Internet, Telecommunications and Signal Processing (WITSP'02)*, Wollongong, Australia, Dec. 2002, pp. 174–185.

[34] J. Lipman, P. Boustead, J. Chicharo, and J. Judge, "Resource aware information dissemination in ad hoc networks," in *Proceedings of the 11th IEEE International Conference on Networks (ICON 2003)*, Sydney, Australis, Sep. 2003.

[35] H. Badis, A. Munaretto, K. A. Agha, and G. Pujolle, "Optimal path selection in a link state qos routing protocol," in *Proceedings of IEEE Vehicular Technology Conference (VTC2004 spring)*, Italy, May 2004.

[36] A. Munaretto, H. Badis, K. A. Agha, and G. Pujolle, "Qos for ad hoc networking based on multiple metrics: bandwidth and delay," in *Proceedings of IEEE MWCN2003*, Singapore, Oct. 2003.

[37] Y. Ge, T. Kunz, and L. Lamont, "Quality of service routing in ad hoc networks using olsr," in *Proceedings of 36th Hawaii Int. Conf. Syst. Sci.*, 2003.

[38] C. Adjih, P. Jacquet, and L. Viennot, "Computing connected dominated sets with multipoint relays," Technical Report, INRIA, Oct. 2002. [Online]. Available: www.inria.fr/rrrt/rr-4597.html

[39] J. Wu, "An enhanced approach to determine a small forward node set based on multipoint relays," in *IEEE Transactions on Parallel and Distributed Systems*, Sept. 2002, pp. 866–881.

[40] X. Chen and J. Shen, "Reducing connected dominating set size with multipoint relays in ad hoc wireless networks," in *Proceedings of the 7th International Symposium on Parallel Architectures*. Algorithms and Networks, May 2004, pp. 539–543.

[41] J. Wu and W. Lou, "Extended multipoint relays to determine connected dominating sets in manets," in *Proceedings of First IEEE Communication Society Conference on Sensor and Ad Hoc Communication and Networks (SECON)*, 2004.

[42] K. Alzoubi, X. Y. Li, Y. Wang, P. J. Wan, and O. Frieder, "Geometric spanners for wireless ad hoc networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, pp. 408–421, April 2003.

[43] B. Gao, Y. Yang, and H. Ma, "An efficient approximation scheme for minimum connected dominating set in wireless ad hoc networks," in *The 60th IEEE Vehicular Technology Conference 2004 (VTC 2004-Fall)*, vol. 4, Sept. 2004, pp. 2744–2748.

[44] P. J. Wan, K. M. Alzoubi, and O. Frieder, "Distributed construction of connected dominating set in wireless ad hoc networks," in *Proceedings of the IEEE INFOCOM'02 Conference*. New York, USA: IEEE Communications Society, June 2002, pp. 1597–1604.

[45] O. Liang, Y. A. Şekercioğlu, and N. Mani, "Gateway multipoint relays – an MPR-based broadcast algorithm for ad hoc networks," in *10th IEEE International Conference on Communications Systems (ICCS'06)*, Oct. 2006.

[46] ——, "Enhanced gateway multipoint relays for constructing a small connected dominating set in wireless ad hoc networks," in *10th IEEE International Conference on Communications Systems (ICCS'06)*, Oct. 2006.

[47] ——, "A low-cost flooding algorithm for wireless sensor networks," in *IEEE Wireless Communications and Networking Conference (WCNC'07)*, March 2007.

[48] ——, "Low-cost flooding: an energy-efficient broadcast algorithm for wireless sensor networks," *IEEE Transactions on Mobile Computing*, submitted.

[49] E. H. J. Callaway and E. H. Callaway, "Wireless sensor networks: architectures and protocols," in *CRC Press*, Aug. 2003.

[50] M. Tubaishat and S. Madria, "Sensor networks: an overview," *IEEE Potentials*, vol. 22, pp. 20–23, April-May 2003.

[51] I. F. Akyildiz, W. Su, and Y. S. an E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, pp. 102–114, Aug. 2002.

[52] M. V. Marathe, H. Breu, H. B. H. III, S. S. Ravi, and D. J. Rosenkrantz, "Simple heuristics for unit disk graph," *Networks*, vol. 25, pp. 59–68, 1995.

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

[54] A. F. Seila, V. Ceric, and P. Takidamalla, *Applied simulation modelling*. Thomson Learning Inc., 2003.

[55] "OMNeT++ Object-Oriented Discrete Event Simulation System," http://www.omnetpp.org.

[56] B. Stroustrup, *The C++ programming language*, 3rd ed. Addison Wesley, 1997.

[57] "Mobility Framework for OMNeT++ Object-Oriented Discrete Event Simulation System," http://mobility-fw.sourceforge.net/.

[58] D. William and T. Brian, *Principles and practices of interconnection networks*. Elsevier/Morgan Kaufmann, 2004, ch. 2, pp. 473–494.

[59] D. C. Montgomery and G. C. Runger, *Applied statistic and probability for engineers*, 3rd ed. John Wiley & Sons, 2003.

[60] H. T. Friis, "A note on a simple transmission formula," in *Proc. of the IRE*, vol. 41, May 1946, pp. 254–256.