Secure and Efficient Wireless Ad Hoc Networking

by

Majid Khabbazian

B.Sc., Sharif University of Technology, 2002
M.Sc., The University of Victoria, 2004

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

The Faculty of Graduate Studies

(Electrical and Computer Engineering)

The University Of British Columbia

(Vancouver)

August, 2008

Abstract

Wireless ad hoc networks have been emerged to support applications, in which it is required/desired to have wireless communications among a variety of devices without relying on any infrastructure or central managements. In ad hoc networks, wireless devices, simply called nodes, have limited transmission range. Therefore, each node can directly communicate with only those within its transmission range and requires other nodes to act as routers in order to communicate with out-of-range destinations. One of the fundamental operations in ad hoc networks is broadcasting, where a node sends a message to all other nodes in the network. This can be achieved through flooding, in which every node transmits the first copy of the received message. However, flooding can impose a large number of redundant transmissions, which can result in significant waste of constrained resources such as bandwidth and battery power. One of the contributions of this work is to propose efficient broadcast algorithms which can significantly reduce the number of redundant transmissions. We also consider some of the security issues of ad hoc networks. In particular, we carefully analyze the effect of the wormhole attack, which is one of the most severe threats against ad hoc networks. We also propose a countermeasure, which is an improvement over the existing timing-based solutions against the wormhole attack. Finally, in the last chapter, we propose novel point compression techniques which can be used in Elliptic Curve Cryptography (ECC). ECC can provide the same level of security as other public key cryptosystems (such as RSA) with substantially smaller key sizes. Smaller keys can result in smaller system parameters, bandwidth savings, faster implementations and lower power consumption. These advantages make ECC interesting for ad hoc networks with restricted devices.

Table of Contents

Appendices

List of Tables

List of Figures

Acknowledgements

First and foremost, I would like to acknowledge my thesis supervisor, Professor Vijay Bhargava, for his guidance and support.
My respectful gratitude goes to my committee: Professor Ian Blake, Professor Lutz Lampe, Victor Leung, Professor Brian Marcus and Professor Vincent Wong, for giving valuable suggestions, providing constructive criticism, and proofreading the thesis.
My PhD experience was made more enjoyable by my friends and colleagues from the ITS lab. I would like to thank all of them for their friendship, help with research problems and our fun discussions and lab outings.
Last but not least, I would like to thank my parents and my wife for their unconditional love, encouragement and support.

Dedication

To my lovely wife

and our little princess

Statement of Co-Authorship

Majid Khabbazian is the primary author of the papers presented in this work. For all these papers, Mr. Khabbazian identified and proposed the research topic, performed the literature survey and research, analyzed the data, performed the simulations and prepared the manuscripts under the supervision and direction of Professor Vijay Bhargava (and Professor Aaron Gulliver for the last chapter). Mr. Khabbazian was assisted in preparing the papers presented in Chapter 4 and Chapter 5 by Mr. Hugues Mercier and Professor Gulliver, respectively.

All this work has been done completely during Mr. Khabbazian's Ph.D. program. None of the papers presented in this work have been received credit or presented in any other thesis work.

Chapter 1

Introduction and Background

1.1 Introduction and Motivation

In various scenarios, it is required or desirable to have wireless communications between different devices (simply called nodes) without relying on any centralized infrastructure. For example, in battlefields or in emergency/rescue operations, it is often impractical to have an infrastructure and to rely on centralized and organized connectivity. Many of such scenarios can be considered as applications of wireless ad hoc networks.

A wireless ad hoc network can be seen as a collection of self-organizing nodes that can communicate over relatively limited bandwidth wireless links. Each node is equipped with a transceiver with limited transmission range and thus, has to rely on other nodes in order to communicate with destinations outside its transmission range. To establish a communication, the sender first needs to discover a route to the destination node. It then requires all the intermediate nodes on the route to act as a router and forward the packets towards the destination. In ad hoc networks, nodes may move, enter or exit the network. Therefore, the network topology can change and thus an established route (hence the communication) may become broken. In this case, a new route may need to be established through a route discovery phase.

Wireless devices in ad hoc networks have typically limited resources such as battery lifetime, bandwidth and computational power. Therefore, it is crucial to design efficient methods for ad hoc networks in order to save these scarce resources. Also, wireless ad hoc networks are vulnerable to several attacks. In real environments, a malicious node can easily join the network and attack it by simply violating the protocol specifications. Identity spoofing, message tampering, eavesdropping, rushing attack [1] and wormhole attack [2] are a few examples of the attacks that can be employed by malicious nodes. Devising security solution for wireless ad hoc networks is not trivial due to the network specific characteristics such as open environment, dy-

namic topology, lack of centralized infrastructure and limited bandwidth, battery lifetime and computational power. The aim and motivation of this work is to construct novel methods in an attempt to improve the security and efficiency of ad hoc networks.

1.2 Wireless Ad Hoc Networks

1.2.1 Modeling Wireless Ad Hoc Networks

A wireless ad hoc network can be modeled as a directed graph in which each vertex represents a wireless node and there is an edge from vertex v_1 to v_2 if and only if v_2's corresponding node is within the transmission range of v_1's corresponding node. In general, the wireless devices may have different transmission ranges. Moreover, two nodes within transmission ranges of each other may not be able to communicate because of, for example, presence of an obstacle. However, to achieve strong theoretical results, it is widely assumed that nodes have identical transmission range R and that two nodes are connected if their Euclidean distance is not greater than R. Using these assumptions, the network can be modeled by a unit disk graph.

1.2.2 Broadcasting and Routing

Broadcasting and routing are two fundamental operations in ad hoc networks. In broadcasting, a single node sends a message to all other nodes in the network. The simplest way of broadcasting is via flooding. In flooding, each node transmits the received message to all its neighbors (the nodes within its transmission range) if it has not received the message before and drops the message otherwise. Clearly, flooding finally terminates and all nodes receive the message if the network is static and connected and if there is no error or collision in transmissions. However, this is achieved at the cost of one transmission per each single node in the network. In practice, not all the nodes are required to transmit the message in order to deliver it to all the nodes in the network. In particular, the number of redundant transmissions (broadcasts) significantly increases as the average number of neighbors of each node increases. There are many alternative broadcast algorithms proposed to reduce the number of transmissions. In Chapters 2 and 3, we discuss some of the best existing broadcast algorithms and propose

two localized broadcast algorithms.

Routing algorithms are responsible for providing end-to-end communications in ad hoc networks. There are numerous routing protocols proposed for ad hoc networks [3]. These protocols can be divided into two main categories: Proactive (Table-Driven) and Reactive (On-Demand) routing protocols. Proactive routing protocols maintain a constant network view by saving routes to all possible destinations in a few tables and updating them every time the network topology changes. Using proactive routing protocols, nodes can quickly start communication as there is no latency in route discovery. However, they can cause a substantial overhead when the mobility rate or the number of nodes in the network is high. In reactive routing protocols, a route discovery is performed only when a new data communication needs to be established. Consequently, routing overhead is significantly reduced as there is no need to maintain routes, which are not used. Ad Hoc On-Demand Distance Vector Routing (AODV) [4] and Dynamic Source Routing (DSR) [5] are two well-known examples of on-demand routing protocols. Both AODV and DSR protocols consist of two major activities: route discovery and route maintenance. Route discovery phase is initiated when the source node has no valid route to the destination. In this phase, the source node broadcasts a Route Request (RREQ) packet. When the destination node receives the RREQ packet, it replies to the source node with a Route Reply (RREP) packet. Packets in AODV only contain the address of the destination node whereas in DSR they also include the address of all the intermediate nodes.

1.2.3 Wormhole Attack

As mentioned earlier, ad hoc networks are vulnerable to many different attacks. The wormhole attack is one of the most severe attacks that can form a serious threat against ad hoc networks. The wormhole attack is typically launched by two (or more) malicious nodes located far from each other. In this attack, the malicious nodes transfer the received packets to each other through a long "virtual" link between them. The link can be formed using an in-band (tunneling) or out-of-band communications and can attract many packets since it can transfer packets between far apart locations with short delays and/or small number of hops. This can be considered as a useful service from the malicious nodes if they do not take advantage of the virtual link. However, in practice, the malicious nodes can employ this link to analyze the network traffic or to make a denial-of-service attack by, for example,

selectively dropping the packets. Unfortunately, the attackers do not need to compromise any node to launch this attack and the attack cannot be countered with cryptography alone. Moreover, as we will analytically show in Chapter 4, using this attack, the malicious nodes can control/disrupt many communications in the network.

The existing solutions against wormhole attack often try to bound the distance a packet can travel by using location or timing information. For example, in timing-based solutions [2, 6], the sender attempts to bound the packet traveling distance or the distance to the receiver by, for instance, timestamping the packet or through a fast bit exchange between itself and the receiver. There are some other techniques which use different approaches such as using directional antennas [7] or graph topology distortion [8] in order to detect wormhole attacks. In Chapter 4, we briefly explain some of these solutions against the wormhole attack and propose a countermeasure which is an improvement over the existing timing-based methods.

1.3 Elliptic Curve Cryptography

Elliptic curve cryptography (ECC) [9] is widely regarded as the strongest asymmetric cryptosystem for a given key length. In other words, ECC can provide the same level of security as other public key cryptosystems (such as RSA) with substantially smaller key sizes. Smaller keys can result in smaller system parameters, bandwidth savings, faster implementations and lower power consumption. These advantages make elliptic curve cryptography appropriate for ad hoc networks with restricted devices such as mobile phones.

ECC is based on arithmetic of elliptic curves. An elliptic curve E over the field $\mathbb{F}$ is a curve in the so called "long Weirestraß form"

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \tag{1.1}$$

where $a_1, \ldots, a_6 \in \mathbb{F}$. Let $Char(\mathbb{F})$ denote the characteristic of the field $\mathbb{F}$. The above equation can be simplified to $E : y^2 = x^3 + ax + b$ and $E : y^2 + xy = x^3 + ax^2 + b$ when $Char(\mathbb{F}) \neq 2, 3$ and $Char(\mathbb{F}) = 2$, respectively. Clearly, each point on this curve can be represented by (x, y), where x and y are called the x-coordinate and y-coordinate of the point, respectively.

An elliptic curve point can also be represented using its x-coordinate and a an extra bit. Having the x-coordinate of the point, the y-coordinate can

be computed by solving the elliptic curve equation. Since it is a quadratic equation, there will be two possible solutions; thus the extra bit is needed to distinguish the correct solution. This simple technique is called single point compression and is patented by Certicom Corporation. In Chapter 5, we propose a double point and a triple point compression schemes, which allow a compact representation of elliptic curve points without the computational cost associated with the single point compression. The double point and triple point compression schemes can be employed to save about 25% and 33% of bandwidth/memory, respectively.

1.4 Thesis Contribution

In this section, we explain the organization of this work along with our contributions in designing efficient and secure algorithms.

The main objective of this work is to develop efficient algorithms and secure schemes for wireless ad hoc networks. As mentioned earlier, broadcasting is a fundamental primitive in ad hoc networks. In broadcasting, many nodes may get involved in transmitting a message. Consequently, it can significantly affect the network performance. The simplest way of performing a broadcast is via flooding. In flooding, each node transmit the message once. This can cause a huge amount of redundant transmission particularly in dense networks. Unfortunately, it is not possible, in general, to remove all redundant transmissions as the problem of minimizing the total number of required transmissions is NP-Hard when the network is modeled using a unit disk graph. Many broadcast algorithm have been proposed in the literature to reduce the number of redundant transmissions. Among the existing methods, localized broadcast algorithms are more attractive as they can cope with the network topology changes. In Chapters 2 and 3, we propose two efficient localized broadcast algorithms. The first proposed algorithm, presented in Chapter 2, is an improvement over Liu's algorithm [10] which is one of the best broadcast algorithms in its category. In the same chapter, we propose another broadcast algorithm with the objective of reducing the total number of transmissions. We demonstrate that the second proposed algorithm is very successful in achieving this objective for the case when the nodes are randomly distributed. However, we show that, similar to many existing algorithms, the proposed algorithm cannot guarantee a good bound on the total number of transmissions in the worst case.

In Chapter 3, we prove that many existing broadcasting algorithms are not able to guarantee both full delivery and a good bound on the total number of transmissions. Nevertheless, we extend our second proposed algorithm and prove that it can achieve both full delivery and a constant approximation ratio to the minimum number of required transmissions. We also propose some techniques to reduce bandwidth and computational overhead of the algorithm. Moreover, we relax several system model assumptions or replace them with more practical ones.

Chapters 4 and 5 deal with the security of ad hoc networks. In Chapter 4, we analyze the effect of the wormhole attack in shortest-path routing protocols with the objective of drawing a more precise picture of the attack's possible damage. We show that two malicious nodes can control/disrupt a large percentage of the communications in the network if they place the wormhole strategically. We also consider and analyze the case where more than two malicious nodes are involved in the wormhole attack. We then propose a timing-based countermeasure and show that it does not have the shortcomings of the existing timing-based solutions such as the need for having tightly synchronized clocks, predicting predicting the sending time and computing signature while having to timestamp the message with its transmission time. In Chapter 5, we introduce two compression schemes that can be used to reduce the memory and bandwidth requirements in storing and transmitting the elliptic curve points. In practice, single point compression scheme may be employed to reduce the memory/bandwidth requirement. However, single point compression, patented by Certicom Corporation, requires a square root operation which is a computationally expensive operations in some finite fields. In contrast to single point compression scheme, decompression of our schemes does not require any square root operation. Therefore, our schemes can be used, rather than single point compression scheme, when square root operation is computationally expensive. In the same chapter, we also introduce a new approach to reduce the computational overhead of the elliptic curve cryptography at the cost of slightly increasing the bandwidth overhead. This approach can be used to speed up elliptic curve cryptography in any devices, particularly those with constrained computational power or servers overloaded with elliptic curve encryption or key agreement requests. Finally, we conclude this work in Chapter 6.

Bibliography

[1] Y. C. Hu, A. Perrig, and D. B. Johnson, "Rushing attacks and defense in wireless ad hoc network routing protocols," *In Proc. of ACM Workshop on Wireless Security (WiSe)*, pp. 30–40, 2003.

[2] ——, "Packet leashes: A defense against wormhole attacks in wireless ad hoc networks," *In Proc. of IEEE INFOCOM*, 2003.

[3] D. Agrawal and Q. Zeng, *Introduction to Wireless and Mobile Systems.* Brooks/Cole Publishing, 2002.

[4] C. Perkins, "Ad hoc on demand distance vector (AODV) routing (internet-draft)," 1997, internet Draft, draft-ietf-manet-aodv-00.txt.

[5] D. Johnson and D. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, Imielinski and Korth, Eds. Kluwer Academic Publishers, 1996.

[6] S. Capkun, L. Buttyan, and J. P. Hubaux, "SECTOR: Secure tracking of node encounters in multi-hop wireless networks," *In Proc. of the first ACM workshop on Security of ad hoc and sensor networks*, 2003.

[7] L. Hu and D. Evans, "Using directional antennas to prevent wormhole attacks," *In Proc. of Network and Distributed System Security Symposium*, 2004.

[8] R. Maheshwari, J. Gao, and S. R. Das, "Detecting wormhole attacks in wireless networks using connectivity information," *In Proc. of IEEE INFOCOM*, 2007.

[9] N. Koblitz, "Elliptic curve cryptosystems," *Math. Comp.*, vol. 48, pp. 203–209, 1987.

[10] H. Liu, P. Wan, X. Jia, X. Liu, and F. Yao, "Efficient flooding scheme based on 1-hop information in mobile ad hoc networks," *In Proc. of IEEE INFOCOM*, 2006.

Chapter 2

Efficient Broadcasting in Wireless Ad Hoc Networks

2.1 Introduction

Broadcasting is a fundamental communication operation, in which one node sends a message to all other nodes in the network. Broadcasting is widely used as a basic mechanism in many ad hoc network protocols. For example, ad hoc on-demand routing protocols, such as AODV [11] and DSR [12], typically use broadcasting in their route discovery phase. Broadcasting is also used for topology updates, network maintenance or simply for sending a control or warning message. The simplest broadcast algorithm is flooding, in which every node broadcasts the message when it receives it for the first time. Using flooding, each node receives the message from all its neighbors in a collision-free network. Therefore, the broadcast redundancy significantly increases as the average number of neighbors increases. High broadcast redundancy can result in high power and bandwidth consumption in the network. Moreover, it increases packet collisions, which can lead to additional transmissions. This can cause severe network congestion or significant performance degradation, a phenomenon called the broadcast storm problem [13]. Consequently, it is crucial to design efficient broadcast algorithms to reduce the number of required transmissions in the network.

A set of nodes is called a Dominating Set (DS) if any node in the network either belongs to the set or is a 1-hop neighbor of a node in the set. The set of broadcasting nodes forms a Connected Dominating Set (CDS). Therefore, the minimum number of required broadcasts is not less than the size of the minimum CDS. Unfortunately, finding the minimum CDS is NP-hard, even for the unit disk graphs [14, 15]. However, there are some distributed

A version of this chapter has been accepted for publication. M. Khabbazian and V.K. Bhargava, Efficient Broadcasting in Mobile Ad Hoc Networks, *IEEE Transactions on Mobile Computing*, 2008.

algorithms to find a minimum CDS with even constant approximations [16, 17]. These algorithms can be employed to find a small-sized CDS that can be used as a virtual backbone for broadcasting in ad hoc networks. However, this approach is not efficient in networks with frequent topology changes, as maintaining a CDS is often costly [18].

The main objective of efficient broadcast algorithms is to reduce the number of transmissions while keeping the bandwidth and computational overhead as low as possible. One approach to classify broadcast algorithms is based on the neighbor information they use. Some broadcast algorithms such as flooding and probabilistic broadcast algorithms [19, 20] do not rely on neighborhood knowledge. These algorithms cannot typically guarantee 100% delivery and/or effectively reduce the number of transmissions. Moreover, to decide whether or not to broadcast, they may use a threshold (such as probability of broadcast) which may not be easy to find for different network situations. In the second category, broadcast algorithms require to have 2-hop or more neighbor information. The broadcast algorithms in this category can reduce the number of transmissions in the network and guarantee 100% delivery [21, 22, 23]. However, they may induce high overhead in highly dynamic networks as they need to maintain 2-hop network connectivity.

In this chapter, we propose two broadcast algorithms based on 1-hop neighbor information. The first proposed algorithm is a neighbor-designating (or sender-based) algorithm. In neighbor-designating algorithms, the broadcasting nodes select a subset of their neighbors to forward the message. We compare our proposed broadcast algorithm to one of the best neighbor-designating broadcast algorithms that use 1-hop information [18]. In [18], Liu *et al.* proposed a broadcast algorithm that reduces the number of transmissions and achieves local optimality by selecting the minimum number of forwarding nodes with minimum time complexity $O(n \log n)$, where n is the number of neighbors. We show that this optimality only holds for a subclass of neighbor-designating broadcast algorithms employing 1-hop information, and prove that our proposed neighbor-designating algorithm can achieve 100% delivery with time complexity $O(n)$. Moreover, Liu's algorithm selects n forwarding nodes in the worst case, while our proposed algorithm selects 11 nodes in the worst case. Based on our simulation results, our neighbor-designating algorithm results in fewer transmissions than does Liu's algorithm. All these nice properties are achieved at the cost of slightly increase in end-to-end delay. Therefore, our first proposed algorithm is preferred to Liu's algorithm when the value of n is typically large and it is

important to bound the packet size.

We also propose a self-pruning (also called receiver-based) broadcast algorithm in this chapter. In self-pruning algorithms, the receiver decides whether or not to broadcast the message. Our proposed self-pruning algorithm is a novel broadcast algorithm that can significantly reduce the number of transmissions in the network. We show that using our proposed self-pruning algorithm, two close neighbors are not likely to broadcast the same message. In other words, we prove that the probability of broadcast for a node N_A exponentially decreases when the distance between N_A and its broadcasting neighbor decreases, or when the density of nodes increases. Based on our experimental results, the number of broadcasts using our self-pruning algorithm is less than one of the best-known approximations for the minimum number of required broadcasts.

The rest of this chapter is organized as follows. In Section 2.2, we describe the system model and network assumptions. In Section 2.3, we discuss our proposed neighbor-designating broadcast algorithm and its characteristics. We propose a simple and highly efficient self-pruning broadcast algorithm in Section 2.4 and prove an interesting property of the algorithm. In Section 2.5, we verify the theoretical results using simulation and compare the number of forwarding nodes of our proposed broadcast algorithms with that of one of the best existing broadcast algorithms and an approximated lower bound of optimal solution. Finally, we provide a summary in Section 2.6.

2.2 System Model

Our system model is very similar to that used by Liu *et al.* [18]. We assume that all nodes are located in a 2-D plane and have a transmission range of R. Therefore, the topology of the network can be represented by a unit disk graph. We assume that the network is connected. Two nodes are considered neighbors if they are in transmission range of each other. We suppose that each node knows its location via a localization technique such as Global Positioning System (GPS) or the lightweight techniques summarized in [24]. Each node periodically broadcasts a very short *Hello* message, which includes its *id* and position. Thus, each node gets the position of its neighbors as well. In the Medium Access Control (MAC) layer, we assume that scheduling is done according to the p-persistent CSMA/CA protocol, which IEEE 802.11 is based on in the broadcast mode. In the p-persistent CSMA/CA protocol,

when a node has a message to transmit, it initiates a defer timer by a random number and starts listening to the channel. If the channel is busy, it continues to listen until the channel becomes idle. When the channel is idle, it starts decrementing the defer timer at the end of each time unit. The message is broadcast when the timer expires.

2.3 An Efficient Neighbor-Designating Broadcast Algorithm

2.3.1 Algorithm Structure

Our first proposed broadcast algorithm is a neighbor-designating algorithm, i.e., each sender selects a subset of nodes to forward the message. Each message can be identified by its source *id* and a sequence number incremented for each message at the source node. Algorithm 2.1 is a general neighbor-designating broadcast algorithm and indicates the structure of our proposed neighbor-designating broadcast algorithm. Upon expiration of the timer, the algorithm requests the MAC layer to schedule a broadcast. The message scheduled in the MAC layer is buffered and then broadcast with a probability p. This adds another delay (i.e., the MAC layer delay) in broadcasting the message. The MAC layer delay in IEEE 802.11 is a function of several factors including the network traffic. Note that there is a chance that a node changes its decision (regarding the selected nodes or regarding whether to broadcast) during the MAC layer delay due to receiving other copies of the message. This chance is not negligible when the delay in the MAC layer is comparable to the average value of the timer set in the broadcast algorithm. As stated in [25], one solution to this problem is a cross-layer design in which the network layer is given the ability to modify or remove packets that are present in the MAC layer queue. This solution allows the broadcast algorithms to perform close to their ideal performance even for very small average timer values [25]. In Chapters 2 and 3, we assume that either the MAC layer delay is negligible compared to the average delay set by the algorithm, or the network layer (hence the algorithm) is able to modify or remove packets buffered in the MAC layer queue.

The neighbor-designating broadcast algorithms can be divided into two subclasses. In the first subclass, each node decides whether or not to broadcast solely based on the first received message and drops the rest of the same

messages that it receives later. Liu's algorithm falls in this subclass. In Liu's algorithm, each node selects the smallest subset of its 1-hop neighbors with the maximum coverage area, where the coverage area of a set of nodes is the union of their transmission coverage. Clearly, if the selected neighbors broadcast the packet the transmission of non-selected neighbors would be redundant, hence they are not required to broadcast the packet. Liu *et al.* proved that their selection algorithm has the optimal time complexity $O(n \log n)$.

In the second subclass of neighbor-designating broadcast algorithms, each node can decide whether or not to broadcast after each message reception. However, if a node broadcasts a message, it will drop the rest of the same messages that it receives in future. Therefore, each message is broadcast at most once by a node using the broadcast algorithms in both subclasses. Our first proposed broadcast algorithm falls in this subclass of neighbor-designating broadcast algorithms. We show that the proposed algorithm can reduce both the computational time complexity of selecting the forwarding nodes and the maximum number of selected nodes in the worst case.

Algorithm 2.1 shows the basic structure of our proposed neighbor-designating broadcast algorithm. As shown in Algorithm 2.1, each node schedules a broadcast for a received message if the node is selected by the sender and if it has not broadcast the same message before. Clearly, each message is broadcast at most once by a node, which is similar to Liu's algorithm. However, in Liu's algorithm, each node may only schedule a broadcast when it receives a message for the first time. In contrast, in Algorithm 2.1, a broadcast schedule can be set at any time. For example, a message can be dropped after the first reception but scheduled for broadcast the second time. Clearly, the main design issue in Algorithm 2.1 is how to select the forwarding nodes.

2.3.2 Forwarding-Node Selection Algorithm

Let us consider point A as the node N_A and a circle $C_{A,R}$ centered at A with a radius R as the transmission range of N_A. We use $\overline{AB}$ to denote the distance between two points A and B. Before delving into the algorithm description and proofs, we need to define the following terms:

Definition 2.1 (Bulged Slice). *As illustrated in Figure 2.1, we define a bulged slice around A as the intersection area of three circles with radius R and centers A, M and N, where $\overline{AM} = R$, $\overline{AN} = R$, $\overline{MN} = R$. Note that*

Algorithm 2.1 A general neighbor-designating algorithm

```
1: Extract the required information from the received message M
2: if M has been broadcast before or does not contain node's id then
3:    drop the message
4: else
5:    set a defer timer
6: end if
7: When defer timer expires
8:   Select a subset of neighbors to forward the message
9:   Attach the list of forwarding node to the message
10:  Schedule a broadcast
```

in any bulged slice AMN we have $\angle MAN = \frac{\pi}{3}$.

Definition 2.2 (Right/Left Bulged Slice). *As shown in Figure 2.2, let A and B be two points such that $0 < \overline{AB} \leqslant R$, and AMN be a bulged slice around A. Suppose that the point B is on one of the arcs $\widehat{AM}$ or $\widehat{AN}$ of the bulged slice AMN. In this case, AMN is called the right bulged slice of B around A if it contains the $\frac{\pi}{3}$ clockwise rotation of point B around A and is called its left bulged slice around A, otherwise.*

Definition 2.3 (Bulged Angle). *Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be two bulged slices around A. The bulged angle $\angle_A(\mathcal{B}_1, \mathcal{B}_2)$ is defined to be equal to $0 \leqslant \alpha < 2\pi$ if $\mathcal{B}_2$ is a α anti-clockwise rotation of $\mathcal{B}_1$ around A.*

Definition 2.4 (B-coverage Set). *A subset of neighbors of N_A is called a B-coverage set of N_A if any non-empty bulged slice around A contains at least one node from the set. A bulged slice is empty if there is no node inside it.*

Definition 2.5 (Slice-Based Selection Algorithm). *A forwarding-node selection algorithm is called a slice-based selection algorithm (or slice-based algorithm) if for any node N_A, it selects a B-coverage set of it.*

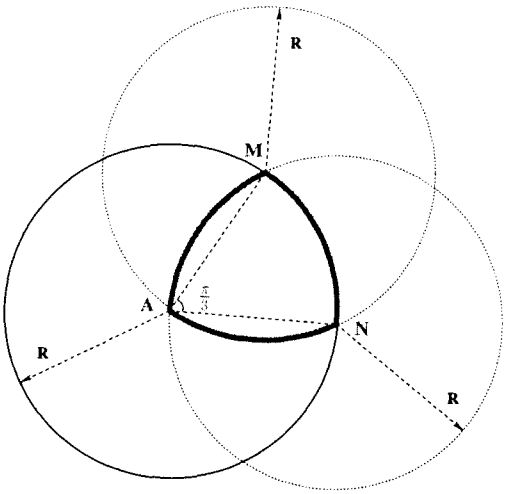

Figure 2.1: A bulged slice around A.

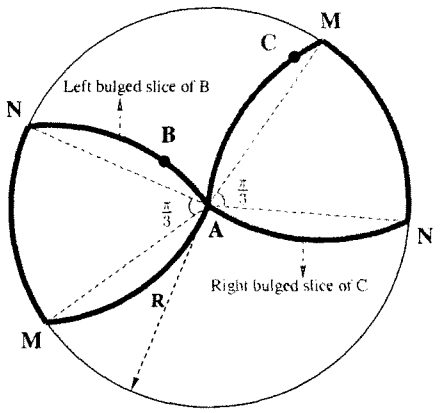

Figure 2.2: Left bulged slice of B and right bulged slice of C around A.

A node can have several different B-coverage sets. Therefore, there are more than one slice-based selection algorithms. For example, a trivial slice-based selection algorithm would be one that selects all of the neighbors as the B-coverage set. Clearly, this algorithm will result in flooding if it is used as the forwarding-node selection scheme in Algorithm 2.1. In this section, we first show that Algorithm 2.1 can achieve 100% delivery if it uses any slice-based algorithm to select the forwarding nodes. We then present an efficient slice-based algorithm that selects 11 nodes in the worst case and has computational complexity $O(n)$, where n is the number of neighbors.

Lemma 2.1. *For any two points P_1 and P_2 inside a bulged slice, we have*

$$\overline{P_1P_2} \leqslant R.$$

Proof. As shown in Figure 2.3, the line passing through P_1 and P_2 intersects the bulged slice AMN at P_1' and P_2'. Clearly, $\overline{P_1P_2} \leqslant \overline{P_1'P_2'}$. Therefore, to prove the lemma it is sufficient to show that $\overline{P_1'P_2'} \leqslant R$. This is easy to show if both P_1' and P_2' are on the same arc of the bulged slice. Thus, without loss of generality, we can assume that P_1' and P_2' are on the arcs $\widehat{AM}$ and $\widehat{AN}$, respectively. Let us consider the perpendicular bisector of the line segment $P_1'M$ (Line L). Line L passes through N because $\overline{NM} = \overline{NP_1'} = R$. Since the point P_2' is on the arc $\widehat{AN}$, the line segment MP_2' will cross the line L at a point Q. Using triangle inequality, we have

$$\overline{P_1'P_2'} \leqslant (\overline{QP_2'} + \overline{QP_1'}) = (\overline{QP_2'} + \overline{QM}) = P_2'M = R.$$

Note that Q is on the line L, hence $\overline{QP_1'} = \overline{QM}$. □

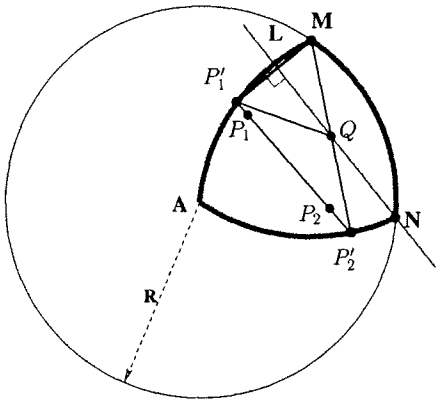

Figure 2.3: Upper bound on the distance between nodes inside a bulged slice.

Consider two points A and B such that $R < \overline{AB} \leqslant 2R$. As shown in Figure 2.4, the line segment AB intersects the circle $C_{A,R}$ at point Q. Let AQM and AQN be the left and right bulged slices of Q around A, respectively. The following lemmas hold:

Lemma 2.2. *A point P is inside the bulged slice AQM or AQN if $\overline{AP} \leqslant R$ and $\overline{BP} \leqslant R$.*

Proof. It is easy to show that for any triangle $\triangle ABC$, $\overline{AM} \leqslant \overline{AB}$ or $\overline{AM} \leqslant \overline{AC}$, where M is a point on the line segment BC. Consequently, in the triangle $\triangle PAB$ (shown in Figure 2.4) we have

$$\overline{PQ} \leqslant \overline{AP} \leqslant R \quad \text{or} \quad \overline{PQ} \leqslant \overline{BP} \leqslant R.$$

Therefore, $\overline{PQ} \leqslant R$. Thus based on the bulged slice definition, the point P is inside the bulged slice AQM or AQN. □

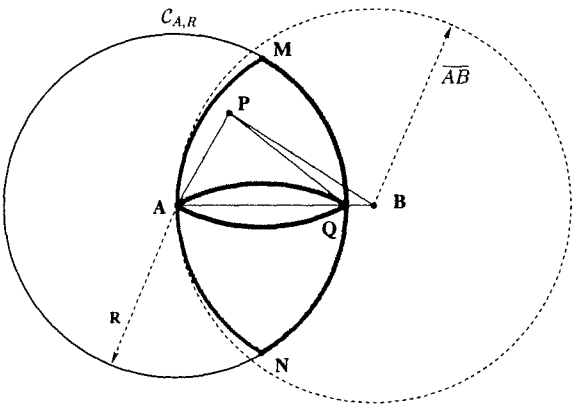

Figure 2.4: Illustration of Lemmas 2.2 and 2.3.

Lemma 2.3. *For any point $P \neq A$ inside the bulged slice AQM or AQN, we have*

$$\overline{BP} < \overline{BA}.$$

Proof. Using triangle inequality we get

$$\overline{BP} \leqslant \overline{BQ} + \overline{QP} \leqslant \overline{BQ} + R = \overline{BA}.$$

The equality holds only when $\overline{BP} = \overline{BQ} + \overline{QP}$ and $\overline{QP} = R$, or simply when $P = A$.

□

Theorem 2.1. *In a collision-free network, Algorithm 2.1 can achieve* 100% *delivery if it uses a slice-based selection algorithm to select the forwarding nodes.*

Proof. Using Algorithm 2.1, each node broadcasts the message at most once. Therefore, broadcasting will eventually terminate. By contradiction, suppose there is at least one node that has not received the message after the broadcasting termination. Let us consider the following set:

$$\begin{aligned}\Lambda = \{(N_X, N_Y, N_Z) | & N_X \text{ has broadcast the message,} \\ & N_Z \text{ has not received the message and} \\ & N_Y \text{ is the neighbor of both } N_X \text{ and } N_Z\}.\end{aligned}$$

Suppose N_S is the node that initiated broadcasting and N_T is a node that has not received the message. The network is connected, thus there is a path between N_S and N_T. Clearly, we can find two neighbor nodes N_C, N_B along the path from N_T to N_S such that N_C has not received the message, while N_B has received it. Suppose that N_B has received the message from N_A. Consequently, $(N_A, N_B, N_C) \in \Lambda$, thus $\Lambda \neq \emptyset$. As a result,

$$\exists (N_{A'}, N_{B'}, N_{C'}) \in \Lambda \text{ s.t. } \forall (N_X, N_Y, N_Z) \in \Lambda : \overline{A'C'} \leqslant \overline{XZ}. \tag{2.1}$$

Obviously, $N_{A'}$ and $N_{C'}$ are not neighbors, because $N_{C'}$ has not received the message. Thus, $\overline{A'C'} > R$. Using Lemma 2.2, B' is inside the bulged slice $A'P_1P_2$ or $A'P_1P_3$, where P_1 is the intersection of line segment $A'C'$ and the circle $C_{A',R}$ and $A'P_1P_2$ and $A'P_1P_3$ are the left and the right bulged slices of P_1 around A', respectively. Without loss of generality, assume that B' is inside the bulged slice $A'P_1P_2$. Since $N_{A'}$ has at least one neighbor (i.e., $N_{B'}$) in this slice, there must be a selected node N_D in the slice that has forwarded the message. Using Lemma 2.1, we get $\overline{B'D} \leqslant R$; hence, nodes N_D and $N_{B'}$ are neighbors. Therefore, $(N_D, N_{B'}, N_{C'}) \in \Lambda$. However, this contradicts (2.1) because using Lemma 2.3 we have

$$\overline{DC'} < \overline{A'C'}.$$

□

Algorithm 2.2 shows our proposed slice-based selection algorithm. Suppose that node N_A uses the proposed algorithm to select the forwarding nodes from its neighbors. Let us assume that N_A stores all of its neighbors' *id*s and locations in an array of length n, where n is the number of neighbors. The algorithm selects the first node N_{S_1} randomly from the array. The first node can also be selected deterministically by, for example, selecting the node

that is the farthest away from N_A. Let $\mathcal{LB}_A(P)$ and $\mathcal{RB}_A(P)$ denote the left bulged slice and right bulged slice of P around A, respectively. Suppose that N_{S_i} is the last node selected by the algorithm. To select the next node, the algorithm iterates through the array and selects the node $N_{S_{i+1}}$ such that it is inside the slice $\mathcal{LB}_A(S_i)$, $\angle_A(\mathcal{LB}_A(S_i), \mathcal{LB}_A(S_{i+1})) \neq 0$ and

$$\begin{aligned} &\forall N_B \text{ inside } \mathcal{LB}_A(S_i) : \\ &\angle_A(\mathcal{LB}_A(S_i), \mathcal{LB}_A(B)) \leqslant \angle_A(\mathcal{LB}_A(S_i), \mathcal{LB}_A(S_{i+1})). \end{aligned} \tag{2.2}$$

If there is no such node, the algorithm selects $N_{S_{i+1}}$ such that

$$\begin{aligned} &\forall N_B \text{ inside } \mathcal{C}_{A,R} : \\ &\angle_A(\mathcal{LB}_A(S_i), \mathcal{RB}_A(S_{i+1})) \leqslant \angle_A(\mathcal{LB}_A(S_i), \mathcal{RB}_A(B)). \end{aligned} \tag{2.3}$$

The algorithm terminates by selecting the last node N_{S_m} if N_{S_m} is inside $\mathcal{LB}_A(S_1)$ or N_{S_1} is inside $\mathcal{LB}_A(S_m)$ or $S_{m+1} = S_1$.

Algorithm 2.2 A slice-based selection algorithm

Input: $List_A[1 \ldots n]$: List of all neighbors of N_A
Output: A B-coverage set of N_A: $\{N_{S_i}\}$

1: $ind \leftarrow 1$
2: $i \leftarrow 0$
3: **repeat**
4: $ang\_max \leftarrow 0$
5: $ang\_min \leftarrow 2\pi$
6: $i \leftarrow i + 1$
7: $N_{S_i} \leftarrow List_A[ind]$
8: $chk \leftarrow false$
9: **for** $j = 1$; $j \leqslant$ length$(List_A)$; j++ **do**
10: **if** $List_A[j]$ is in $\mathcal{LB}_A(S_i)$ **then**
11: **if** $\angle_A(\mathcal{LB}_A(S_i), \mathcal{LB}_A(List_A[j])) > ang\_max$ **then**
12: $chk \leftarrow true$
13: $ind\_max \leftarrow j$
14: $ang\_max \leftarrow \angle_A(\mathcal{LB}_A(S_i), \mathcal{LB}_A(List_A[j]))$
15: **end if**
16: **else**
17: **if** $\angle_A(\mathcal{LB}_A(S_i), \mathcal{RB}_A(List_A[j])) < ang\_min$ **then**
18: $ind\_min \leftarrow j$
19: $ang\_min \leftarrow \angle_A(\mathcal{LB}_A(S_i), \mathcal{RB}_A(List_A[j]))$
20: **end if**
21: **end if**
22: **end for**
23: **if** chk **then**
24: $ind \leftarrow ind\_max$
25: **else**
26: $ind \leftarrow ind\_min$
27: **end if**
28: **until** S_1 is in $\mathcal{LB}_A(List_A[ind])$ OR $List_A[ind]$ is in $\mathcal{LB}_A(S_1)$
29: **if** $ind \neq 1$ **then**
30: $N_{S_{i+1}} \leftarrow List_A[ind]$
31: **end if**

Lemma 2.4. *Suppose the proposed algorithm selects m nodes* $\{N_{S_1}, N_{S_2}, ..., N_{S_m}\}$. *For any* $1 \leqslant i < m-2$, *we have*

$$\angle_A(\mathcal{LB}_A(S_i), \mathcal{LB}_A(S_{i+2})) > \frac{\pi}{3}.$$

Proof. Based on (2.2), (2.3) and the algorithm termination condition, we can show that

$$\angle_A(\mathcal{LB}_A(S_i), \mathcal{LB}_A(S_{i+2})) > \angle_A(\mathcal{LB}_A(S_i), \mathcal{LB}_A(S_{i+1}))$$

for any $1 \leqslant i < m-2$. By contradiction, assume that

$$\angle_A(\mathcal{LB}_A(S_i), \mathcal{LB}_A(S_{i+2})) \leqslant \frac{\pi}{3}.$$

Therefore, S_{i+2} is inside $\mathcal{LB}_A(S_i)$. Thus, using (2.2) we have

$$\angle_A(\mathcal{LB}_A(S_i), \mathcal{LB}_A(S_{i+2})) \leqslant \angle_A(\mathcal{LB}_A(S_i), \mathcal{LB}_A(S_{i+1}))$$

which is a contradiction. □

Theorem 2.2. *The proposed slice-based selection algorithm will select at most 11 nodes.*

Proof. By contradiction, assume that the algorithm selects more than 11 nodes. Therefore, S_{11} is not in $\mathcal{LB}_A(S_1)$. Using Lemma 2.4, we get

$$\angle_A(\mathcal{LB}_A(S_1), \mathcal{LB}_A(S_{11})) = \sum_{i=1}^{5}(\angle_A(\mathcal{LB}_A(S_{2i-1}), \mathcal{LB}_A(S_{2i+1}))) > 5 \times \frac{\pi}{3}.$$

Therefore, $\angle_A(\mathcal{LB}_A(S_{11}), \mathcal{LB}_A(S_1)) < (2\pi - 5 \times \frac{\pi}{3}) = \frac{\pi}{3}$. Consequently, S_1 is inside $\mathcal{LB}_A(S_{11})$, thus the proposed slice-based algorithm will terminate after selecting S_{11}. □

The above theorem gives an upper bound on the number of nodes selected by the proposed selection algorithm. In Section 2.5, using simulation we show that the average number of selected nodes (when the nodes are distributed uniformly) is less than 6.

Theorem 2.3. *Time complexity of the proposed slice-based selection algorithm is* $O(n)$, *where n is the number of neighbors.*

Proof. The algorithm selects the first node in $O(1)$. To select each of the other nodes, the algorithm performs $O(n)$ operations by checking all the neighbors in the array. Therefore, the complexity of the algorithm is $O(m \times n)$, where m is the number of selected nodes. Using Theorem (2.2) we have $m \leqslant 11$, thus the time complexity of algorithm is $O(n)$. □

2.3.3 Reducing the Number of Forwarding Nodes

In the sender-based broadcasting algorithms, each broadcasting node attaches a list of its selected forwarding nodes to the message before broadcasting it. This procedure will increase the bandwidth and power required to broadcast the message. As shown earlier, our proposed slice-based selection algorithm reduces the number of selected forwarding nodes to 11 in the worst case. In this section, we show how to further reduce the number of selected nodes.

Recall that the proposed slice-base algorithm selects a subset of N_A's neighbors such that there is at least one selected node in any non-empty bulged slice around A. Suppose N_A extracts the list of the forwarding nodes from each message it receives. Let $\mathcal{L}_A$ be a subset of N_A's neighbors that has broadcast the message or been selected by other nodes to forward it. Since all of the selected forwarding nodes are required to broadcast the message, it is sufficient for N_A to find a subset of its neighbors $\mathcal{S}_A$ such that any non-empty bulged slice around A contains at least one node from $\mathcal{S}_A \cup \mathcal{L}_A$. Algorithm 2.2 can be simply extended to achieve this in $O(n)$. Note that the extended algorithm can start with a node from $\mathcal{L}_A$ and select any node in $\mathcal{L}_A$ as soon as it appears in the left bulged slice of the previously selected node. Finally, the extended algorithm removes all of the nodes in $\mathcal{L}_A$ from the set of selected nodes.

2.3.4 Maximizing the Minimum Node Weight of B-coverage Set

Suppose node N_A assigns a weight to each of its neighbors. The weight can represent the neighbor's battery lifetime, its distance to N_A, the average delay of the node, the level of trust or a combination of them. In some scenarios, we may desire to find a B-coverage set such that its minimum node weight is maximum or its maximum node weight is minimum among that of all B-coverage sets. For example, assume that the weight of each node represents its battery lifetime in a wireless network. It may be desirable to select the nodes with a higher battery lifetime to forward the message in order to keep the nodes with a lower battery lifetime alive. Algorithm 2.3 shows how to find a B-coverage set such that its minimum node weight is maximum among that of all B-coverage sets. A similar approach can be used to find a B-coverage set such that its maximum node weight is minimum.

Algorithm 2.3 Maximizing the minimum node weight

Input: $List_A[1 \ldots n]$: List of all neighbors of N_A
Output: A B-coverage set of N_A with highest minimum node weight

1: $SList_A \leftarrow sort(List_A)$ {Sort the neighbor nodes by their weights} $\{SList[i] \geqslant SList[j] \Leftrightarrow i \leqslant j\}$
2: $H \leftarrow n; \quad T \leftarrow 1; \quad m \leftarrow \lfloor \frac{n}{2} \rceil$
3: $St \leftarrow Algorithm\_2.2(SList[1])$
4: **if** St is a B-coverage set for N_A **then**
5: **return** SList[1]
6: **end if**
7: **while** $H > T + 1$ **do**
8: $St \leftarrow Algorithm\_2.2(SList[1 \ldots m])$ {Pass m nodes with the highest weights to Algorithm 2.2 as the input}
9: **if** St is a B-coverage set for N_A **then**
10: $H \leftarrow m$
11: $m \leftarrow \lceil \frac{T+m}{2} \rceil$
12: **else**
13: $T \leftarrow m$
14: $m \leftarrow \lceil \frac{H+m}{2} \rceil$
15: **end if**
16: **end while**
17: **return** (Algorithm_2.2($SList[1 \ldots H]$))

Algorithm 2.3 first sorts the nodes by their weights in decreasing order. Then, in each step it passes m nodes with the highest weights to Algorithm 2.2 as input and gets a set of (at most 11) nodes as output, where $1 \leqslant m \leqslant n$ is an integer initially set to $\lceil \frac{n}{2} \rceil$. If the output set is a B-coverage set, Algorithm 2.3 sets H to m and decreases m to $\lceil \frac{T+m}{2} \rceil$, where T and H are variables initially set to 1 and n, respectively. Otherwise, it sets T to m and increases m to $\lceil \frac{H+m}{2} \rceil$. After a finite number of steps we get $H = T + 1$. Algorithm 2.3 then returns the output of Algorithm_2.2($SList[1 \ldots H]$).

Corollary 2.1. *Algorithm 2.3 will select at most 11 nodes.*

Proof. The proof is clear, as Algorithm 2.3 returns an output of Algorithm 2.2 (Line 17). □

Theorem 2.4. *Time complexity of Algorithm 2.3 is $O(n \log n)$.*

Proof. Algorithm 2.3 requires $O(n \log n)$ operations to sort the list of neighbors $List_A[1 \ldots n]$. The computational complexity of Algorithm 2.2 is $O(n)$. Therefore, Algorithm 2.3 performs $O(n)$ operations in each iteration of the while loop. The while loop terminates after $O(\log n)$ iterations because it uses a binary search approach to find the minimum value of H. Consequently, the order of Algorithm 2.3 is $O(n \log n + \log n \times n)$. □

Theorem 2.5. *Minimum weight of nodes of the B-coverage set selected by Algorithm 2.3 is maximum among that of all B-coverage sets.*

Proof. Suppose that St_{min} is a B-coverage set such that the minimum weight of nodes in St_{min} is greater than or equal to that of other B-coverage sets. Let $N_X \in St_{min}$ be the node with the minimum weight in St_{min}. Assume N_A has K neighbors with weights greater than or equal to the weight of N_X. Therefore, the output of Algorithm_2.2($SList[1 \ldots K]$) is a B-coverage set. Note that Algorithm 2.3 finds the minimum H such that the output of Algorithm_2.2($SList[1 \ldots H]$) is a B-coverage set for N_A. Therefore, $H \leqslant K$ and thus the minimum weight of nodes of the B-coverage set selected by Algorithm 2.3 is greater than or equal to the weight of N_X. □

2.3.5 Similarity With A Topology Control Algorithm

In [26] and [27], the authors proposed a cone-based topology control algorithm, where each node makes local decisions about its transmission power.

The objective of the algorithm is to minimize the transmission power of each node without violating the network connectivity. In order to do that, each node N_A transmits with the minimum power P_α, such that in every non-empty cone of degree α around N_A, there is some node that N_A can reach with power P_α. A cone is non-empty, if there is at least a node in the cone that N_A can reach using its maximum power. For $\alpha = \frac{2\pi}{3}$, they proved that the network remains connected if the cone-based algorithm is employed.

Suppose that we use cones instead of bulged slices in the proposed forwarding-node selection algorithm. Therefore, the algorithm will select the forwarding node set such that any non-empty cone of degree α around N_A contains at least one node from the forwarding node set. Surprisingly, this algorithm will not guarantee full delivery. Figure 2.5 shows a counterexample for the case where $\alpha = \frac{2\pi}{3}$. Figure 2.6 shows that even for $\alpha = \frac{\pi}{3}$, full delivery cannot be guaranteed. In both Figures 2.5 and 2.6, the node N_A initiates broadcasting and selects only N_B and N_C to forward the message. Suppose that N_D is close enough to the point M such that it is the only node that can reach N_E. In this case, N_E will not receive the message because N_D is not selected by neither N_B nor N_C to forward the message. Note that the cone-based and the forwarding-node selection algorithms use different approaches. In the cone-based algorithm, a node N_A *increases its power* from zero until there is a node in each non-empty cone around N_A. However, in the forwarding-node selection algorithm, a node N_A *selects* some nodes (the forwarding nodes) until there is a selected node in each non-empty bulged slice around N_A.

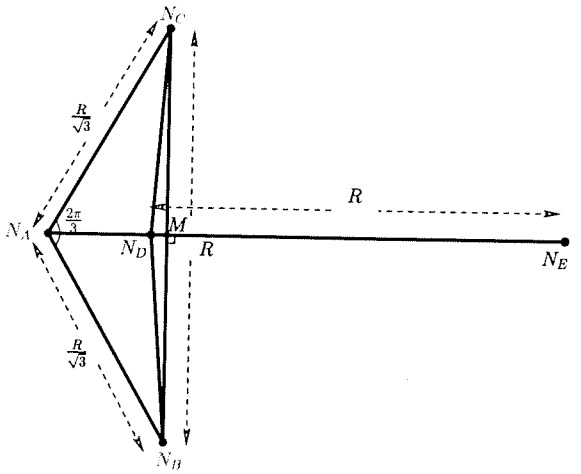

Figure 2.5: The counterexample for $\alpha = \frac{2\pi}{3}$.

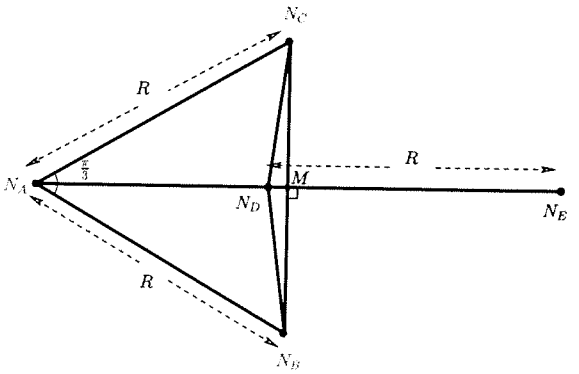

Figure 2.6: A counterexample for $\alpha = \frac{\pi}{3}$.

2.4 A Highly Efficient Self-Pruning broadcast algorithm

In this section, we propose a novel self-pruning broadcast algorithm that can significantly reduce redundant broadcasts in the network. As mentioned earlier, in self-pruning broadcast algorithms the receiver of the message decides whether or not to broadcast the message. Therefore, a potential advantage of self-pruning broadcast algorithms over neighbor-designating ones is that they do not increase the size of the message by adding a list of forwarding nodes.

2.4.1 Algorithm Structure

Algorithm 2.4 shows a general approach used in several self-pruning broadcast algorithms [23, 28]. Our proposed self-pruning broadcast algorithm employs this approach. Clearly, the main design issue of Algorithm 2.4 is to determine whether or not to broadcast a received message. A trivial algorithm is to refrain broadcasting if and only if all the neighbors have received the message during the defer period. Although this algorithm is simple to implement, it has limited effect in reducing the number of redundant broadcasts. Suppose N_A's defer time expires at t_0. Using the above strategy, node N_A will broadcast if some of its neighbors (at least one) have not received the message by t_0. However, this broadcast is redundant if all such neighbors receive the message from other nodes after time t_0. This scenario typically occurs when t_0 is small compared to the maximum defer time. In the next section, we introduce a responsibility-based scheme (RBS) that can significantly reduce the redundant broadcasts.

Algorithm 2.4 A general self-pruning algorithm

```
1: Extract the required information from the received message M
2: if M has been received before then
3:    drop the message
4: else
5:    set a defer timer
6: end if
7: When defer timer expires
8:    decide whether or not to schedule a broadcast
```

2.4.2 A Responsibility-Based Scheme

Algorithm 2.5 shows the proposed Responsibility-Based Scheme (RBS). The main idea of Algorithm 2.5 is that a node avoids broadcasting if it is not responsible for any of its neighbors. A node N_A is not responsible for a neighbor N_B if N_B has received the message or if there is another neighbor N_C such that N_C has received the message and N_B is closer to N_C than it is to N_A. Suppose N_A stores *ids* of all its neighbors that have broadcast the message during the defer period. When executed by a node N_A, Algorithm 2.5 first uses this information to determine which neighbors have not received the message (Lines 1–9 of Algorithm 2.5). It then returns *false* if and only if it finds a neighbor N_B that has not received the message and

$$\overline{AB} \leqslant \overline{BC}$$

for any N_A's neighbor N_C that has received the message. The output of RBS determines whether or not the broadcast is redundant.

Algorithm 2.5 A responsibility-based scheme (RBS)

Input: $List_A$: List of all neighbors of N_A, and $List_B$: List of broadcasting neighbors

Output: true or false

```
1: List_C ← List_A
2: for i = 1; i ⩽ length(List_C); i++ do
3:    for j = 1; j ⩽ length(List_B); j++ do
4:       if dist(List_C[i], List_B[j]) ⩽ R then
5:          removeElement(List_C[i], List_C)
6:          break
7:       end if
8:    end for
9: end for
10: List_D ← List_A − List_C
11: for i = 1; i ⩽ length(List_C); i++ do
12:    check ← true
13:    for j = 1; j ⩽ length(List_D); j++ do
14:       if dist(List_C[i], List_D[j]) ⩽ dist(List_C[i], N_A) then
15:          check ← false
16:          break
17:       end if
18:    end for
19:    if check then
20:       return (false)
21:    end if
22: end for
23: return (true)
```

Example 2.1. *As shown in Figure 2.7, N_A has five neighbors. Suppose that N_A has received a message from N_F. Note that N_A has the position of all its neighbors. Therefore, it can easily find that N_E and N_D have received the message but N_B and N_C have not. As shown in Figure 2.7, N_A is not required to broadcast because*

$$\overline{BE} < \overline{BA} \quad \text{and} \quad \overline{CD} < \overline{CA}.$$

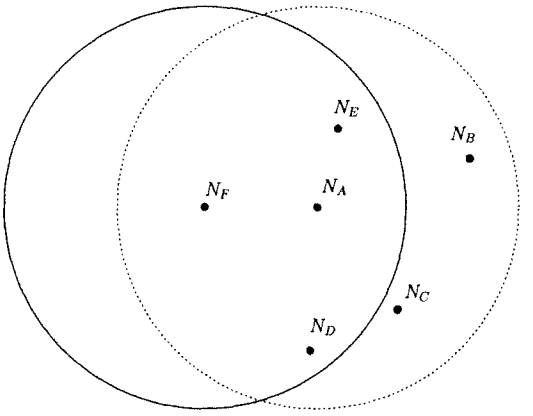

Figure 2.7: An example of an RBS decision.

Example 2.2. *Figure 2.8 shows an example of using the proposed self-pruning algorithm in a network with* 33 *nodes. In this example, the node S initiates the broadcast. It is easy to see that the nodes $A_1 \ldots A_7$ are not required to broadcast based on the responsibility condition. Therefore, they can immediately drop the packet after they receive it from S. Other neighbors of S initiate a defer timer with a random number. Thus, the next broadcasting node is random (it is F_1 in this example). Note that after F_1's transmission, some other nodes, which were previously set a defer timer, may drop the packet. For example, node B_1 would be no longer responsible for any of its neighbors after F_1's transmission. At last, as shown in Figure 2.8, only 7 nodes (represented by stars) will broadcast the packet.*

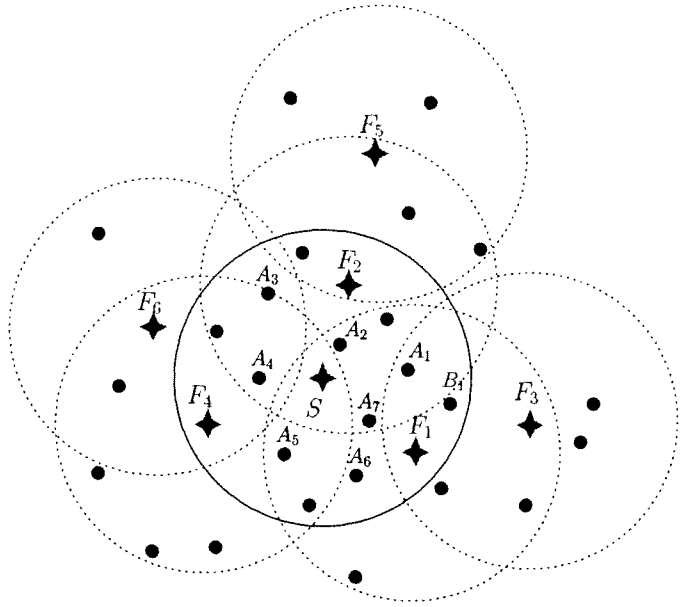

Figure 2.8: An instance of using the proposed broadcast algorithm.

Theorem 2.6. *In a collision-free network, Algorithm 2.4 can achieve* 100% *delivery if it uses the proposed responsibility-based scheme to determine whether or not to broadcast.*

Proof. Using Algorithm 2.4, each node broadcasts a message at most once. Therefore, broadcasting will eventually terminate. By contradiction, suppose there is at least one node that has not received the message after the broadcasting termination. Let us consider the set

$$\begin{aligned}\Lambda = \{(N_X, N_Y) | & N_X \text{ and } N_Y \text{ are neighbors,} \\ & N_X \text{ has received the message and} \\ & N_Y \text{ has not received the message}\}\end{aligned}$$

Suppose N_S is the node that initiated broadcasting and N_T is a node that has not received the message. The network is connected, thus there is a path between N_S and N_T. Clearly, we can find two neighbor nodes N_B and N_A along the path from N_T to N_S such that N_B has not received the message, while N_A has. Consequently, $(N_A, N_B) \in \Lambda$, thus $\Lambda \neq \emptyset$. As a result,

$$\exists (N_{A'}, N_{B'}) \in \Lambda \quad \text{s.t.} \quad \forall (N_X, N_Y) \in \Lambda : \quad \overline{A'B'} \leqslant \overline{XZ}. \tag{2.4}$$

Clearly, $N_{A'}$ has not broadcast since $N_{B'}$ has not received the message. Therefore, there must be a node $N_{C'}$ such that $N_{C'}$ has received the message and $\overline{C'B'} < \overline{A'B'} \leqslant R$. This result contradicts to (2.4), since $(N_{C'}, N_{B'}) \in \Lambda$. □

Theorem 2.7. *Time complexity of the proposed RBS is $O(n^2)$, where n is the number of neighbors.*

Proof. Algorithm 2.5 consists of two parts. In the first part (Lines 1–9), the algorithm generates a list of neighbors that have not received the message ($List_C$). Clearly, the time complexity of this part is $O(kn)$, where $1 < k \leqslant n$ is the number of broadcasting neighbors. In the second part, the algorithm checks whether there is a node N_B such that N_B has not received the message and $\overline{BA} \leqslant \overline{BC}$ for any neighbor $N_C \in List_D$. The time complexity of this part is $O(lm)$, where $0 \leqslant l \leqslant n$ is the number of neighbors that have not received the message and $1 \leqslant m \leqslant n$ is the number of neighbors that have received it. Therefore, the complexity of the algorithm is $O(lm + kn)$.

□

2.4.3 A Property of the Proposed RBS

In the simulation section (Section 2.5), we show that the proposed RBS can significantly reduce the number of transmissions in the network. In particular, our simulation shows that using RBS, the average number of broadcasts is less than one of the best-known approximations for the minimum number of required broadcasts. To justify this, we prove a property of the proposed RBS.

Assume that nodes are placed randomly inside a square area of size $L \times L$ using a homogeneous planar Poisson distribution. Therefore, nodes are independently and uniformly distributed in the area. Moreover, we have

$$Prb(\text{number of nodes in area } \tau = k) = \frac{(\delta\tau)^k e^{-\delta\tau}}{k!},$$

where δ is the density of nodes [29, 30]. Suppose node N_B receives the message from N_A for the first time. For simplicity, assume that circle $C_{A,2R}$ is completely inside the square area. Corollary 2.2 shows that the probability that N_B broadcasts the message exponentially decreases when the distance $\overline{AB}$ decreases or when the node density δ increases. This result is further confirmed by simulation in Section 2.5.

Example 2.3. *Suppose* $R = 250m$, $L = 1000m$ *and* $\delta L^2 = 300$ *(i.e., there are about* 300 *nodes in the network). Let* $Prb(Brd_B)$ *be the probability that* N_B *broadcasts the message after receiving it from* N_A. *Using Theorem 2.8 we get* $Prb(Brd_B) \leqslant 1.26 \times 10^{-2}$, $Prb(Brd_B) \leqslant 1.4 \times 10^{-3}$ *and* $Prb(Brd_B) \leqslant 10^{-4}$ *when* $\overline{AB} = 100m$, $\overline{AB} = 80m$ *and* $\overline{AB} = 60m$, *respectively.*

Let $\mathcal{R}_1, \mathcal{R}_2, \ldots \mathcal{R}_k$ be k non-overlapping regions inside the network. Suppose that $\zeta_{\mathcal{R}}$ is the event,

$$\zeta_{\mathcal{R}} = \{\text{The region } \mathcal{R} \text{ contains at least one node}\}.$$

Since the nodes are placed by homogeneous planar Poisson distribution, the events $\zeta_{\mathcal{R}_i}$ are independent [30]. Consequently, we have

$$Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_k}) = Prb(\zeta_{\mathcal{R}_1}) Prb(\zeta_{\mathcal{R}_2}) \ldots Prb(\zeta_{\mathcal{R}_k}). \tag{2.5}$$

Lemma 2.5 generalizes (2.5) to the case where the regions may overlap.

Lemma 2.5. *Let* $\mathcal{R}_1, \mathcal{R}_2, \ldots \mathcal{R}_k$ *be* k *regions inside the network. We have*

$$Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_k}) \geqslant Prb(\zeta_{\mathcal{R}_1}) Prb(\zeta_{\mathcal{R}_2}) \ldots Prb(\zeta_{\mathcal{R}_k}).$$

Proof. The proof is by induction on the number of regions. The lemma is true if the number of regions is one (i.e., $k = 1$). Suppose that the inequality holds for $k = d$ regions. We have

$$Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d} | \bar{\zeta}_{\mathcal{R}_{d+1}}) = Prb(\zeta_{\mathcal{R}_1 - \mathcal{R}_{d+1}}, \zeta_{\mathcal{R}_2 - \mathcal{R}_{d+1}}, \ldots, \zeta_{\mathcal{R}_d - \mathcal{R}_{d+1}}), \tag{2.6}$$

where $\bar{\zeta}_{\mathcal{R}_i}$ is the complement of $\zeta_{\mathcal{R}_i}$ and $R_i - R_j$ is the collection of all points inside R_i and outside R_j. Note that

$$Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d} | \zeta_{\mathcal{R}_1 - \mathcal{R}_{d+1}}, \zeta_{\mathcal{R}_2 - \mathcal{R}_{d+1}}, \ldots, \zeta_{\mathcal{R}_d - \mathcal{R}_{d+1}}) = 1.$$

Thus

$$Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d}) \geqslant Prb(\zeta_{\mathcal{R}_1 - \mathcal{R}_{d+1}}, \zeta_{\mathcal{R}_2 - \mathcal{R}_{d+1}}, \ldots, \zeta_{\mathcal{R}_d - \mathcal{R}_{d+1}}). \tag{2.7}$$

It follows from (2.6) and (2.7) that

$$Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d}) \geqslant Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d} | \bar{\zeta}_{\mathcal{R}_{d+1}}). \tag{2.8}$$

We have

$$Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d}) = Prb(\zeta_{\mathcal{R}_{d+1}}) Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d} | \zeta_{\mathcal{R}_{d+1}}) + Prb(\bar{\zeta}_{\mathcal{R}_{d+1}}) Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d} | \bar{\zeta}_{\mathcal{R}_{d+1}}).$$

Therefore, using (2.8) we get

$$Prb(\zeta_{\mathcal{R}_{d+1}}) Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d} | \zeta_{\mathcal{R}_{d+1}}) \geqslant (1 - Prb(\bar{\zeta}_{\mathcal{R}_{d+1}})) Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d}).$$

Thus, using induction hypothesis we get

$$\begin{aligned} &Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d}, \zeta_{\mathcal{R}_{d+1}}) \geqslant \\ &Prb(\zeta_{\mathcal{R}_{d+1}}) Prb(\zeta_{\mathcal{R}_1}, \zeta_{\mathcal{R}_2}, \ldots, \zeta_{\mathcal{R}_d}) \geqslant \\ &Prb(\zeta_{\mathcal{R}_1}) Prb(\zeta_{\mathcal{R}_2}) \ldots Prb(\zeta_{\mathcal{R}_d}) Prb(\zeta_{\mathcal{R}_{d+1}}). \end{aligned}$$

□

Lemma 2.6. *Let $\mathcal{D}_{A,R}$ and $\mathcal{D}_{B,R}$ be two disks with radius R and centers A and B, respectively. Suppose $\overline{AB} \leqslant R$. As shown in Figure 2.9, consider a point Q such that $R < \overline{QA}$ and $\overline{QB} \leqslant R$. Let $\mathcal{D}_{Q,\overline{QB}}$ be a disk with radius $\overline{QB}$. We have*

$$\Delta(\mathcal{I}(\mathcal{D}_{A,R}, \mathcal{D}_{B,R}, \mathcal{D}_{Q,\overline{QB}})) \geqslant \frac{\pi(R - \overline{AB})^2}{3},$$

where $\mathcal{I}(\mathcal{D}_{A,R}, \mathcal{D}_{B,R}, \mathcal{D}_{Q,\overline{QB}})$ is the intersection of disks $\mathcal{D}_{A,R}$, $\mathcal{D}_{B,R}$ and $\mathcal{D}_{Q,\overline{QB}}$ and $\Delta(R)$ is the area of region R.

Proof. For any point P on the circle $\mathcal{C}_{A,R}$ we have

$$\overline{BP} \leqslant \overline{AP} - \overline{AB} = R - \overline{AB}.$$

Therefore, as shown in Figure 2.9, the disk $\mathcal{D}_{B,(R-\overline{AB})}$ is inside $\mathcal{I}(\mathcal{D}_{A,R}, \mathcal{D}_{B,R})$. Consequently, we have

$$\Delta(I(\mathcal{D}_{A,R}, \mathcal{D}_{B,R}, \mathcal{D}_{Q,\overline{QB}})) \geqslant \Delta(I(\mathcal{D}_{B,(R-\overline{AB})}, \mathcal{D}_{Q,\overline{QB}})).$$

Since $\overline{QA} \geqslant R$, using triangle inequality we get

$$\overline{QB} \geqslant \overline{QA} - \overline{AB} \geqslant R - \overline{AB}.$$

Therefore, we have

$$\angle QBC \geqslant \frac{\pi}{3} \quad \text{and} \quad \angle QBD \geqslant \frac{\pi}{3}$$

and hence $\angle CBD \geqslant \frac{2\pi}{3}$. Therefore,

$$\Delta(I(\mathcal{D}_{B,(R-\overline{AB})}, \mathcal{D}_{Q,\overline{QB}})) \geqslant \frac{\Delta(\mathcal{D}_{B,(R-\overline{AB})})}{3} = \frac{\pi(R-\overline{AB})^2}{3}.$$

□

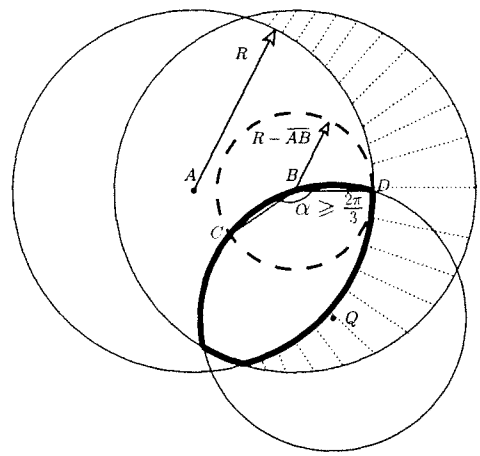

Figure 2.9: Finding a lower bound for $\Delta(I(\mathcal{D}_{A,R}, \mathcal{D}_{B,R}, \mathcal{D}_{Q,\overline{QB}}))$.

Theorem 2.8. *Suppose $d \leqslant R$ is the distance between two nodes N_A and N_B. We have*

$$Prb(Brd) \leqslant 1 - e^{-\delta\gamma e^{-\delta\frac{\pi(R-d)^2}{3}}},$$

where $Prb(Brd)$ is the probability that N_B broadcasts the message after receiving it from N_A, and

$$\gamma = \Delta(\mathcal{D}_{B,R}) - \Delta(\mathcal{I}(\mathcal{D}_{A,R}, \mathcal{D}_{B,R}))$$

is the area of the hatched crescent shown in Figure 2.9.

Proof. Node N_B is not required to broadcast if and only if

$$\zeta^* : \forall N_Q \in \Lambda : \quad \exists N_P \text{ inside } I(\mathcal{D}_{A,R}, \mathcal{D}_{B,R}, \mathcal{D}_{Q,\overline{QB}}),$$

where

$$\Lambda = \{N_X | \overline{AX} > R \text{ and } \overline{BX} \leqslant R\}.$$

Note that the nodes' positions have Poisson distribution. Therefore, using Lemma 2.6 we get

$$\begin{aligned} &Prb(\exists N_P \text{ inside } I(\mathcal{D}_{A,R}, \mathcal{D}_{B,R}, \mathcal{D}_{Q,\overline{QB}})) = \\ &1 - e^{-\delta\Delta(I(\mathcal{D}_{A,R},\mathcal{D}_{B,R},\mathcal{D}_{Q,\overline{QB}}))} \geqslant 1 - e^{-\delta\frac{\pi(R-d)^2}{3}}. \end{aligned} \tag{2.9}$$

Thus

$$Prb(Brd) = 1 - Prb(\zeta^*) = 1 - \sum_{k=0}^{\infty} Prb(|\Lambda| = k) Prb(\zeta^* | (|\Lambda| = k)).$$

where $|\Lambda|$ is the cardinality of the set Λ. Therefore, using (2.9) and Lemma 2.5 we get

$$Prb(Brd) \leqslant 1 - \sum_{k=0}^{\infty} \frac{(\delta\gamma)^k e^{-\delta\gamma}}{k!} (1 - e^{-\delta\frac{\pi(R-d)^2}{3}})^k = 1 - e^{-\delta\gamma e^{-\delta\frac{\pi(R-d)^2}{3}}},$$

where γ is the area of the hatched crescent in Figure 2.9 (collection of all points Q, $QA > R$ and $QB \leqslant R$)

$$\gamma = \Delta(\mathcal{D}_{B,R}) - \Delta(\mathcal{I}(\mathcal{D}_{A,R}, \mathcal{D}_{B,R})) = R^2(\pi - 2\arccos(\frac{d}{2R})) + d\sqrt{R^2 - (\frac{d}{2})^2}.$$

□

Corollary 2.2. *Using Theorem 2.8, we get*

$$Prb(Brd) \leqslant \delta\gamma e^{-\delta\frac{\pi(R-\overline{AB})^2}{3}},$$

Proof. consider the function

$$f(x) = x + e^{-x} - 1.$$

It is easy to show that $f(x)$ has a global minimum at $x = 0$. Therefore, we have

$$1 - e^{-x} \leqslant x$$

for any real number x. As a result, we get

$$Prb(Brd) \leqslant 1 - e^{-\delta\gamma e^{-\delta\frac{\pi(R-d)^2}{3}}} \leqslant \delta\gamma e^{-\delta\frac{\pi(R-d)^2}{3}}.$$

□

It is also possible that node N_B receives the message from more than one neighbor in its defer period. In this case, the number of N_B's neighbors that have received the message increases and the number of that have not received the message decreases. Consequently, the probability that N_B is required to broadcast the message further decreases compared to the case where N_B receives the message from only one neighbor. In Chapter 3, we prove that RBS can guarantee that the number of forwarding nodes is within a constant factor of the optimal solution (minimum CDS), if it is provided with partial information about 2-hop neighbors.

2.5 Simulation

2.5.1 Average Number of Nodes Selected by the Proposed Sliced-Based Algorithm

In Section 2.3, we proved that the proposed forwarding-node selection algorithm selects 11 nodes in the worst case. In practice, the number of selected nodes is typically less than 11. To avoid the complexity of mathematical analysis, we used simulation to find the average number of selected nodes. For a given number of neighbors $1 \leqslant n \leqslant 160$, we randomly put n points inside a circle with radius R. We then ran the proposed selection algorithm and obtained the number of selected nodes. To get the average number of selected nodes, we ran simulation 10^6 times for each given n. As shown in Figure 2.10, the average number of selected nodes is less than 6 and approaches 5 when n increases. Note that the proposed sliced-based selection algorithm does not necessarily select a B-coverage with a minimum number of nodes. However, there is a sliced-based selection algorithm that can find

a B-coverage with a minimum number of nodes in $O(n \log n)$ and can consequently reduce the average number of selected nodes. It is worth mentioning that Figure 9 shows the average number of selected nodes by the source node (the node that initiates the broadcasting). For the rest of broadcasting nodes, the average number of selected nodes is at least one less than that for the source node because of the optimization technique introduced in Section 2.3.

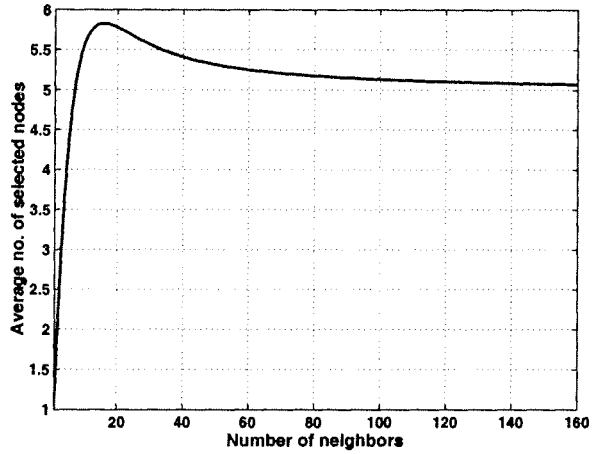

Figure 2.10: Average number of nodes selected by the proposed slice-based algorithm.

2.5.2 Probability of Broadcast using the Proposed RBS

Suppose that the proposed self-pruning algorithm is used for broadcasting in the network. Assume that node N_B receives a message from N_A for the first time. It has been proven that the probability of N_B broadcasting the message ($Prb(Brd_B)$) exponentially decreases when the distance $\overline{AB}$ decreases or when the node density δ increases. We used simulation to confirm this theoretical result. For the simulation, we considered two nodes N_A and N_B with distance $0 < d \leqslant R$ from each other. We uniformly placed nodes with density δ inside the network and checked whether or not N_B was required to broadcast the message. We ran simulation 10^6 times for a given δ and R. We then estimated $Prb(Brd_B)$ by the ratio of the number of times N_B was

required to broadcast over total number of runs.

Figures 2.11–2.14 show the simulation results for several values of δ, d and R. As shown in these figures, the probability of broadcast exponentially decreases when d decreases or when δ increases. For example, when R = 300m and $\delta = 4 \times 10^{-4}$, the probability of broadcast is 0.1 for d = 250m and reduces to 10^{-4} for d = 200m. This property can justify why the proposed self-pruning algorithm can significantly reduce the number of transmissions in the network. Figure 2.15 illustrates an instance of using RBS for the case where R = 300m, $\delta = 4 \times 10^{-4}$ and nodes are placed in a square area of $1000 \times 1000\text{m}^2$. As shown in Figure 2.15, only 9 nodes (represented by stars) among 400 nodes broadcast the message.

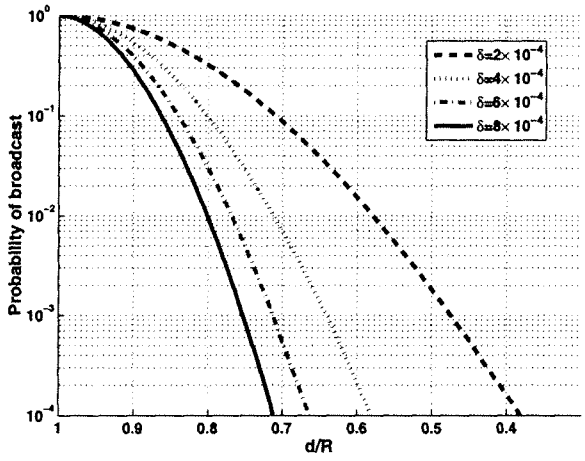

Figure 2.11: Probability of broadcast for R = 300m.

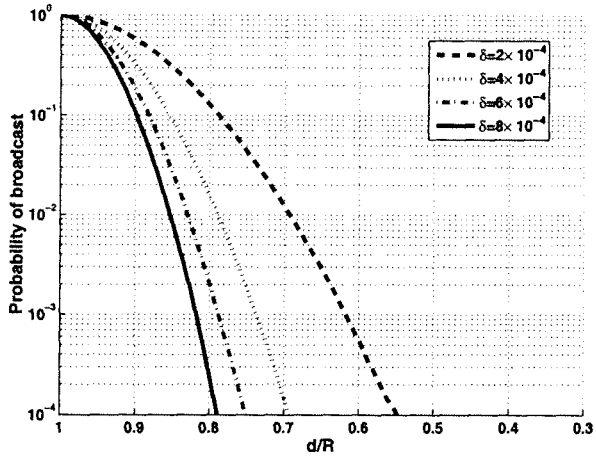

Figure 2.12: Probability of broadcast for $R = 400$m.

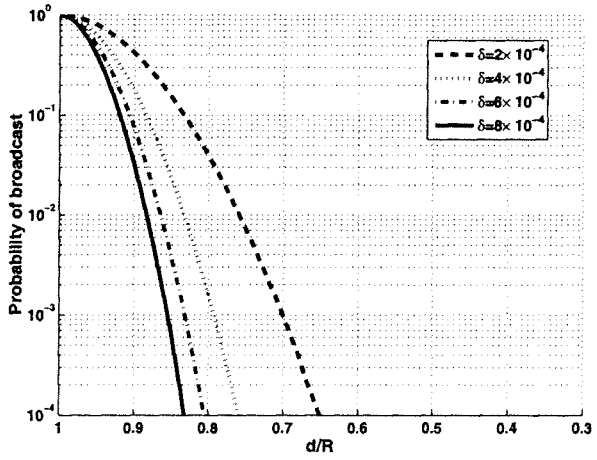

Figure 2.13: Probability of broadcast for $R = 500$m.

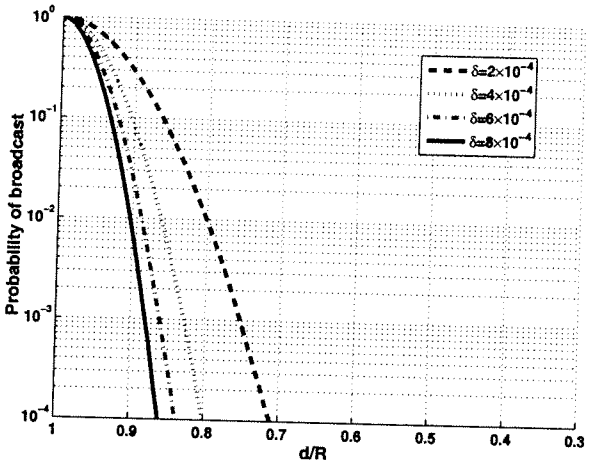

Figure 2.14: Probability of broadcast for $R = 600$m.

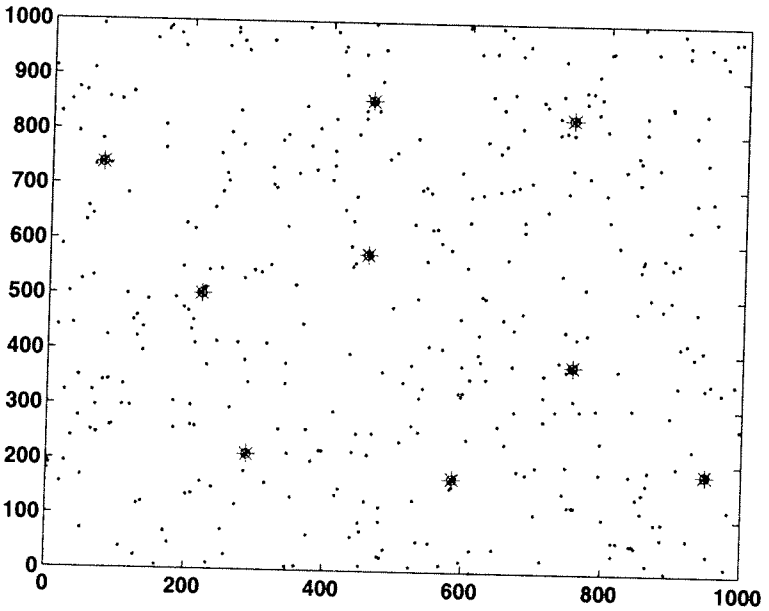

Figure 2.15: Broadcasting nodes in a $1000 \times 1000\text{m}^2$ square area with 400 nodes.

2.5.3 Performance of Proposed Neighbor-Designating and Self-Pruning Algorithms

The main objective of efficient broadcast algorithms is to reduce the number of transmissions. Therefore, we considered the ratio of broadcasting nodes over the total number of nodes as the metric to evaluate the performance of proposed broadcast algorithms. Using the ns-2 simulator, we evaluated this metric against two parameters: transmission range and node density. In each simulation run, we uniformly distributed N nodes in a $1000 \times 1000\, m^2$ square area, where $N = \delta \times$ (network area). A randomly generated topology was discarded if it leaded to a disconnected network. Only one broadcasting occurred in each simulation run by a randomly selected node. Table 2.1 summarizes some of the parameters used in ns-2. As shown in the table, the total number of nodes, N, varies within $25 - 1000$ and the transmission range varies within $50 - 300$ m. As a result, the simulation covers very sparse and very dense networks as well as the networks with large diameters. Table 2.2 shows the name and brief description of the algorithms used (in the simulation) for performance comparison.

Figures 2.16 and 2.17 show the average ratio of broadcasting nodes for 100 separate runs. The performance of our proposed algorithm is compared with the performance of Liu's algorithm [18] and the Edge Forwarding algorithm [31]. Using the Edge Forwarding algorithm, each node N_A divides its transmission coverage into six equal-size sectors A_{P_i}, where $1 \leqslant i \leqslant 6$. If N_A receives a new broadcast from another node, say N_B, it first determines the partition of N_B, say B_{P_j}, in which it is located. As the basic forwarding rule, the node N_A forwards the message if there exist at least one partition A_{P_k} such that no other host can be found in $B_{P_j} \cap A_{P_k}$. Using the basic and advanced forwarding rules proposed in [31], a node is not typically required to forward the packet unless it is close to the transmission perimeter of the broadcasting node.

In [18], Liu *et al.* showed that the number of redundant broadcasts using their broadcast algorithm is significantly lower than that of previous notable broadcast algorithms [31, 32]. As proved earlier, our proposed neighbor-designating algorithm has lower computational complexity and selects fewer forwarding nodes than Liu's algorithm. The simulation results, shown in Figures 2.16 and 2.17, indicate two interesting facts. First, our proposed neighbor-designating algorithm does not require more broadcasts than Liu's proposed broadcast algorithm. Secondly, the number of transmissions using

RBS is significantly lower than the number associated with the other implemented algorithms. In fact, as shown in Figures 2.16 and 2.17, this number is even less than one of the best-known approximations (ratio 8 approximation) for the minimum number of required broadcasts [16]. Note that in RBS, the probability that two close nodes broadcast the same message is very low. As a result, the number of broadcasting nodes is statistically bounded in a finite region (e.g., the transmission range of a node). However, using the slice-based and Liu's algorithm, the chance that two close nodes broadcast the same message is not negligible. For example, using Liu's algorithm, a node N_A selects the smallest subset of its 1-hop neighbors with the maximum coverage area, where the coverage area of a set of nodes is the union of their transmission coverage. As expected, most of the nodes around the transmission boundary of N_A will be selected by N_A since they often have contributions in the maximum coverage area of N_A's 1-hop neighbors. Therefore, it is likely for Liu's algorithm to select two close nodes around the transmission boundary of a node.

We repeated the simulation to consider a few more scenarios. In the first scenario, we changed the node distribution from uniform to a 2-dimensional Gaussian distribution. Both center and variance of the Gaussian distribution were randomly selected for each run. The variance was selected from the range 200-400 to avoid very dense population of nodes around the center of the distribution. As shown in Figure 2.18, the number of broadcasting nodes decreases for all the broadcast algorithms when a Gaussian distribution is used to distribute the nodes in the region. The simulation results indicate that the RBS algorithm still performs significantly better than other broadcast algorithms considered in this work.

In the second scenario, we used uniform distribution to distribute the nodes and evaluated the impact of message-reception failure on the performance of broadcast algorithms. We considered two networks of 100 and 400 nodes. The probability of message-reception failure was assumed to be equal and independent for each node in the network. For both networks, the maximum transmission range was set to 250m. Figures 2.19 and 2.20 compare the average delivery ratio of the broadcast algorithms for different probabilities of message-reception failure. As shown in these figures, the Edge Forwarding algorithm is the most robust broadcast algorithm against message-reception failure. This is because the impact of message loss is less when the broadcast redundancy is high. Interestingly, the robustness of the RBS algorithm significantly improves as the node density increases. The

simulation results indicate that the slice-based algorithm is the least robust broadcast algorithm against message-reception failure. This is mainly due to the fact that the slice-based algorithm selects a small number of nodes (less than 6 on average) to forward the message. Therefore, when the probability of message-reception failure is high, it is very likely that most of the selected nodes fail to receive and thus, forward the message.

Finally, we simulated the broadcast algorithms in a mobile wireless settings. The nodes were initially distributed using a uniform distribution. In the simulation, we used random walk mobility model and set the maximum velocity to 10m/s. We fixed the transmission range to 250m and varied the total number of nodes within $25 - 1000$. The simulation results indicate that all the broadcast algorithms considered in this chapter can achieve a high delivery ratio (above 95% on average) for $N \geqslant 50$, where N is the total number of nodes in the network. This is mainly because the implemented broadcast algorithms make broadcasting decisions "on-the-fly". For $N = 25$, the implemented algorithms failed to achieve a high delivery ratio because the network could easily get disconnected due to nodes' mobility. Clearly, broadcast algorithms cannot achieve high delivery ratio in such scenarios. It is worth mentioning that for $N \geqslant 50$, the ratio of broadcasting nodes is almost the same as the case where there is no mobility (see Figure 2.16).

Parameter	**Value**
Simulator	ns-2 (version 2.27)
MAC Layer	IEEE 802.11
Propagation Model	two-ray ground
Packet Size	256 bytes
Bandwidth	2 Mb/sec
Size of Square Area	$1000 \times 1000\text{m}^2$
Transmission Range	50–300m
Number of Nodes	25–1000

Table 2.1: Simulation parameters

Algorithm	Brief Description	Reference
Edge Forwarding	Nodes divide their transmission range into six equal sectors	[31]
Liu's Alg.	Each node selects the smallest subset of its neighbors with maximum coverage	[10]
Ratio-8 Approx.	A non-localized algorithm that gives an approx. factor 8 to MCDS. It is used as a benchmark.	[16]
Slice-Based Alg.	Our proposed neighbor-designating algorithm	Section 2.3
RBS Alg.	Our proposed self-pruning algorithm	Section 2.4

Table 2.2: Algorithms used in the simulation

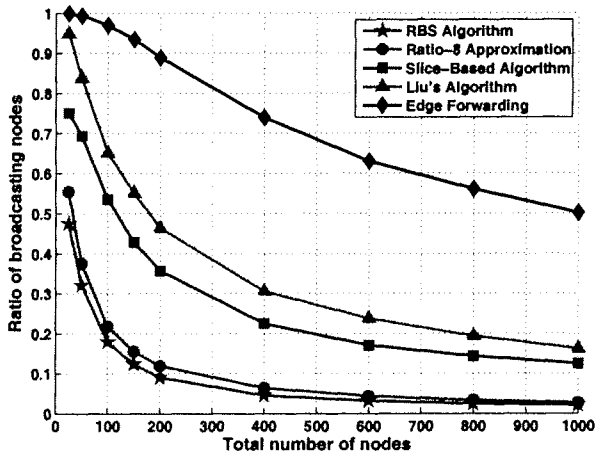

Figure 2.16: Ratio of broadcasting nodes vs. total number of nodes (uniform distribution).

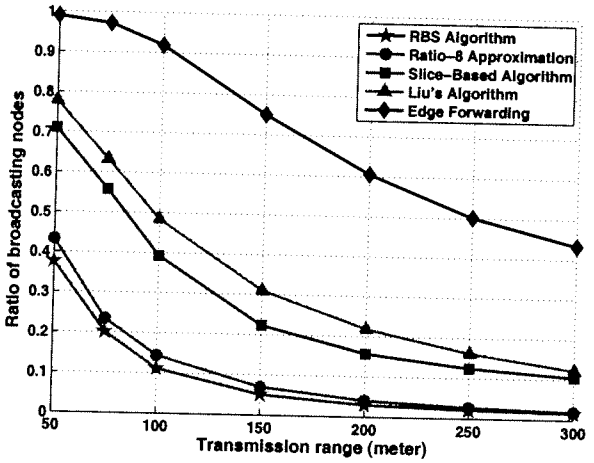

Figure 2.17: Ratio of broadcasting nodes vs. transmission range (uniform distribution).

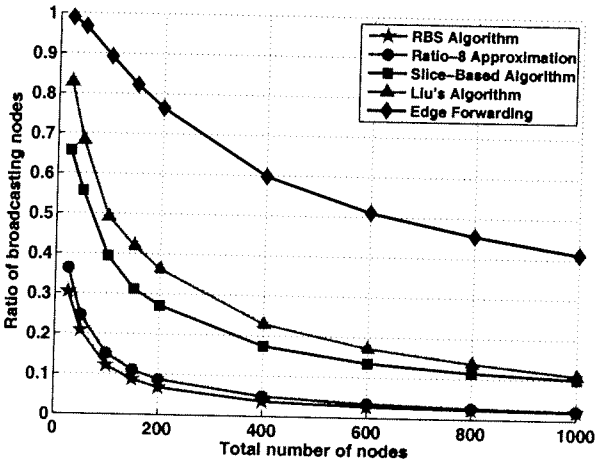

Figure 2.18: Ratio of broadcasting nodes vs. total number of nodes (Gaussian distribution).

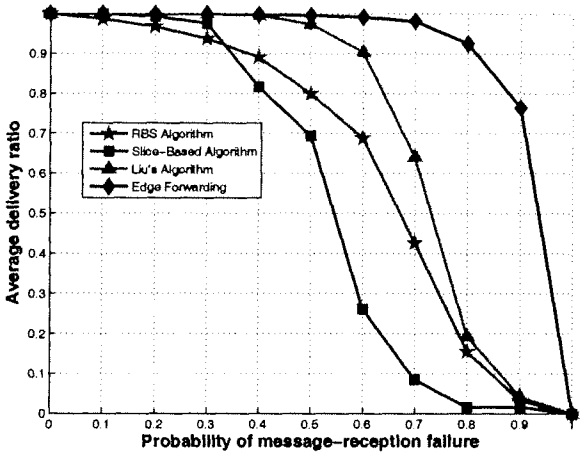

Figure 2.19: Average delivery ratio vs. probability of message-reception failure ($N = 100$).

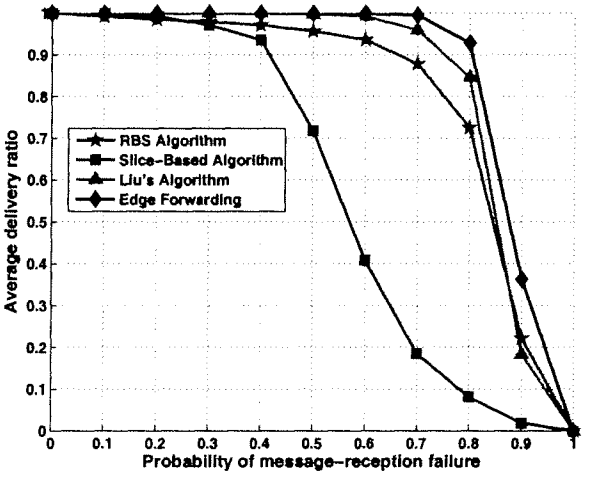

Figure 2.20: Average delivery ratio vs. probability of message-reception failure ($N = 400$).

2.6 Summary

In the first part of this chapter, we proposed a forwarding-node selection algorithm that selects at most 11 nodes in $O(n)$, where n is the number of neighbors. This limited number of nodes is an improvement over Liu's algorithm, which selects n nodes in the worst case and has time complexity $O(n \log n)$. Moreover, we showed that our proposed forwarding-node selection algorithm results in fewer broadcasts in the network.
In the second part of the chapter, we proposed an efficient self-pruning algorithm and showed why it significantly reduces the number of forwarding nodes in the network. Interestingly, the 2-hop-based version of our proposed self-pruning algorithm can guarantee constant approximation to the optimal solution (minimum CDS). To the best of our knowledge, this is the first broadcast algorithm that constructs a connected dominating set (CDS) "on-the-fly" and can guarantee both full delivery and a constant approximation ratio to the optimal solution.

Bibliography

[11] C. Perkins, "Ad hoc on demand distance vector (AODV) routing (internet-draft)," 1997, internet Draft, draft-ietf-manet-aodv-00.txt.

[12] D. Johnson and D. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, Imielinski and Korth, Eds. Kluwer Academic Publishers, 1996.

[13] S. Ni, Y. Tseng, Y. Chen, and J. Sheu, "The broadcast storm problem in a mobile ad hoc network," *In Proc. ACM International Conference on Mobile Computing and Networking (MOBICOM)*, pp. 151–162, 1999.

[14] M. Garey and D. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness.* New York, NY, USA: W. H. Freeman & Co., 1990.

[15] B. Clark, C. Colbourn, and D. Johnson, "Unit disk graphs," *Discrete Mathematics*, vol. 86, pp. 165–177, 1990.

[16] P. Wan, K. Alzoubi, and O. Frieder, "Distributed construction of connected dominating set in wireless ad hoc networks," *In Proc. of IEEE INFOCOM*, 2002.

[17] S. Funke, A. Kesselman, U. Meyer, and M. Segal, "A simple improved distributed algorithm for minimum CDS in unit disk graphs," *ACM Trans. Sensor Networks (TOSN)*, vol. 2, no. 3, pp. 444–453, 2006.

[18] H. Liu, P. Wan, X. Jia, X. Liu, and F. Yao, "Efficient flooding scheme based on 1-hop information in mobile ad hoc networks," *In Proc. of IEEE INFOCOM*, 2006.

[19] Y. Tseng, S. Ni, and E. Shih, "Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc networks," *In Proc. International Conference on Distributed Computing Systems (ICDCS)*, pp. 481–488, 2001.

[20] Y. Sasson, D. Cavin, and A. Schiper, "Probabilistic broadcast for flooding in wireless mobile ad hoc networks," *In Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1124–1130, 2003.

[21] W. Lou and J. Wu, "Double-covered broadcast (DCB): a simple reliable broadcast algorithm in manets," *In Proc. of IEEE INFOCOM*, pp. 2084–2095, 2004.

[22] J. Wu and F. Dai, "Broadcasting in ad hoc networks based on self-pruning," *In Proc. of IEEE INFOCOM*, 2003.

[23] W. Peng and X. Lu, "On the reduction of broadcast redundancy in mobile ad hoc networks," *In Proc. ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pp. 129–130, 2000.

[24] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher, "Range-free localization schemes in large scale sensor networks," *In Proc. ACM International Conference on Mobile Computing and Networking (MOBICOM)*, pp. 81–95, 2003.

[25] A. Keshavarz-Haddad, V. Ribeiro, and R. Riedi, "DRB and DCCB: efficient and robust dynamic broadcast for ad hoc and sensor networks," *In Proc. of IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, June 2007.

[26] R. Wattenhofer, L. Li, P. Bahl, and Y. Wang, "Distributed topology control for power efficient operation in multihop wireless ad hoc networks," *In Proc. of IEEE INFOCOM*, pp. 1388–1397, 2001.

[27] L. Li, J. Halpern, P. Bahl, Y. Wang, and R. Wattenhofer, "A cone-based distributed topology-control algorithm for wireless multi-hop networks," *IEEE/ACM Transactions on Networking*, vol. 13, pp. 147–159, 2005.

[28] M. Sun, W. Feng, and T. Lai, "Broadcasting in ad hoc networks based on self-pruning," *In Proc. of IEEE Global Telecommunications Conference (GLOBECOM)*, pp. 2842—2846, 2001.

[29] A. Papoulis, *Probability and statistics.* Prentice-Hall, Inc., 1990.

[30] R. Chang and R. Lee, "On the average length of delaunay triangulations," *BIT Numerical Mathematics*, vol. 24, pp. 269–273, 1984.

[31] Y. Cai, K. Hua, and A. Phillips, "Leveraging 1-hop neighborhood knowledge for efficient flooding in wireless ad hoc networks," *In Proc. IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pp. 347–354, 2005.

[32] J. Wu and H. Li, "On calculating connected dominating set for efficient routing in ad hoc wireless networks," *In Proc. International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DialM)*, pp. 7–14, 1999.

Chapter 3

Localized Broadcasting with Bounded Transmission Redundancy

3.1 Introduction

In this chapter, we carefully analyze and explore our proposed self-pruning broadcast algorithm (presented in Chapter 2) for the case where each node is provided with partial information about its 2-hop neighbors. After a brief introduction, we explain the related work in more details and specify the contribution of this work.

As discussed in Chapter 2, broadcasting is a fundamental communication primitive, which has many applications, including route discovery in wireless ad hoc routing protocols, and is frequently used to adapt network changes caused by the dynamic nature of ad hoc networks. Unfortunately, it is not practical to design a delivery-guaranteed broadcast algorithm that eliminates all redundant transmissions. This is because the problem of finding the minimum Connected Dominating Set (CDS) in a unit disk graph can be reduced to the problem of eliminating all of the redundant transmissions. It is well known that finding the Minimum CDS (MCDS) of a unit disk graph is NP-hard in general [33, 34]. Note that every CDS can be used as a backbone of the network to broadcast the message. On the other hand, the forwarding nodes in the delivery-guaranteed broadcast algorithms form a CDS. Therefore, broadcast algorithms can be used to find a CDS if a source node is selected to initiate the broadcast. Consequently, the problems of finding a minimum CDS and of designing an optimum broadcast algorithm can be reduced to each other.

A version of this chapter has been accepted for publication. M. Khabbazian and V.K. Bhargava, Localized Broadcasting with Guaranteed Delivery and Bounded Transmission Redundancy, *IEEE Transactions on Computers*, 2008.

To reduce the number of redundant transmissions broadcast algorithms typically impose some bandwidth overhead to, for example, collect neighbor information via message exchanges. It is desirable to reduce this bandwidth overhead to improve the practicality of the broadcast algorithm for ad hoc networks with frequent topology changes. In [35], the authors prove that every distributed algorithm for constructing a non-trivial CDS has the lower message complexity bound of $\Omega(N \log N)$, where N is the number of nodes and the message size is a constant multiple of the number of bits representing the node *ids* (a CDS is said to be trivial if it consists of all nodes). Let us define a distributed broadcast algorithm as a non-trivial broadcast algorithm if the forwarding nodes always form a non-trivial CDS. In this chapter, we show that the same lower bound does not hold for non-trivial broadcast algorithms, although they are closely related to the problem of finding a non-trivial CDS. In fact, we show that an extension of our proposed non-trivial broadcast algorithm requires $O(N)$ messages, where the message size is a constant. Moreover, we show that the number of forwarding nodes using the extended algorithm is within a constant factor of the optimum solution (i.e., minimum CDS).

Computational overhead also plays an important role in designing efficient broadcast algorithms. Flooding is an ideal broadcast algorithm in terms of computational overhead, since each node requires almost no computation (it simply broadcasts the first copy of the received message). The existing broadcast algorithms typically require some computation for selecting forwarding nodes or for self-pruning. In this chapter, we show how to efficiently implement the proposed broadcast algorithm using some computational geometry techniques to reduce the computational overhead.

3.1.1 Related Work

Different approaches can be used to classify the existing broadcast algorithms. One approach is based on whether or not they use a previously constructed backbone. One solution for reducing the number of redundant transmissions is to form a connected dominating set of nodes and use them as a backbone to broadcast the message. As mentioned earlier, finding the Minimum CDS (MCDS) is an NP-hard problem. However, there are distributed algorithms to find a small-sized CDS with constant approximation ratio to the MCDS [35, 36]. The main drawback of this solution is that maintaining a CDS is often costly in networks with frequent topology changes [37].

The second approach to classifying the existing broadcast algorithms is based on whether global or local information is employed by the algorithm. An algorithm is global if it uses whole or partial global state information. On the other hand, a distributed algorithm is localized if it is based on solely local information. It is clear that global broadcast algorithms are not appropriate for mobile ad hoc networks due to the dynamic nature of the network. In addition, they may not scale well when the number of nodes in the network increases.

The localized broadcast algorithms typically use $k \geqslant 0$ rounds of information exchange to collect k-hop neighbor information. Therefore, they can be classified based on the number of rounds of information exchanges. These algorithms can be further categorized based on whether or not they use any "side information" piggybacked in the packets. Our proposed algorithm is based on one round of information exchange but uses partial 2-hop neighbor information obtained by extracting the information piggybacked in the packets.

Some broadcast algorithms do not require any information exchange (i.e., $k = 0$). Flooding and probabilistic broadcast algorithms such as [38] and [39] fit in this category. Probabilistic algorithms cannot typically guarantee full delivery. However, they can reduce the number of redundant transmissions at low communication overhead and have great potential in unreliable communication environments. To decide whether or not to broadcast, probabilistic broadcast algorithms often use a threshold (such as probability of broadcast). Choosing a correct value of threshold is difficult. Moreover, the optimal value of the threshold may change due to network topology changes. An adaptive approach to this problem was presented in [40].

Most existing localized broadcast algorithms use k rounds of neighbor information, where $k \geqslant 1$ is a small number. These algorithms can be further divided into neighbor-designating (sender-based) and self-pruning (receiver-based). In neighbor-designating algorithms [37, 41], the forwarding status of each node is determined by other nodes. In other words, each broadcasting node selects a subset of its k-hop neighbors to forward the message. The main design challenge for neighbor-designating algorithms is to choose a small subset of nodes to forward the message. For example, in [37], the authors proposed a broadcast algorithm based on 1-hop neighbor information that selects the smallest subset of its 1-hop neighbors with the maximum coverage area, where the coverage area of a set of nodes is the union of their transmission coverage. When 2-hop neighbor information is available, it can

be extended to select the minimum number of 1-hop neighbors that cover all 2-hop neighbors. This procedure is known as the minimum forwarding set problem. There are heuristic algorithms in the literature that give a constant approximation ratio to the minimum forwarding set problem [42]. However, as shown in [41], even an optimal solution to this problem may not result in a small-sized CDS in the network. In [41], the authors extended the minimum forwarding set problem to the case where the complete 2-hop topology information is available. Note that two rounds of information exchange provide partial topology information for the 2-hop neighbor set. To get complete 2-hop topology information, each node must either get the position information of the 2-hop neighbors or perform three rounds of information exchange [41]. The authors show that their proposed algorithm can provide a probabilistic constant approximation ratio to MCDS, but point out that it performs poorly when the network becomes extremely dense.

In the self-pruning algorithms, each node determines its status (forwarding/non-forwarding) using local information [43, 44, 45]. Clearly, the main design challenge with regard to self-pruning algorithms is to find an effective self-pruning condition. Different self-pruning conditions have been proposed in the literature. For example, one simple self-pruning condition is whether all the neighbors have received the message before the defer timer expiration. Similar to the neighbor-designating algorithms, the existing self-pruning algorithms do not provide a constant approximation ratio to the optimal solution (MCDS).

3.1.2 Our Contribution

In the first part of the chapter, we consider two general structures commonly used in neighbor-designating and self-pruning algorithms based on 1-hop neighbor information. We show that using these structures, we are not able to guarantee both full delivery and a good bound on the number of forwarding nodes in the network. An open question is whether localized broadcast algorithms can provide both full delivery and constant approximation to the optimal solution (MCDS). As mentioned earlier, finding MCDS is an NP-hard problem. In the absence of global network information, this problem becomes even more challenging. It is a common belief that localized broadcast algorithms (e.g. self-pruning algorithms) cannot guarantee a constant approximation ratio to the optimal solution [43, 46]. One contribution of this chapter is to show that localized broadcast algorithms are in fact able

to guarantee a reasonable bound on the number of forwarding nodes.

In particular, we prove that our proposed self-pruning algorithm can provide full delivery as well as guaranteeing a constant approximation ratio to the optimal solution (MCDS) if the nodes are allowed to piggyback information with the broadcast packet. We design efficient algorithms with nearly optimum computational complexity to compute the proposed self-pruning condition.

It is proved in [35] that every distributed algorithm for constructing a non-trivial CDS has a lower message complexity bound of $\Omega(N \log N)$, where N is the number of nodes and the message size is a constant multiple of the number of bits representing the node *ids*. We show that the same message complexity lower bound does not apply to the non-trivial broadcast algorithms. In fact, we show that the message complexity of an extension of our proposed algorithm is $O(N)$, where N is the number of nodes and the message size is a constant. We prove that the extended broadcast algorithm can also guarantee full delivery and a constant approximation ratio to MCDS.

We show how to reduce the bandwidth requirements by reducing the amount of piggybacked information in each broadcast packet. We relax several system-model assumptions, or replace them with practical ones, to improve the practicality of the proposed broadcast algorithm. Finally, we verify the analytical results using simulation and provide a comparison with one of the best broadcast algorithms based on 1-hop information. The simulation results show that the proposed broadcast algorithm performs better than what was analytically guaranteed for the worst case.

3.2 System Model

In this section, we describe the system-model assumptions which are similar to those presented in Chapter 2. We discuss how to relax some of these assumptions in Section 3.5.

We consider a wireless network as a collection of N nodes represented by points located at their positions in a 2-D plane. Each node is equipped with an omni-directional antenna that has radio transmission range of R. Two distinct nodes are called neighbors if they are in transmission range of each other (i.e., the Euclidean distance between them is less than or equal to R). We assume that each node is able to obtain its position using an existing positioning technique, such as the Global Positioning System (GPS). Each

node periodically broadcasts a short *Hello* message containing its unique *id* and its position. Therefore, each node knows the position of its 1-hop neighbors as well.

To prove that a broadcast algorithm guarantees full delivery, we assume that nodes are static during the broadcast and that the Medium Access Control (MAC) layer is ideal, i.e., there are no transmission errors such as collision and contention. In addition, we assume that the network is connected. A broadcast algorithm can be considered a delivery-guaranteed broadcast algorithm if it guarantees full delivery under these assumptions. Note that without these assumptions, even flooding may not achieve full delivery.

3.3 Broadcast Algorithms Based on 1-hop Neighbor Information

In this section, we consider two general structures commonly used in neighbor-designating and self-pruning algorithms based on 1-hop neighbor information. These structures require one round of information exchange and employ only 1-hop neighbor information to make decision. We show that using these structures we are not able to guarantee both full delivery and a good bound on the number of forwarding nodes in the network.

3.3.1 Neighbor-Designating Algorithms

Algorithm 3.1 shows a general structure of neighbor-designating broadcast algorithms based on 1-hop neighbor information. As shown in Algorithm 3.1, each node schedules a broadcast for a received message if the node is selected by the sender and if it has not received the same message before. Let us represent a node N_A with a point A located at its position and use $\overline{PQ}$ to denote the Euclidean distance between two points P and Q. Suppose Algorithm 3.1 is used as the basic structure in a delivery-guaranteed broadcast algorithm based on 1-hop neighbor information. Let P be a point such that $\overline{AP} \leqslant R$ and for every node $N_B \neq N_A$ in the network $\overline{BP} > R$. Recall that each node uses 1-hop neighbor information, thus only N_A knows whether or not there is a node at the position of point P. Consequently, N_A must be selected to forward the message in order to guarantee full delivery.

For example, suppose N_A is a node on the boundary of the network. As shown in Figure 3.1, let L be a line containing A such that all the nodes

are either on the line or one side of it. This situation occurs, for instance, when A is a vertex of the network convex hull [47]. Consider a point P on the other side of the line L such that $\overline{AP} = R$ and $AP \perp L$. It is clear that $\overline{BP} > R$ for every node $N_B \neq N_A$. Therefore, N_A has to be selected to forward the message. This implies that most of the nodes around the boundary of the network (including all the nodes on the convex hull [47] of the network) will broadcast the message if Algorithm 3.1 is used as the basic structure in a delivery-guaranteed broadcast algorithm based on 1-hop neighbor information. Consequently, neighbor-designating algorithms based on 1-hop neighbor information may not be efficient if a large number of nodes (compared to the total number of nodes) are located around the boundary of the network. The following theorem shows that a broadcast algorithm based on 1-hop neighbor information cannot guarantee both full delivery and a good bound on the number of forwarding nodes if it uses Algorithm 3.1 as its basic structure.

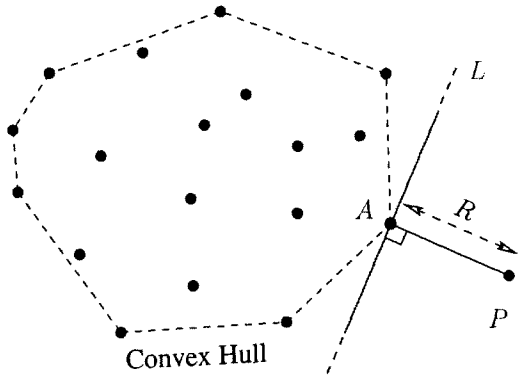

Figure 3.1: An example of a node on the network boundary.

Theorem 3.1. *Suppose Algorithm 3.1 is used as the basic structure in a delivery-guaranteed broadcast algorithm based on 1-hop neighbor information. The ratio of the number of broadcasting nodes over the minimum number of required broadcasts can be as large as N, where N is the number of nodes in the network.*

Proof. As shown in Figure 3.2, suppose that all the nodes are located on a line segment AB of length R or on a circle $C_{O,\frac{R}{2}}$ with a radius $\frac{R}{2}$, where R is the radio transmission range. For every node N_A, we can find a point

Algorithm 3.1 A general structure of neighbor-designating algorithms

```
1: Extract the required information from the received message M
2: if M has been received before or does not contain node's id then
3:    drop the message
4: else
5:    set a defer timer
6: end if
7: When defer timer expires
8:   Based on 1-hop neighbor information:
9:   Select a subset of neighbors to forward the message
10:  Attach the list of forwarding nodes to the message
11:  Broadcast the message
```

P outside the network boundary such that $\overline{AP} = R$ and AP is orthogonal to the line segment AB/Circle $C_{O,\frac{R}{2}}$. It is easy to show that $\overline{BP} > R$ for every node $N_B \neq N_A$. Therefore, all of the nodes will broadcast the message. However, one broadcast is enough to transmit a message to all the nodes in the network for each scenario. Consequently, the ratio of broadcasting nodes over the minimum number of required broadcasts is N. □

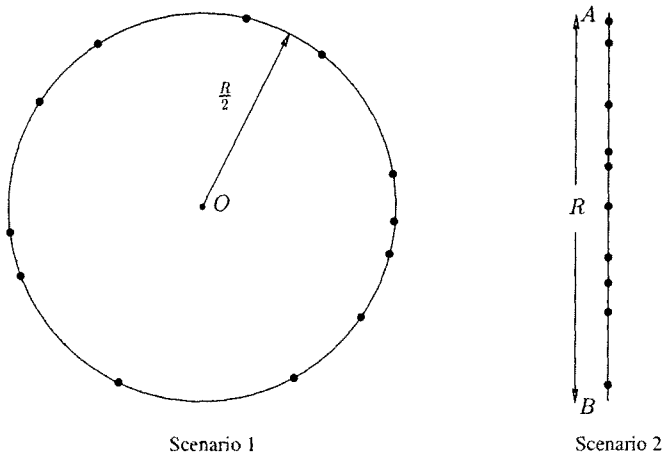

Figure 3.2: Two worst-case examples for Theorem 3.1.

3.3.2 Self-Pruning Algorithms

Algorithm 3.2 shows a general structure of self-pruning broadcast algorithms based on 1-hop neighbor information. Using this structure, a node schedules a broadcast for the first received copy of the message. When the defer timer expires, the node may refrain from broadcasting the message if a certain self-pruning condition is satisfied. Theorem 3.2 shows that this structure cannot guarantee both full delivery and a good bound on the number of forwarding nodes. Note that in Algorithm 3.2, no information is piggybacked with the broadcast packet. An extension of Algorithm 3.2 is to allow nodes to piggyback some information in the broadcast packet. Clearly, this extension of Algorithm 3.2 is stronger than both Algorithm 3.1 and Algorithm 3.2. In the next section, we show that a broadcast algorithm based on 1-hop neighbor information can guarantee both full delivery and a good bound on the number of forwarding nodes (a constant approximation ratio to MCDS) if it employs this extension of Algorithm 3.2.

Theorem 3.2. *Suppose Algorithm 3.2 is used as the basic structure in a delivery-guaranteed broadcast algorithm based on 1-hop neighbor information. The ratio of the number of broadcasting nodes over the minimum number of required broadcasts is $\theta(N)$ in the worst case.*

Proof. Consider the case where all the nodes are on two circles $C_{O,\frac{R}{2}}$ and $C_{O,\frac{3R}{2}}$ centered at O with radii $\frac{R}{2}$ and $\frac{3R}{2}$, respectively. As shown in Figure 3.3, suppose that for every node N_{A_i} on $C_{O,\frac{R}{2}}$ there is a corresponding node N_{B_i} on $C_{\frac{3R}{2}}$ (and vice versa) such that $\overline{A_iB_i} = R$. By contradiction, suppose that neither N_{A_i} nor N_{B_i} broadcast the message. Note that N_{A_i} and N_{B_i} do not share any neighbor. Since they have only 1-hop neighbor information and there is no information piggybacked in the broadcast packet, N_{A_i} (or N_{B_i}) cannot be sure whether its corresponding node has received the message. Therefore, in order to guarantee full delivery, either N_{A_i} or N_{B_i} has to broadcast the message. However, only a constant number of transmissions is enough to send the message to all the nodes. Therefore, the ratio of broadcasting nodes over the minimum number of required broadcasts is $\theta(N)$. □

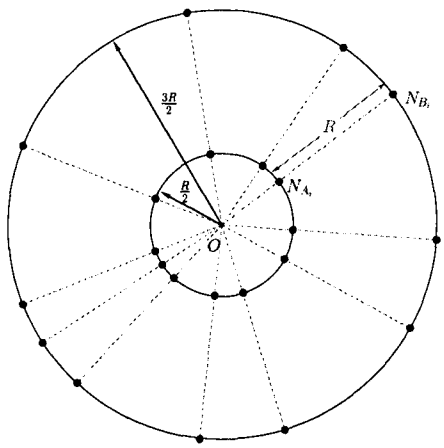

Figure 3.3: A worst-case example for Theorem 3.2.

Algorithm 3.2 A general structure of self-pruning algorithms

```
1: if the message M has been received before then
2:     drop the message
3: else
4:     set a defer timer
5: end if
6: When defer timer expires
7:   Decide whether or not to broadcast M
```

3.4 An Efficient Self-Pruning Broadcast Algorithm

In the previous section, we considered two structures typically used in the broadcast algorithms based on 1-hop neighbor information. We showed that these structures cannot guarantee the following properties simultaneously:

- full network delivery
- good bound on the number of broadcasting nodes

An open question is whether a localized broadcast algorithm (e.g., a neighbor-knowledge-based broadcast algorithm) is able to guarantee these properties. In this section, we propose a novel self-pruning broadcast algorithm based

on one round of information exchange that not only guarantees full delivery but also a constant approximation ratio to the optimal solution (MCDS). As its basic structure, the proposed broadcast algorithm (Algorithm 3.3) uses a simple extension of Algorithm 3.2 in which each node is allowed to piggyback some information within the broadcast packet. In Algorithm 3.3, each node adds a list of its 1-hop neighbor *id*s and positions into the broadcast packet. Upon receiving a packet, this information is extracted and used to determine the node's status (forwarding/non-forwarding) based on the following self-pruning condition.

Definition 3.1 (Responsibility Condition). *Node N_A is pruned (has non-forwarding status) if it is not responsible for any of its neighbors. Node N_A is not responsible for its neighbor N_B if N_B has received the message or if there is another node N_C such that N_C has received the message and N_B is closer to N_C than N_A.*

Algorithm 3.3 The proposed broadcast algorithm

1: Extract the required information from the received packet
2: Add the broadcasting node and its 1-hop neighbor information (except the node's own information) to the list $List_{ID}^{Rec}$
3: **if** the message has been received before **then**
4: drop the message
5: **else**
6: set a defer timer
7: **end if**
8: When defer timer expires
9: **if** the responsibility condition is satisfied **then**
10: Remove the previous information attached to the message
11: Attach the list of 1-hop neighbors to the message
12: Broadcast the packet
13: **end if**

Suppose N_A computes a list of nodes that have received the message ($List_{N_A}^{Rec}$) by collecting the information piggybacked in the broadcast packets. To check the responsibility condition, N_A can first determine which neighbors have not received the message. The responsibility condition is then satisfied if and only if there is a neighbor N_B that has not received the message and

$$\forall N_C \in List_{N_A}^{Rec}: \quad \overline{AB} \leqslant \overline{CB}.$$

In Section 3.4.2, we describe efficient algorithms for computing the responsibility condition.

Example 3.1. *As shown in Figure 3.4, N_A has six neighbors. Suppose that N_A has received the message from N_H. Recall that N_A extracts the list of N_H's neighbors from the received packet. Therefore, it knows that N_E, N_F and N_G have received the message and N_B, N_C and N_D have not. According to the responsibility condition, N_A is not required to broadcast because*

$$\overline{BE} < \overline{BA}, \quad \overline{CF} < \overline{CA} \quad \text{and} \quad \overline{DG} < \overline{DA} \,(\text{or } \overline{DF} < \overline{DA}).$$

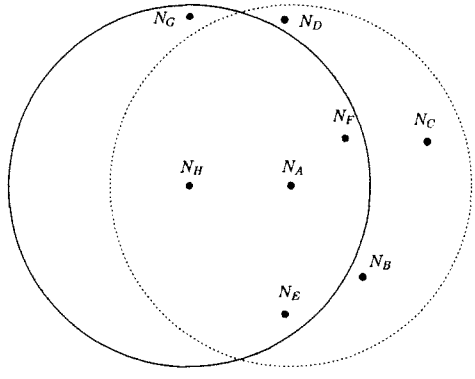

Figure 3.4: An example of self-pruning based on the responsibility condition.

3.4.1 Analysis of the Proposed Broadcast Algorithm

In this section, we prove that Algorithm 3.3 can achieve full delivery and a constant approximation ratio to MCDS. In order to prove these properties, we assume that nodes are static during the broadcast, the networks is connected and the MAC layer is ideal, i.e., there is no communication error. As mentioned earlier in Section 3.2, even flooding cannot guarantee full delivery without these assumptions.

Theorem 3.3. *Algorithm 3.3 guarantees that all the nodes in the network will receive the message.*

Proof. Using Algorithm 3.3, each node broadcasts the message at most once. Therefore, broadcasting will eventually terminate. By contradiction, suppose there is at least one node that has not received the message after the

broadcast termination. Let us consider the set

$$\Lambda = \{(N_X, N_Y) | N_X \text{ and } N_Y \text{ are neighbors, } N_X \text{ has received the message and } N_Y \text{ has not received the message}\}$$

Suppose N_S is the source node (the node that initiated the broadcast) and N_T is a node that has not received the message. The network is connected, thus there is a path between N_S and N_T. Clearly, we can find two neighbor nodes N_B and N_A along the path from N_T to N_S such that N_B has not received the message, while N_A has. Consequently, $(N_A, N_B) \in \Lambda$, thus $\Lambda \neq \emptyset$. As a result,

$$\exists (N_{A'}, N_{B'}) \in \Lambda \text{ s.t. } \forall (N_X, N_Y) \in \Lambda : \ \overline{A'B'} \leqslant \overline{XY}. \tag{3.1}$$

Clearly, $N_{A'}$ has not broadcast since $N_{B'}$ has not received the message. Therefore, there must be a node $N_{C'}$ such that $N_{C'}$ has received the message and $\overline{C'B'} < \overline{A'B'} \leqslant R$. This result contradicts to (3.1), since $(N_{C'}, N_{B'}) \in \Lambda$. □

Lemma 3.1. *Using Algorithm 3.3, the number of broadcasting nodes inside a disk $D_{O,\frac{R}{4}}$ with a radius $\frac{R}{4}$ is bounded by a constant.*

Proof. As shown in Figure 3.5, consider three disks $D_{O,\frac{R}{4}}$, $D_{O,\frac{3R}{4}}$ and $D_{O,\frac{5R}{4}}$, centered at O with radii $\frac{R}{4}$, $\frac{3R}{4}$ and $\frac{5R}{4}$, respectively. Let us define

$$\mathcal{R}_1 - \mathcal{R}_2 = \{P | P \in \mathcal{R}_1 \text{ and } P \notin \mathcal{R}_2\},$$

where $\mathcal{R}_1$ and $\mathcal{R}_2$ are two regions and P is a point. Suppose k is the minimum number such that for every set of k points $P_i \in D_{O,\frac{5R}{4}} - D_{O,\frac{3R}{4}}$, $1 \leqslant i \leqslant k$, we have

$$\exists P_i, P_j : \quad i \neq j \quad \text{and} \quad \overline{P_i P_j} \leqslant \frac{R}{2}.$$

Note that the area $D_{O,\frac{5R}{4}} - D_{O,\frac{3R}{4}}$ can be covered with a constant number of disks with radius $\frac{R}{4}$. If the number of points inside $D_{O,\frac{5R}{4}} - D_{O,\frac{3R}{4}}$ is greater than the number of covering disks, at least one covering disk will contain more than one point. For two points P_1 and P_2 inside a disk with a radius $\frac{R}{4}$, we have $\overline{P_1 P_2} \leqslant \frac{R}{2}$. Therefore, k is bounded by a constant (the number of covering disks plus one).

We prove that the number of broadcasting nodes inside $D_{O,\frac{R}{4}}$ is less than or equal to k. By contradiction, suppose that there are more than k broadcasting nodes inside $D_{O,\frac{R}{4}}$. Let $N_{A_i} \in D_{O,\frac{R}{4}}$, $0 \leqslant i \leqslant k$ be the first $k+1$

broadcasting nodes ordered chronologically based on their broadcast time. Based on the responsibility condition, for every node N_{A_i} there is a corresponding neighbor N_{B_i} such that $N_{B_i} \notin List^{Rec}_{N_{A_i}}$ and $\overline{A_iB_i} \leqslant \overline{CB_i}$ for every node $N_C \in List^{Rec}_{N_{A_i}}$, where $List^{Rec}_{N_{A_i}}$ is the list of nodes that have received the message based on N_{A_i}'s collected information. Note that every node inside $D_{O,\frac{3R}{4}}$ will receive the message after N_{A_0}'s broadcast. Therefore, $N_{B_i} \notin D_{O,\frac{3R}{4}}$ for $1 \leqslant i \leqslant k$. On the other hand, N_{B_i} is a neighbor of N_{A_i}, thus $N_{B_i} \in D_{O,\frac{5R}{4}}$. Consequently,

$$\forall 1 \leqslant i \leqslant k: \quad N_{B_i} \in D_{O,\frac{5R}{4}} - D_{O,\frac{3R}{4}}.$$

It follows that

$$\forall 1 \leqslant i \leqslant k: \quad \overline{A_iB_i} > \frac{R}{2}. \tag{3.2}$$

Suppose $1 \leqslant i < j \leqslant k$. The nodes N_{A_i} and N_{A_j} are neighbors, because $\overline{A_iA_j} \leqslant \frac{R}{2}$. Recall that N_{A_i} piggybacks the list of its neighbors within the broadcast packet. Therefore, $N_{B_i} \in List^{Rec}_{N_{A_j}}$, thus $N_{B_i} \neq N_{B_j}$. It follows that $N_{B_1} \ldots N_{B_k}$ are k different points inside $D_{O,\frac{5R}{4}} - D_{O,\frac{3R}{4}}$. Therefore

$$\exists N_{B_i}, N_{B_j}: \quad i < j \quad \text{and} \quad \overline{B_iB_j} \leqslant \frac{R}{2}. \tag{3.3}$$

This is a contradiction, because $N_{B_i} \in List^{Rec}_{N_{A_j}}$ and based on (3.2) and (3.3)

$$\overline{A_jB_j} > \frac{R}{2} \Longrightarrow \overline{A_jB_j} > \overline{B_iB_j}.$$

Thus, according to the responsibility condition, N_{A_j} is not responsible for N_{B_j}. □

Theorem 3.4. *Algorithm 3.3 gives a constant approximation ratio to the optimal solution (MCDS).*

Proof. Clearly, a disk with radius R can be covered with a constant number of disks with radius $\frac{R}{4}$. Therefore, based on Lemma 3.1, the number of broadcasting nodes inside a disk with radius R is bounded by a constant C_{min}. Let $|MCDS|$ be the number of nodes in the MCDS. Each broadcasting node is inside the transmission range of at least one node in th MCDS. On the other hand, the number of broadcasting nodes inside the transmission range of each node in the MCDS is bounded by C_{min}. Therefore, the total number of broadcasting nodes is not more than $C_{min} \times |MCDS|$. □

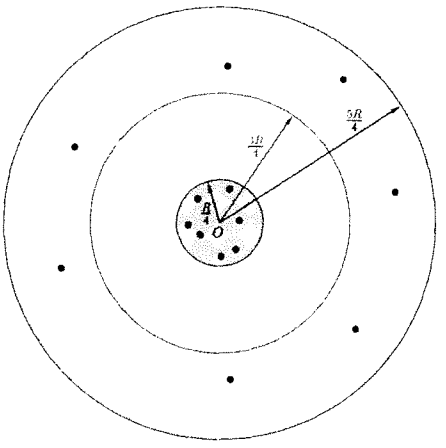

Figure 3.5: Three co-centered disks; $N_{A_i} \in D_{O,\frac{R}{4}}$ and $N_{B_i} \in D_{O,\frac{5R}{4}} - D_{O,\frac{3R}{4}}$.

The following theorem shows that the message complexity of the proposed broadcast algorithm is $O(N)$, where N is the total number of nodes in the network and each message contains a node's *id* and its position. In Section 3.5, we will show that the message size can be reduced to a constant number of bits by removing the node *id* from the message and using a constant number of bits to represent each node's position.

Theorem 3.5. *The message complexity of Algorithm 3.3 is $O(N)$, where N is the total number of nodes in the network and each message contains a node's id and its position.*

Proof. There are two types of messages used in the proposed broadcast algorithm. First, each node broadcasts its *id* and position in a short *Hello* message. Second, each broadcasting node piggybacks the list of its 1-hop neighbor *id*s and positions into the broadcast packet. We will consider this packet as n separate messages, where n is the number of 1-hop neighbors of the broadcasting node. Therefore, each message contains exactly one node's *id* together with its position. As shown in the proof of Theorem 3.4, the number of broadcasting neighbors of each node is bounded by a constant C_{min}. Therefore, each node's *id* and position appears in at most $C_{min} + 1$ messages, including the *Hello* message. Consequently, the total number of messages is not more than $(C_{min} + 1) \times N$. □

3.4.2 Efficient Algorithms for Computing the Responsibility Condition

The problem of computing the responsibility condition can be expressed by the following geometry problem:

Definition 3.2 (Closest-Point Problem (CPP)). *Let* $\mathcal{P} = \{P_0, P_1, \ldots, P_m\}$ *and* $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_n\}$ *be two disjoint sets of points. Is there a point* $Q \in \mathcal{Q}$ *such that*

$$\forall P \in \mathcal{P}: \quad \overline{QP_0} \leqslant \overline{QP}?$$

The closest-point problem attempts to establish whether or not P_0 is the closest point in $\mathcal{P}$ to at least one point in $\mathcal{Q}$. One approach to solving CPP is to find the closest point in $\mathcal{P}$ to every point in $\mathcal{Q}$. The trivial algorithm for finding the closest point in $\mathcal{P}$ to a query point $Q \in \mathcal{Q}$ is to compute the distance from Q to all the points in $\mathcal{P}$, keeping track of the "closest point so far". Clearly, using this algorithm, the computational complexity of solving the CPP would be $O(m \times n)$. Finding the closest point in $\mathcal{P}$ to a query point $Q \in \mathcal{Q}$ is a well-known problem called the Nearest Neighbor Search (NNS) problem. To solve the NNS problem, several space-partitioning methods, including *kd*-tree and the Voronoi diagram [47], can be used. A *kd*-tree is a data structure used for organizing points in k-dimensional space. A *kd*-tree for the set of 2-D points $\mathcal{P}$ can be constructed in $O(m \log m)$. Moreover, the computational complexity of a query is $O(\log m)$ [47]. Therefore, using *kd*-tree, the CPP can be solved in $O(m \log m + n \log m)$. Similarly, we can use the Voronoi diagram to solve CPP. The Voronoi diagram is a partitioning of a plane with n generating points into convex polygons called Voronoi cells, such that each cell contains exactly one generating point and every node in the cell is closer to the generating point of the cell than to other generating points. It is known that the Voronoi diagram can be constructed in $O(n \log n)$ and that each cell has $O(n)$ edges in the worst case. Therefore, using the set of points $\mathcal{P}$ as the generating points, the Voronoi diagram can be generated in $O(m \log m)$. The node P_0 is the closest node in $\mathcal{P}$ to a query point Q if and only if Q is in the Voronoi cell of P_0. We can check whether a query point Q is in the Voronoi cell of P_0 in $O(\log m)$, since the Voronoi cell of P_0 is a convex polygon with at most $O(m)$ edges. Therefore, the CPP can be solved in $O(m \log m + n \log m)$ using the Voronoi diagram. Note that we only need to compute the Voronoi cell of P_0. The Voronoi cell of P_0 can be computed in $O(m \log h)$, where h is the number of edges of the cell. The

average number of vertices of a Voronoi cell is less than six [47], thus h is a constant on average. Therefore, the average case computational complexity of this approach is $O(m+n)$.

Suppose P_0 is not the closest point in $\mathcal{P}$ to any node in $\mathcal{Q}$. In this case, $\Omega(n)$ is a lower bound for every algorithm to solve the CPP for sets $\mathcal{P}$ and $\mathcal{Q}$, because the algorithm cannot ignore any node in the set $\mathcal{Q}$. On the other hand, every algorithm to solve the CPP has to consider all the points in $\mathcal{P}$ if P_0 is the closest point in $\mathcal{P}$ to a node in $\mathcal{Q}$. Thus, $\Omega(m)$ is also a lower bound for such algorithms. Considering both cases, it follows that $\Omega(m+n)$ is a lower bound for every algorithm to solve the CPP. We described algorithms that compute the CPP in $O((n+m)\log m)$, in the worst case. These algorithms are nearly optimal in terms of computational complexity, because $\log m$ is a small factor in practice. Moreover, by constructing the Voronoi cell of P_0, we showed that the CPP can be solved in $O(n+m)$ in the average case.

Theorem 3.6. *The worst-case complexity of computing the responsibility condition is $O(\Delta_G \log \Delta_G)$, where Δ_G is the maximum node degree of the network.*

Proof. Based on Algorithm 3.3, a node N_A stores the broadcasting neighbors' information and their 1-hop neighbors' information (except N_A's information) in $List_{N_A}^{Rec}$. The node N_A can compute $List_{N_A}^{NotRec}$, the list of its neighbors that have not received the message, in $O(l \times n)$, where $1 \leqslant l \leqslant n$ is the number of broadcasting neighbors and n is the total number of neighbors. Consider the set of points

$$\mathcal{P} = \{P_0, P_1, \ldots, P_m\}$$

where P_0 is located at the position of N_A and $P_1, P_2, \ldots, P_m$ are located at the positions of the nodes in $List_{N_A}^{Rec}$. Suppose

$$\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_k\}$$

is the set of points located at the position of the nodes in $List_{N_A}^{NotRec}$. As mentioned earlier, by constructing the Voronoi cell of P_0, we can compute the CPP in $O((m+k)\log m)$ for sets $\mathcal{P}$ and $\mathcal{Q}$. Clearly, the responsibility condition is satisfied if and only if there exists a point $Q \in \mathcal{Q}$ such that

$$\forall P \in \mathcal{P}: \quad \overline{QP_0} \leqslant \overline{QP}.$$

As shown in the proof of Theorem 3.4, l (the number of broadcasting neighbors of N_A) is bounded by a constant. Moreover, we have $k \leqslant n$, $m \leqslant l \times \Delta_G$ and $n \leqslant \Delta_G$, where Δ_G is the maximum node degree of the network. Therefore, the complexity of computing the responsibility condition is

$$O(l \times n) + O((m + k) \log m) = O(\Delta_G \log \Delta_G).$$

Note that the CPP can be solved in $O(k+m)$ in the average case. Therefore, the average complexity of computing the responsibility condition is

$$O(l \times n) + O(m + k) = O(\Delta_G).$$

□

3.4.3 Reducing Bandwidth Requirements

In Algorithm 3.3, each forwarding node piggybacks the list of its 1-hop neighbors in the broadcast packet. In this section, we show how a forwarding node can reduce bandwidth overhead by piggybacking a list of a subset of its 1-hop neighbors.

Definition 3.3 (Representative Set). *Let $\mathcal{P}$ be a set of points $\{P_1, P_2, \ldots, P_n\}$ inside a circle $C_{O,R}$. The set $\mathcal{S} \subseteq \mathcal{P}$ is a representative set for $\mathcal{P}$ if for every point Q outside $C_{O,R}$ (i.e., $\overline{OQ} > R$)*

$$\exists P' \in \mathcal{S} \quad s.t. \quad \forall P \in \mathcal{P}: \quad \overline{P'Q} \leqslant \overline{PQ}.$$

A representative set $\mathcal{S}_{min}$ of $\mathcal{P}$ is called the minimum representative set of $\mathcal{P}$ if $\mathcal{S}_{min}$ has the minimum cardinality among all the representative sets of $\mathcal{P}$.

Definition 3.4 (Representative Neighbor Set). *We say P is the corresponding point of the node N_X if P is located at the position of N_X. Let $\mathcal{P}$ be the set of corresponding points of N_A's neighbors. A subset of N_A's neighbors is called a representative neighbor set if their corresponding point set is a representative set of $\mathcal{P}$. The set of N_A's representative neighbors with the minimum number of nodes is called the minimum representative neighbor set.*

Lemma 3.2. *In Algorithm 3.3, a forwarding node can piggyback a list of its representative neighbors instead of the list of all its neighbors, without affecting the forwarding status of any other node in the network.*

Proof. Let $List_{N_A}^{Rep}$ be a list of representative nodes of N_A. Suppose N_A piggybacks $List_{N_A}^{Rep}$ instead of the list of all its neighbors and $N_B \notin List_{N_A}^{Rep}$ is a neighbor of N_A. The node N_B is required in computing the responsibility condition of N_A's neighbor N_C if N_C has a neighbor N_D that has not received the message and $\overline{BD} < \overline{CD}$. Note that N_D is not a neighbor of N_A since all of N_A's neighbors have received the message. Therefore, N_D is outside $C_{A,R}$, where $C_{A,R}$ is the transmission range of N_A. Based on the definition of representative neighbor set, there exists a node $N_E \in List_{N_A}^{Rep}$ such that

$$\overline{ED} \leqslant \overline{BD} \quad \Longrightarrow \quad \overline{ED} < \overline{CD}.$$

Therefore, for each neighbor of N_A required in computing the responsibility condition of N_C, we can find a node in $List_{N_A}^{Rep}$ that leads to the same result. □

Lemma 3.2 shows that forwarding nodes can save bandwidth by piggybacking the list of their representative neighbors instead of the list of all their neighbors. Clearly, finding the minimum representative neighbor set is equivalent to computing the minimum representative set of $\mathcal{P}$, where $\mathcal{P}$ is the set of points corresponding to the node's neighbors. Let us denote the Voronoi diagram of $\mathcal{P}$ by $\mathrm{Vor}(\mathcal{P})$. Assume that $Cell(P)$ is the set of vertices of the cell of $\mathrm{Vor}(\mathcal{P})$ that corresponds to a point $P \in \mathcal{P}$. Note that $Cell(P)$ includes a vertex v_∞ "at infinity" if the Voronoi cell of P has an infinite edge. Theorem 3.7 shows that the minimum representative set of $\mathcal{P}$ is unique and can be computed in $O(|\mathcal{P}| \log |\mathcal{P}|)$.

Lemma 3.3. *Suppose $\mathcal{P}$ is a set of points inside the circle $C_{O,R}$ and $\mathcal{S}_{min}$ is the minimum representative set of $\mathcal{P}$. We have*

$$P \in \mathcal{S}_{min} \quad \Longleftrightarrow \quad \exists V \in Cell(P): \quad \overline{OV} > R.$$

Proof. Suppose that

$$\forall V \in Cell(P): \quad \overline{OV} \leqslant R.$$

In this case, the entire Voronoi cell of P will be inside the circle $C_{O,R}$. Therefore, for every point Q outside the circle we have

$$\exists P' \in \mathcal{P}: \quad \overline{P'Q} \leqslant \overline{PQ},$$

because Q is outside the Voronoi cell of P. Consequently, P is not required to be in a representative set of $\mathcal{P}$, thus it is not in the minimum representative set of $\mathcal{P}$. Now, suppose that

$$\exists V \in Cell(P): \quad \overline{OV} > R.$$

Since the point P is inside the circle, the line segment PV crosses the circle in a point U. Consider a point Q on the line segment UV such that $Q \neq U$ and $Q \neq V$. Clearly, Q is outside the circle $C_{O,R}$, i.e., $\overline{OQ} > R$. Since the Voronoi cell of P is a convex polygon, the point Q is inside the cell and is not located on any edge of it. Therefore, we have

$$\forall P' \in \mathcal{P} \backslash \{P\}: \quad \overline{PQ} < \overline{P'Q}.$$

Consequently, P must be in every representative set of $\mathcal{P}$ including its minimum representative set. □

Theorem 3.7. *Suppose $\mathcal{P}$ is a set of points inside a circle $C_{O,R}$. The minimum representative set of $\mathcal{P}$ is unique and can be computed in $O(|\mathcal{P}| \log |\mathcal{P}|)$.*

Proof. Based on Lemma 3.3, a point P is in the minimum representative set of $\mathcal{P}$ if and only if

$$\exists V \in Cell(P): \quad \overline{OV} > R.$$

There is a single Voronoi diagram associated with each set of generating points. Therefore, the minimum representative set is unique. From Lemma 3.3, it also follows that the minimum representative set of $\mathcal{P}$ is a subset of every representative set of $\mathcal{P}$. To compute the minimum representative set of $\mathcal{P}$, we first compute the Voronoi diagram Vor($\mathcal{P}$). For each Voronoi cell, we then check whether it has a vertex outside the circle $C_{O,R}$. The Voronoi diagram of $\mathcal{P}$ can be computed in $O(|\mathcal{P}| \log |\mathcal{P}|)$. Moreover, the average number of vertices of a cell is less than six [47]. In other words, the total number of vertex-checking operations is less than $6 \times |\mathcal{P}|$. Therefore, the complexity of computing the representative set is

$$O(|\mathcal{P}| \log |\mathcal{P}|) + O(|\mathcal{P}|) = O(|\mathcal{P}| \log |\mathcal{P}|).$$

□

In the worst case, we have $|\mathcal{S}_{min}| = |\mathcal{P}|$, where $\mathcal{S}_{min}$ is the minimum representative set of $\mathcal{P}$. This situation occurs, for example, when all the

points in $\mathcal{P}$ are located in a line segment or on a circle. However, it can be shown that

$$\lim_{|\mathcal{P}|\to\infty} \frac{E(|\mathcal{S}_{min}|)}{|\mathcal{P}|} = 0,$$

where $E(|\mathcal{S}_{min}|)$ is the expected cardinality of the minimum representative set of $\mathcal{P}$. In order to avoid complex analytical analysis, we used simulation to analyze $E(|\mathcal{S}_{min}|)$. The simulation results are presented in Section 3.6.

Suppose that the forwarding nodes can piggyback a list of their representative neighbors instead of the list of all their neighbors. In this case, Theorem 3.8 shows that a node can simply avoid broadcasting if it is not in the list of the nodes piggybacked in at least one received packet.

Theorem 3.8. *A node N_A can set its status to non-forwarding for a message M if it receives M in a packet that does not list N_A as a representative node.*

Proof. Suppose N_A receives the message from node N_B and that N_A is not in the list of N_B's representative nodes piggybacked in the packet. We prove that N_A's status is non-forwarding based on the responsibility condition. N_A's status is non-forwarding if all of its neighbors have received the message, so we assume that this is not the case. Suppose N_C is a neighbor of N_A that has not received the message. Therefore, N_C is not a neighbor of N_B, hence it is outside $C_{B,R}$, where $C_{B,R}$ is the transmission range of N_B. Since N_A is not in the representative neighbor set of N_B, there exists a node $N_D \neq N_A$ such that N_D is in the representative neighbors set of N_B and

$$\overline{DC} \leqslant \overline{AC}.$$

Note that $\overline{DC} \neq \overline{AC}$, because based on Lemma 3.3, the Voronoi cell of the point A is inside $C_{B,R}$. Therefore,

$$\overline{DC} < \overline{AC}.$$

It follows that N_A is not responsible for any of its neighbors and therefore has a non-forwarding status. □

3.5 Relaxing Some of the System-Model Assumptions

The assumptions made in Section 3.2 are often used in the existing broadcast algorithms in order to model the network. Some of these assumptions are

not practical. For example, a node may obtain its position using the Global Positioning System (GPS). However, the position is not accurate due to several sources of error including positioning error and the limited number of bits used to represent the position. In this section, we discuss how to relax some of the system-model assumptions or replace them with more practical ones in order to improve the practicality of our proposed broadcast algorithm. We show that all of the extensions of the proposed algorithm can provide both full delivery and a constant approximation ratio to the optimal solution under the new assumptions.

3.5.1 Broadcasting Under Uncertain Position Information

As mentioned earlier, the assumption of having the precise position is not practical. In practice, there are several sources of error. For example, the position information obtained by GPS typically includes some errors, which vary between different GPS devices. Another source of error is roundoff error or representation error, which is associated with the fact that a finite number of bits is used to represent the position information. Typically, each node in the network is able to reduce the representation error by assigning a fairly large number of memory bits to represent a number. However, to reduce the bandwidth and power overhead, it is desirable to use as few bits as possible to represent the position information added in the *Hello* messages or the broadcast packets. In this section, we show that a constant number of bits suffices to maintain the main properties of the proposed algorithm.

Let δ be an upper bound on the maximum position error and $\mathcal{E} \geqslant \delta$ an upper bound for maximum error in computing the distance between two points. Suppose that $\mathcal{E}$ is known by all the nodes in the network. We have

$$\overline{AB} - \mathcal{E} \leqslant \Gamma(\overline{AB}) \leqslant \overline{AB} + \mathcal{E}, \tag{3.4}$$

where $\Gamma(\overline{AB})$ is the approximated distance between the points A and B. Algorithm 3.4 shows how to compute the responsibility condition under uncertain position information. Let $List_{N_A}^{Neigh}$ be the list of N_A's neighbors, $List_{N_A}^{NotRec}$ be the list of N_A's neighbor that have not received the message and $List_{N_A}^{Rec}$ be the list of nodes that have received the message, i.e., all of the broadcasting nodes together with their 1-hop neighbors. Algorithm 3.4

returns true if and only if N_A has a neighbor $N_B \in List_{N_A}^{NotRec}$ such that

$$\forall N_C \in List_{N_A}^{Rec}: \quad \Gamma(\overline{AB}) - \mathcal{E} \leqslant \Gamma(\overline{BC}) + \mathcal{E}. \tag{3.5}$$

The output of Algorithm 3.4 determines whether the responsibility condition is satisfied.

Algorithm 3.4 Computing the responsibility condition under uncertain position information

Input: $List_{ID}^{Neigh}$, $List_{ID}^{NotRec}$, $List_{ID}^{Rec}$ and the max error $\mathcal{E}$
Output: true or false

```
1: for i = 1; i ⩽ length(List_ID^NotRec); ++i do
2:    d ← dist(List_ID^NotRec[i], ID)
3:    chk ← true
4:    for j = 1; j ⩽ length(List_ID^Rec); ++j do
5:       if dist(List_ID^NotRec[i], List_ID^Rec[j]) < d − 2E then
6:          chk ← false
7:          break
8:       end if
9:    end for
10:   if (chk) then
11:      return true
12:   end if
13: end for
14: return false
```

Theorem 3.9. *Algorithm 3.3 guarantees full delivery under uncertain position information if it uses Algorithm 3.4 to compute the responsibility condition.*

Proof. The proof is similar to that of Theorem 3.3. The broadcasting will eventually terminate, since each node transmits the message only once. By contradiction, suppose there is a node that has not received the message after termination of broadcasting. Let us consider the set

$$\begin{aligned}\Lambda = \{(N_X, N_Y) | & N_X \text{ and } N_Y \text{ are neighbors,} \\ & N_X \text{ has received the message and} \\ & N_Y \text{ has not received the message}\}\end{aligned}$$

Suppose N_S is the source node and N_T is a node that has not received the message. The network is connected, thus there is a path between N_S and N_T. Clearly, we can find two neighbor nodes N_B and N_A along the path from N_T to N_S such that N_B has not received the message, while N_A has. Consequently, $(N_A, N_B) \in \Lambda$, thus $\Lambda \neq \emptyset$. As a result,

$$\exists (N_{A'}, N_{B'}) \in \Lambda \text{ s.t. } \forall (N_X, N_Y) \in \Lambda : \ \overline{A'B'} \leqslant \overline{XY}. \tag{3.6}$$

Node $N_{B'}$ has not received the message, hence it is not a neighbor of any broadcasting node. Consequently, $N_{A'}$ does not consider $N_{B'}$ as a node that has received the message. Clearly, $N_{A'}$ has not broadcast since $N_{B'}$ has not received the message. Considering the responsibility condition, it follows that there is a node $N_{C'}$ such that $N_{C'}$ has received the message and

$$\Gamma(\overline{B'C'}) + \mathcal{E} < \Gamma(\overline{A'B'}) - \mathcal{E}.$$

Using (3.4) we get

$$\overline{B'C'} < \overline{A'B'}.$$

This result contradicts (3.6), since $(N_{C'}, N_{B'}) \in \Lambda$.

□

Theorem 3.10. *Using Algorithm 3.4 to compute the responsibility condition, the proposed broadcast algorithm (Algorithm 3.3) guarantees a constant approximation ratio to the optimum solution under uncertain position information if* $\mathcal{E} = \frac{R}{8} - \lambda R$, *where* $0 < \lambda \leqslant \frac{1}{8}$ *and* $\frac{1}{\lambda}$ *is bounded by a constant number.*

Proof. We show that the number of broadcasting nodes inside a disk $D_{O,\frac{R}{4}}$ with a radius $\frac{R}{4}$ is bounded by a constant. Using the same approach used in the proof of Theorem 3.4, it then follows that the total number of broadcasting nodes is bounded by a constant factor of that of the optimum solution. Let $D_{O,\frac{R}{4}}$, $D_{O,\frac{3R}{4}}$ and $D_{O,\frac{5R}{4}}$ be three disks centered at O with radii $\frac{R}{4}$, $\frac{3R}{4}$ and $\frac{5R}{4}$, respectively. Suppose k is the minimum number, such that for every set of k points $P_i \in D_{O,\frac{5R}{4}} - D_{O,\frac{3R}{4}}$, $1 \leqslant i \leqslant k$, we have

$$\exists P_i, P_j : \quad i \neq j \quad \text{and} \quad \overline{P_iP_j} \leqslant \lambda R.$$

Note that $\frac{1}{\lambda}$ is bounded by a constant, so the area $D_{O,\frac{5R}{4}} - D_{O,\frac{3R}{4}}$ can be covered with a constant number of disks with radius $\frac{\lambda}{2}R$. If the number of

points inside $D_{O,\frac{5R}{4}} - D_{O,\frac{3R}{4}}$ is greater than the number of covering disks, at least one covering disk will contain more than one point. For two points P_1 and P_2 inside a disk with radius $\frac{\lambda}{2}R$, we have $\overline{P_1P_2} \leqslant \lambda R$. Therefore, k is bounded by a constant (the number of covering disks plus one). We prove that the number of broadcasting nodes inside $D_{O,\frac{R}{4}}$ is less than or equal to k. By contradiction, suppose that there are more than k broadcasting nodes inside $D_{O,\frac{R}{4}}$. Let $N_{A_i} \in D_{O,\frac{R}{4}}$, $0 \leqslant i \leqslant k$ be the first $k+1$ broadcasting nodes ordered chronologically based on their broadcast time. Based on the responsibility condition, for each node N_{A_i} there is a corresponding neighbor N_{B_i} such that $N_{B_i} \notin List^{Rec}_{N_{A_i}}$ and

$$\forall N_C \in List^{Rec}_{N_{A_i}}: \quad \Gamma(\overline{A_iB_i}) - \mathcal{E} \leqslant \Gamma(\overline{B_iC}) + \mathcal{E},$$

where $List^{Rec}_{N_{A_i}}$ is the list of nodes that have received the message based on N_{A_i}'s collected information. Suppose $1 \leqslant i < j \leqslant k$. Nodes N_{A_i} and N_{A_j} are neighbors, because $\overline{A_iA_j} \leqslant \frac{R}{2}$. Recall that N_{A_i} piggybacks the list of all its neighbors within the broadcast packet. Therefore, $N_{B_i} \in List^{Rec}_{N_{A_j}}$, thus $N_{B_i} \neq N_{B_j}$. Note that every node $N_C \in D_{O,\frac{3R}{4}}$ will receive the message after N_{A_0}'s broadcast, because $\overline{A_0C} \leqslant R$. Therefore, N_C is in the list of nodes piggybacked by N_{A_0} in the broadcast packet. Thus

$$\forall 1 \leqslant i \leqslant k: \quad N_C \in List^{Rec}_{N_{A_i}}.$$

It follows that

$$\forall 1 \leqslant i \leqslant k: \quad N_{B_i} \notin D_{O,\frac{3R}{4}}.$$

On the other hand, N_{B_i} is a neighbor of N_{A_i}, thus $N_{B_i} \in D_{O,\frac{5R}{4}}$. Consequently,

$$\forall 1 \leqslant i \leqslant k: \quad N_{B_i} \in D_{O,\frac{5R}{4}} - D_{O,\frac{3R}{4}}.$$

Thus,

$$\forall 1 \leqslant i \leqslant k: \quad \overline{A_iB_i} > \frac{R}{2}. \tag{3.7}$$

There are k different nodes $N_{B_1} \ldots N_{B_k}$ inside $D_{O,\frac{5R}{4}} - D_{O,\frac{3R}{4}}$. Therefore,

$$\exists N_{B_i}, N_{B_j}: \quad i < j \quad \text{and} \quad \overline{B_iB_j} \leqslant \lambda R. \tag{3.8}$$

From (3.4) we have

$$\Gamma(\overline{B_iB_j}) + \mathcal{E} \leqslant \overline{B_iB_j} + 2\mathcal{E}.$$

Thus, using (3.8) we get

$$\Gamma(\overline{B_iB_j}) + \mathcal{E} \leqslant \frac{R}{8} + \mathcal{E}. \tag{3.9}$$

On the other hand, we have $\overline{A_jB_j} > \frac{R}{2}$. Hence,

$$\Gamma(\overline{A_jB_j}) - \mathcal{E} > \frac{R}{2} - 2\mathcal{E}. \tag{3.10}$$

Since $\mathcal{E} \leqslant \frac{R}{8}$, we have

$$\frac{R}{8} + \mathcal{E} \leqslant \frac{R}{2} - 2\mathcal{E}.$$

Thus, based on (3.9) and (3.10) we get

$$\Gamma(\overline{B_iB_j}) + \mathcal{E} < \Gamma(\overline{A_jB_j}) - \mathcal{E}$$

This result is a contradiction, because based on (3.5), N_{A_j} is not responsible for N_{B_j}. □

Example 3.2. *Suppose that $\mathcal{E} \leqslant \frac{R}{9}$. Therefore, we have*

$$\mathcal{E} = \frac{R}{8} - \lambda R,$$

where $\lambda \geqslant \frac{1}{72}$. Note that $\frac{1}{\lambda} (\leqslant 72)$ is bounded by a constant. Thus, based on Theorem 3.10, the proposed broadcast algorithm can guarantee a constant approximation ratio to the optimal solution if the maximum error $\mathcal{E}$ is less than or equal to $\frac{R}{9}$.

As shown in Theorem 3.10, Algorithm 3.3 can guarantee a constant approximation ratio to MCDS if the maximum error is bounded by a factor of the transmission range. As a result, employing the same approach used in the proof of Theorem 3.5, we can show that the message complexity of Algorithm 3.3 under uncertain position information is still $O(N)$, where N is the total number of nodes in the network and each message contains a node *id* and its position. Note that node *ids* can be simply removed from the message without affecting the functionality of the proposed broadcast algorithm. It is because the responsibility condition is based solely on the approximate/precise position information of the nodes.

Suppose that all of the nodes in the network are in an $L \times L$ square area. Clearly, each node requires an infinite number of bits in general to precisely represent its position. However, allowing errors in position information, node position can be represented by a limited number of bits, provided that the size of square area is finite. For example, assume that we allow the maximum position error of αR. As shown in Figure 3.6, the $L \times L$ square area can be partitioned into $\lceil \frac{L}{\sqrt{2}\alpha R} \rceil \times \lceil \frac{L}{\sqrt{2}\alpha R} \rceil$ grid cells of size $\sqrt{2}\alpha R \times \sqrt{2}\alpha R$. Let the tuple (I, J) denote the coordinate of the cell in the grid. For instance, (1,1) indicates the bottom left cell of the gird. Suppose each node uses the coordinate of the cell in which it is located to indicate its position. In this case, the position of the node can be estimated by the position of the center of the cell. Therefore, the number of bits required to represent the approximate position is at most

$$\lceil \lg_2 \left(\lceil \frac{L}{\sqrt{2}\alpha R} \rceil \right) \rceil + \lceil \lg_2 \left(\lceil \frac{L}{\sqrt{2}\alpha R} \rceil \right) \rceil \approx 2 \lg_2 \left(\frac{L}{\alpha R} \right)$$

and the maximum position error is one-half of the cell diagonal, i.e.,

$$\frac{\sqrt{2}(\sqrt{2}\alpha R)}{2} = \alpha R.$$

Note that the number of bits required to represent the approximate position increases as $\frac{L}{R}$ increases.

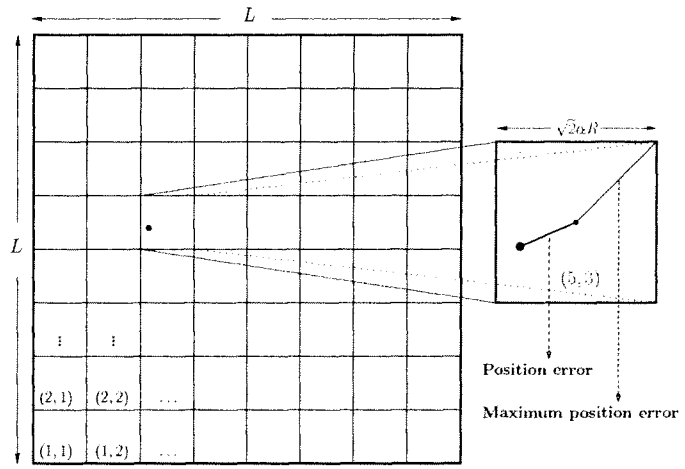

Figure 3.6: Partitioning the network into square cells.

Lemma 3.4. *Suppose that the node N_A is required to send its position with the maximum error αR to its neighbor N_B. In this case, N_A requires only a constant number of bits to represent its position if $\frac{1}{\alpha}$ is bounded by a constant.*

Proof. Suppose (i, j) is the coordinate of the cell in which N_A is located and that the side of each grid cell is $\sqrt{2}\alpha R$. Node N_A can transmit the tuple

$$(i \mod (\lfloor\frac{\sqrt{2}}{\alpha}\rfloor + 2), j \mod (\lfloor\frac{\sqrt{2}}{\alpha}\rfloor + 2))$$

to node N_B. Suppose (i, j) and (i', j') are the coordinates of two different cells such that

$$i \equiv i' \mod (\lfloor\frac{\sqrt{2}}{\alpha}\rfloor + 2),$$

and

$$j \equiv j' \mod (\lfloor\frac{\sqrt{2}}{\alpha}\rfloor + 2).$$

Let P_1 and P_2 be two points inside the cells (i, j) and (i', j'), respectively. It is easy to show that

$$\overline{P_1P_2} \geqslant (\lfloor\frac{\sqrt{2}}{\alpha}\rfloor + 1) \times (\sqrt{2}\alpha R) \implies \overline{P_1P_2} > 2R.$$

It follows that a circle with radius R cannot intersect both cells. Therefore, N_B can uniquely identify the cell using the tuple

$$(i \mod (\lfloor\frac{\sqrt{2}}{\alpha}\rfloor + 2), j \mod (\lfloor\frac{\sqrt{2}}{\alpha}\rfloor + 2)).$$

Consequently, N_A is required to transmit only

$$2\lceil\lg_2\left(\lfloor\frac{\sqrt{2}}{\alpha}\rfloor + 2\right)\rceil \approx 2\lg_2(\frac{1}{\alpha}) + 2$$

bits, which is constant if $\frac{1}{\alpha}$ is bounded by a constant. Node N_B can approximate the position of N_A by taking the position of the center of the cell. □

As mentioned earlier, the message complexity of Algorithm 3.3 is $O(N)$ under uncertain position information if the maximum position error is bounded. Moreover, using Lemma 3.4 , it follows that the size of each message can be

reduced to a constant number of bits by removing the node *id* and using a constant number of bits to represent the approximate position. In [35], the authors proved that every distributed algorithm for constructing a non-trivial CDS has the lower message complexity bound of $\Omega(N \log N)$, where N is the number of nodes and the message size is a constant multiple of the number of bits representing the node *id*s. Although finding a non-trivial CDS and designing a non-trivial broadcast algorithm are closely related problems, our results show that the same message complexity lower bound does not apply for the non-trivial broadcast algorithms.

3.5.2 Relaxing the Homogeneous Network Assumption

In practice, devices may have different transmission ranges. Suppose G' is a graph for which there is a link from N_A to N_B if N_B is in transmission range of N_A. Let G be an undirected graph obtained by removing uni-directional links of the graph G'. We assume that G is connected and define two nodes as neighbors if there is a link between them in G (i.e., they are in transmission range of each other). Note that many wireless medium access control protocols, such as IEEE 802.11, require bi-directional links.

Let us use R_{N_A} to denote a lower bound of N_A's transmission range. Suppose each node includes the lower bound of its transmission range in the *Hello* message. Moreover, assume that the forwarding nodes include the lower bounds of their neighbors' transmission ranges in the broadcast packets. The responsibility condition of node N_A is satisfied if N_A has a neighbor N_B such that N_B has not received the message, and for every node N_C that has received the message

$$\overline{AB} \leqslant \overline{CB} \quad \text{or} \quad \overline{CB} > R_{N_C}.$$

Using a similar approach as that used in the proof of Theorem 3.3, we can show that Algorithm 3.3 guarantees full delivery in a heterogeneous network if it employs the new responsibility condition.

Suppose R_{min} is a lower bound for the transmission ranges of all the nodes in the network and R_{max} is an upper bound. The following theorem shows that the proposed broadcast algorithm can still guarantee a good bound on the number of forwarding nodes in a heterogeneous network if $\frac{R_{max}}{R_{min}}$ is bounded by a constant.

Theorem 3.11. *The proposed broadcast algorithm can guarantee a constant approximation ratio to the optimal solution in a heterogeneous network if* $\frac{R_{max}}{R_{min}}$ *is bounded by a constant.*

Proof. Let $D_{O,\frac{R_{min}}{4}}$, $D_{O,\frac{3R_{min}}{4}}$ and $D_{O,\frac{R_{min}}{4}+R_{max}}$ be three disks centered at O with radii $\frac{R_{min}}{4}$, $\frac{3R_{min}}{4}$ and $\frac{R_{min}}{4}+R_{max}$, respectively. Suppose k is the minimum number such that for every set of k points $P_i \in D_{O,\frac{R_{min}}{4}+R_{max}} - D_{O,\frac{3R_{min}}{4}}$, $1 \leqslant i \leqslant k$, we have

$$\exists P_i, P_j: \quad i \neq j \quad \text{and} \quad \overline{P_iP_j} \leqslant \frac{R_{min}}{2}.$$

Note that $\frac{R_{max}}{R_{min}}$ is bounded by a constant, so the area $D_{O,\frac{R_{min}}{4}+R_{max}} - D_{O,\frac{3R_{min}}{4}}$ can be covered with a constant number of disks with radius $\frac{R_{min}}{4}$. If the number of points inside $D_{O,\frac{R_{min}}{4}+R_{max}} - D_{O,\frac{3R_{min}}{4}}$ is greater than the number of covering disks, at least one covering disk will contain more than one point. For two points P_1 and P_2 inside a disk with radius $\frac{R_{min}}{4}$, we have $\overline{P_1P_2} \leqslant \frac{R_{min}}{2}$. Therefore, k is bounded by a constant (the number of covering disks plus one). Using a technique similar to that used in the proof of Lemma 3.1, we can show that the number of broadcasting nodes inside $D_{O,\frac{R_{min}}{4}}$ is less than or equal to k. Since $\frac{R_{max}}{R_{min}}$ is bounded by a constant, a disk with radius R_{max} can be covered with a constant number of disks with radii R_{min}. It follows that the total number of broadcasting nodes is bounded by a constant factor of that of the optimum solution (refer to the proof of Theorem 3.4).

□

3.5.3 Broadcasting in Three-Dimensions

We assumed in Section 3.2 that the nodes are placed in a 2-D plane. In general, the nodes can be located in 3-D space, as the network area may not be perfectly flat. In this case we assume that the nodes are provided with their position in 3-D. Interestingly, all the properties of the proposed broadcast algorithm are preserved in 3-D. Assuming $\overline{AB}$ as the Euclidean distance of two points in 3-D, we can use the proof of Theorem 3.3 to show that Algorithm 3.3 guarantees full delivery in 3-D space. Note that the transmission range of a node in 3-D is a sphere. Replacing circles by spheres in the proofs of Lemma 3.1 and Theorem 3.4, we can similarly argue that Algorithm 3.3

guarantees a constant approximation ratio to the optimal solution in 3-D. Finally, using a technique similar to that used for 2-D, we can show that, in 3-D, the proposed algorithm not only preserves its main properties under uncertain position information (provided that the maximum error is bounded) but also can benefit from the proposed bandwidth-overhead reduction technique introduced in Section 3.4.3.

3.6 Simulation

3.6.1 Reducing Bandwidth Overhead

As shown in Section 3.4.3, a forwarding node can piggyback the list of a subset of its neighbors (its representative neighbor set) instead of all of them without violating the status (forwarding/non-forwarding) of any other node in the network. To avoid the complexity of mathematical analysis, we used simulation to find the average cardinality of the minimum representative set. For a given number of neighbors $1 \leqslant n \leqslant 500$, we randomly put n points inside a unit circle and computed the minimum representative set $\mathcal{S}_{min}$. To get the average cardinality $E(|\mathcal{S}_{min}|)$, we ran the simulation 10^4 times for each given n. Figure 3.7 shows the ratio $\frac{E(|\mathcal{S}_{min}|)}{n} \times 100$. As shown in Figure 3.7, the ratio $\frac{E(|\mathcal{S}_{min}|)}{n}$ decreases as the number of neighbors increases. For instance, the minimum number of representative nodes is on average less than 50% of total number of nodes when $n \geqslant 33$.

3.6.2 Performance of the Proposed Broadcast Algorithm

To verify the theoretical results, we used the ns-2 simulator to compute the average number of forwarding nodes. In ns-2, the total size of the data packet was fixed to 256 bytes and the bandwidth of the wireless channel was set to 2 Mb/s. In each simulation run, we uniformly distributed N nodes in a $1000 \times 1000\, m^2$ square area. A randomly generated topology was discarded if it leaded to a disconnected network. Only one broadcast was initiated in each simulation run by a randomly selected node. Table 3.1 summarizes some of the parameters used in ns-2.

Figure 3.8 illustrates an instance of using the proposed algorithm for the case where R = 300m, N = 400 and nodes are placed in a square area of

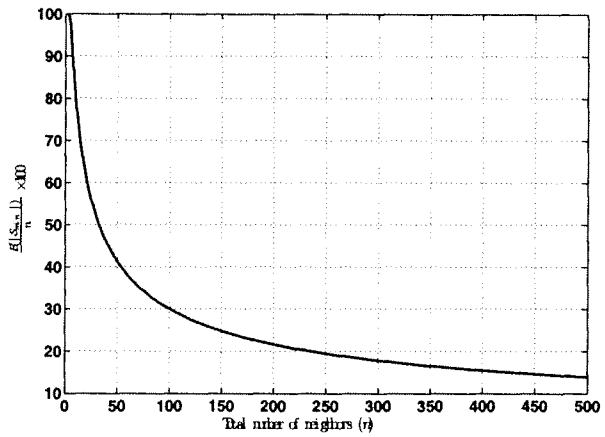

Figure 3.7: The ratio $\frac{E(|S_{min}|)}{n} \times 100$ against the total number of neighbors.

$1000 \times 1000\text{m}^2$. As shown in this figure, only 10 nodes (represented by stars) among 400 nodes broadcast the message. Figures 3.9 and 3.10 show the ratio of the number of broadcasting nodes over the total number of nodes. To get the results shown in Figure 3.9, we varied the transmission range from 50 to 300 meters and fixed the total number of nodes to 1000. Figure 3.10 shows the result of the experiment for which the transmission range was fixed to 250 meters and the total number of nodes varied from 25 to 1000. We compared the performance of our proposed algorithm with that of the broadcast algorithm proposed in [37]. In [37], Liu *et al.* showed that the number of broadcasting nodes using their proposed flooding algorithm is significantly lower than that of previous notable broadcast algorithms [48, 49]. We also considered the ratio-8 approximation of MCDS [35] as a benchmark. As shown in Figures 3.9 and 3.10, our proposed broadcast algorithm can significantly reduce the number of forwarding nodes. Moreover, it slightly outperforms the ratio-8 approximation. In Chapter 2, we proved that using a weaker version of the proposed broadcast algorithm the probability of two neighbor nodes transmitting the same message exponentially decreases when the distance between them decreases or when the node density increases. This property can further justify why the proposed broadcast algorithm can significantly reduce the number of transmissions in the network.

Another metric evaluated using simulation was the algorithm delay, which we define as the time between the first and the last transmission in a single

message broadcast. Figure 3.11 shows the average delay of our proposed algorithm and compares it with that of Liu's broadcast algorithm. As shown in this figure, the average delay of our broadcast algorithm is 80% to 50% of that of Liu's algorithm when the total number of nodes varies from 50 to 1000.

We also evaluated the performance of the proposed algorithm when there is uncertainty in position information. We set the transmission range to $R = 250$ meters and fixed the maximum position error to αR, where $0.01 \leqslant \alpha \leqslant 0.15$. Figure 3.12 shows the average number of broadcasting nodes for a given α when the number of nodes varies within $25 - 1000$. As shown in Figure 3.12, the performance of the broadcast algorithm slightly degrades as the maximum position error increases. Finally, we evaluated the performance of the proposed algorithm for the case where nodes can have different transmission ranges. In the simulation, we randomly placed $25 \leqslant N \leqslant 1000$ nodes in the network. The transmission range of each node was selected randomly from the interval $[R_{min}, 250]$, where $R_{min} < 250$ is the minimum transmission range. The value of R_{min} varied within 150 to 250 meters. As shown in Figure 3.13, the simulation results indicate that the performance of the proposed broadcast algorithm slightly decreases as R_{min} decreases. Based on the simulation results shown in Figure 3.9, this effect is expected as the number of transmissions increases when the transmission range decreases.

Parameter	**Value**
Simulator	ns-2 (version 2.27)
MAC Layer	IEEE 802.11
Propagation Model	two-ray ground
Packet Size	256 bytes
Bandwidth	2 Mb/sec
Size of Square Area	$1000 \times 1000\text{m}^2$
Transmission Range	50–300m
Number of Nodes	25–1000

Table 3.1: Simulation parameters

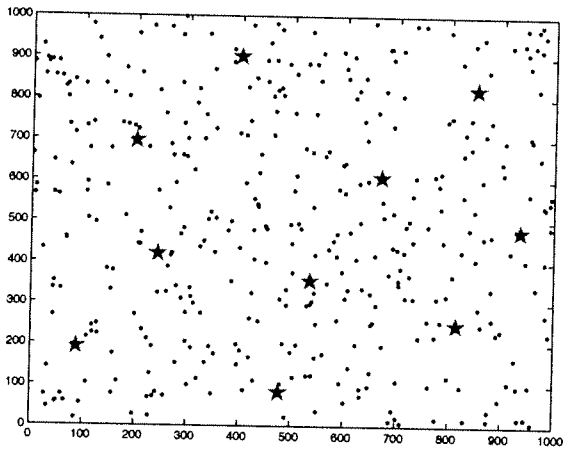

Figure 3.8: Broadcasting nodes in a $10^3 \times 10^3 m^2$ square area with 400 nodes.

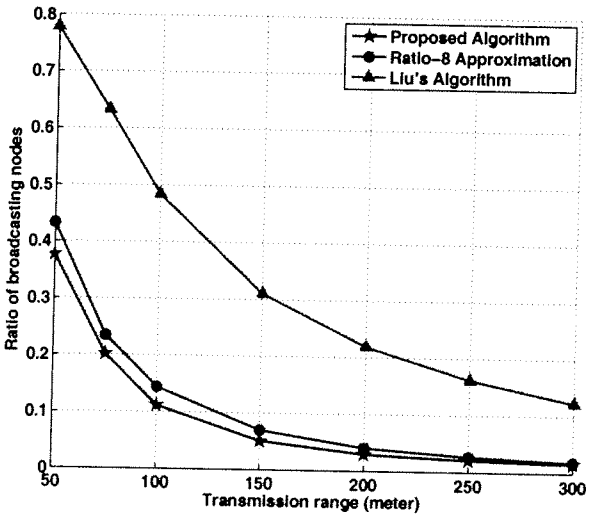

Figure 3.9: Ratio of broadcasting nodes vs. transmission range.

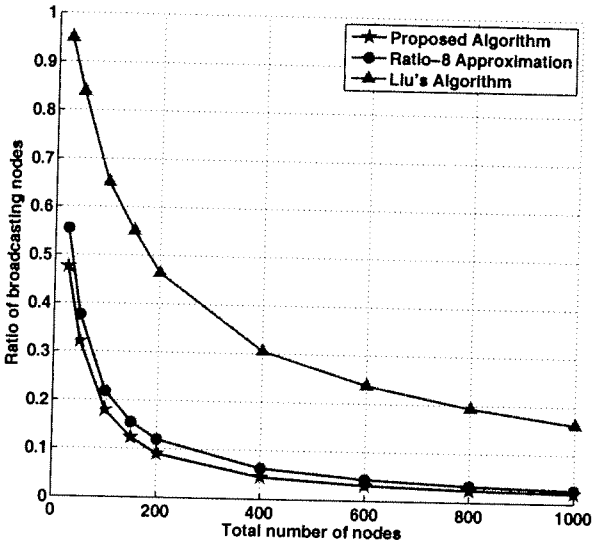

Figure 3.10: Ratio of broadcasting nodes vs. total number of nodes.

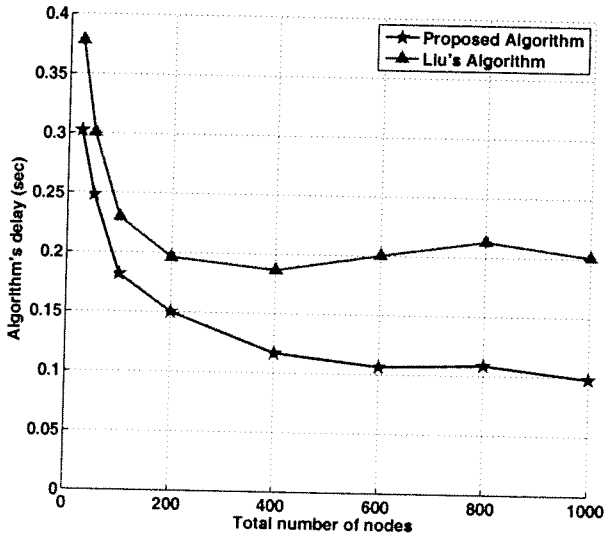

Figure 3.11: Average delay vs. total number nodes.

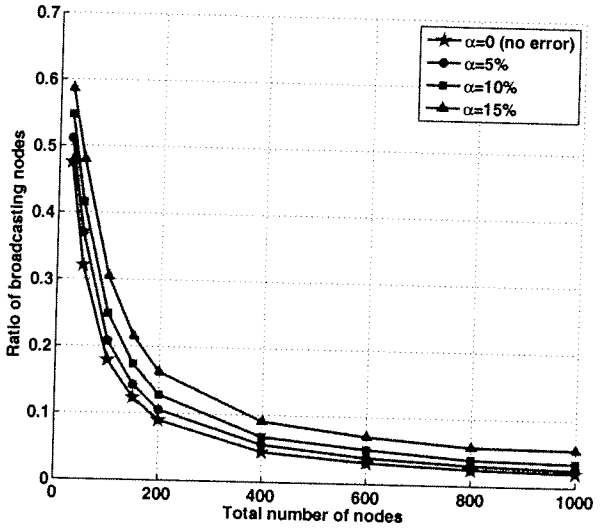

Figure 3.12: Ratio of broadcasting nodes of the proposed algorithm with uncertain position information.

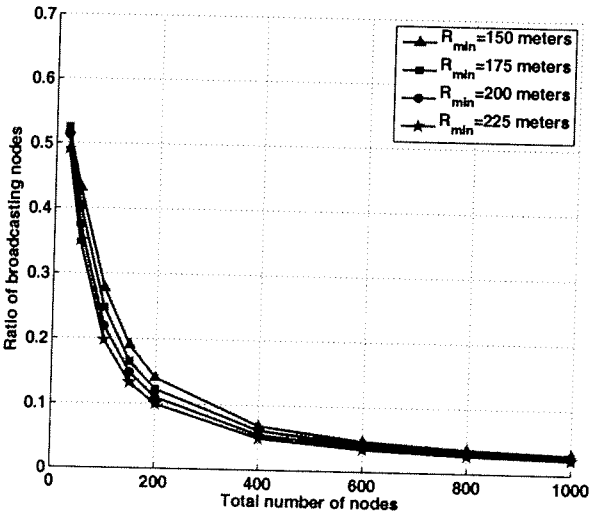

Figure 3.13: Ratio of broadcasting nodes of the proposed algorithm in a network with nodes having different transmission ranges.

3.7 Summary

We considered two general structures typically used in broadcast algorithms based on 1-hop neighbor information. We showed that a broadcast algorithm cannot guarantee both full delivery and a good bound on the number of transmissions if it uses any of these structures. It is commonly believed that a localized broadcast algorithm is not able to guarantee a good bound on the number of transmissions. We presented a localized broadcast algorithm based on 1-hop information and proved that it guarantees both full delivery and a constant approximation ratio to the optimal solution. The proposed broadcast algorithm has low message and computational complexities. In fact, we proved that the message complexity of the proposed algorithm is less than the lower message complexity bound of finding a non-trivial CDS and that its computational complexity is nearly optimal. Moreover, we proposed a technique to further reduce the bandwidth overhead. We then relaxed several system-model assumptions, or replaced them with practical ones, to improve the practicality of the broadcast algorithm. Finally, we verified the analytical results using simulation.

Bibliography

[33] M. Garey and D. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness.* New York, NY, USA: W. H. Freeman & Co., 1990.

[34] B. Clark, C. Colbourn, and D. Johnson, "Unit disk graphs," *Discrete Mathematics*, vol. 86, pp. 165–177, 1990.

[35] P. Wan, K. Alzoubi, and O. Frieder, "Distributed construction of connected dominating set in wireless ad hoc networks," *In Proc. of IEEE INFOCOM*, 2002.

[36] S. Funke, A. Kesselman, U. Meyer, and M. Segal, "A simple improved distributed algorithm for minimum CDS in unit disk graphs," *ACM Trans. Sensor Networks (TOSN)*, vol. 2, no. 3, pp. 444–453, 2006.

[37] H. Liu, P. Wan, X. Jia, X. Liu, and F. Yao, "Efficient flooding scheme based on 1-hop information in mobile ad hoc networks," *In Proc. of IEEE INFOCOM*, 2006.

[38] Y. Tseng, S. Ni, and E. Shih, "Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc networks," *In Proc. International Conference on Distributed Computing Systems (ICDCS)*, pp. 481–488, 2001.

[39] Y. Sasson, D. Cavin, and A. Schiper, "Probabilistic broadcast for flooding in wireless mobile ad hoc networks," *In Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1124–1130, 2003.

[40] P. Kyasanur, R. Choudhury, and I. Gupta, "Smart gossip: An adaptive gossip-based broadcasting service for sensor networks," *In Proc. IEEE International Conference on Mobile Adhoc and Sensor Sysetems (MASS)*, pp. 91–100, 2006.

[41] J. Wu, W. Lou, and F. Dai, "Extended multipoint relays to determine connected dominating sets in manets," *IEEE Trans. on Computers*, vol. 55, no. 3, pp. 334–347, 2006.

[42] G. Călinescu, I. Mandoiu, P.-J. Wan, and A. Zelikovsky, "Selecting forwarding neighbors in wireless ad hoc networks," *Mobile Networks and Applications*, vol. 9, no. 2, pp. 101–111, 2004.

[43] J. Wu and F. Dai, "Broadcasting in ad hoc networks based on self-pruning," *In Proc. of IEEE INFOCOM*, 2003.

[44] W. Peng and X. Lu, "On the reduction of broadcast redundancy in mobile ad hoc networks," *In Proc. ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pp. 129–130, 2000.

[45] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 13, pp. 14–25, 2002.

[46] J. Wu and W. Lou, "Forward-node-set-based broadcast in clustered mobile ad hoc networks," *Wireless Communications and Mobile Computing*, vol. 3, no. 2, pp. 155–173, 2003.

[47] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications,* 2^{nd} *ed.* New York: Springer-Verlag, 2000.

[48] Y. Cai, K. Hua, and A. Phillips, "Leveraging 1-hop neighborhood knowledge for efficient flooding in wireless ad hoc networks," *In Proc. IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pp. 347–354, 2005.

[49] J. Wu and H. Li, "On calculating connected dominating set for efficient routing in ad hoc wireless networks," *In Proc. International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DialM)*, pp. 7–14, 1999.

Chapter 4

Wormhole Attack in Wireless Ad Hoc Networks: Analysis and Countermeasure

4.1 Introduction

The *wormhole attack* [50] is one of the most severe security attacks encountered in wireless ad hoc networks. It can significantly disrupt the communications across the network, is hard to detect and can be implemented without having a cryptography key or knowing the network routing protocol. In a wormhole attack, the attacker receives packets at one location in the network, "tunnels" them to another location and replays them there. A single malicious node can launch this attack by, for example, broadcasting the route requests at a high power level. The wormhole attack is much more powerful if it is launched by more than one attacker. In this case, attackers can tunnel the packets to each other by simply encapsulating them or by using an out-of-band link. Once a wormhole is established, malicious nodes can use it for traffic analysis or to make a Denial-of-Service (DoS) attack by dropping certain data or control packets.

The wormhole attack can be launched in two different modes. In the *hidden mode*, the attackers do not use their identities so they remain hidden from the legitimate nodes. In fact, the attackers act as two simple transceivers that capture messages at one end of the wormhole and replicate them at the other end. In this way, they can create a *virtual link* between two far-off nodes by, for example, "tunneling" their *Hello* messages. The existing wormhole detection schemes [50, 51, 52] typically consider this mode. Clearly, the

A version of this chapter has been accepted for publication. M. Khabbazian, H. Mercier and V.K. Bhargava, Severity Analysis and Countermeasures for the Wormhole Attack in Wireless Ad Hoc Networks, *IEEE Transactions on Wireless Communications*, 2008.

attackers require no cryptographic keys to launch the wormhole attack in the hidden mode.

In the *participation mode*, it is assumed that the attackers possess valid cryptographic keys that can be used to launch a more powerful attack. In this mode, the attackers make no virtual links between the legitimate nodes. Instead, they participate in the routing as legitimate nodes and use the wormhole to deliver the packets sooner and/or with a smaller number of hops. As in the hidden mode, the attackers can drop data packets after being included in the route between the source and the destination.

In this chapter, we analyze the effect of the wormhole attack on shortest-path routing protocols. When the nodes are uniformly distributed, we show that strategic placement of a single wormhole can disrupt on average 32% of the communications across the network. However, we show that ($n \geqslant 2$) attackers cannot disrupt on average more than $(1 - \frac{1}{n})$ of all of the communications. We also analyze the effect of the wormhole attack in which the malicious nodes target a specific node (such as the sink node in a wireless sensor network). In this scenario, we show that two attackers can disrupt/control 30% to 90% of all communications between the target node and the other nodes in the network. We explain how three malicious nodes can disrupt/control almost all of the communications independent of target node location in the network. All analytical results are further confirmed by simulation.

The wormhole attack may have different effects on different network topologies. We show how to compute the maximum effect of the wormhole attack on a given network topology. We then evaluate the maximum effect of the wormhole attack on grid network topologies. For these networks, we show that the wormhole attack can disrupt/control about 40% to 50% of all communications if the attackers strategically place the wormhole.

Finally, we propose a timing-based countermeasure against the wormhole attack that is an improvement over existing schemes based on timing analysis. Using the proposed solution, the nodes do not need to have synchronized clocks, and are not required to predict the sending time or to be capable of fast switching between the receive and send modes. Moreover, the nodes do not need to communicate with all their neighbors one-to-one. We show that the proposed countermeasure can be used together with one of the recently proposed countermeasures in order to improve the probability of detection.

The rest of the chapter is organized as follows. In Section 4.2, we present and categorize related work on the subject. In Section 4.3, we introduce a

new analytic model to measure the effect of the wormhole attack and present analytical and simulation results. In Section 4.4, we show how to compute the maximum effect of the wormhole attack on a generic network. We propose a countermeasure based on timing analysis and compare it with the existing timing-based solutions in Section 4.5. A summary of this chapter is presented in Section 4.6.

4.2 Related Work

The existing countermeasures against the wormhole attack can be divided into proactive and reactive countermeasures. Proactive countermeasures attempt to prevent wormhole formation, typically through specialized hardware used to achieve accurate time synchronization or time measurement, or to transmit maximum power in a particular direction. Among proactive countermeasures, timing-based solutions attempt to restrict the maximum distance between two neighbors by computing the packet travel time. For example, in [50], the authors introduced packet leashes as a countermeasure against the wormhole attack. In their method, the sender adds the transmission time to the packet in order to restrict the packet's maximum transmission distance. In [51], the authors showed how to estimate the distance without having a tight synchronization between the nodes. In this scheme, each node can estimate the distance to another node by sending a challenge bit and receiving its instant response. A similar approach was used in [53] for secure single-hop pairwise time synchronization. In practice, all of the aforementioned timing-based methods suffer from some of the following shortcomings.

- The nodes require tightly synchronized clocks;
- each node has to be capable of fast switching between the receive and send modes;
- each node needs one-to-one communication with all its neighbors;
- each node requires predicting the sending time and computing signature while having to timestamp the message with its transmission time.

In Section 4.5, we show that our proposed timing-based countermeasure does not have any of these shortcomings, thus is more suitable for practical implementation of solutions based on packet travel time measurements.

Location information can also be used in packet leashes in order to defend against the wormhole attack [50]. A practical proactive method based on location information is proposed in [54]. In [55], a small fraction of the nodes called guards have access to location information. These nodes monitor local traffic looking for a wormhole. Directional antennas can also be used to mitigate the wormhole attack [52]. In this approach, the transmitter shares its directional information with the receiver. The receiver can prevent the wormhole endpoint from masquerading as a false neighbor by detecting the direction of the arrived signal and comparing it with the directional information provided by the sender and by finding a correctly positioned verifier node. A wormhole may also be avoided by extracting the sending device's radio fingerprint and verifying whether the radio signal originated from a legitimate device [56].

There are also some proactive countermeasures which do not rely on any specialized hardware. For example, recent work [57] explains how to use the local connectivity information to detect wormhole attack. We will show in this chapter that their approach does not work if the malicious nodes selectively forward the *Hello* messages. A typical drawback of most of the aforementioned proactive schemes (except that proposed in [57]) is that they cannot detect a wormhole running in the participation mode. This is due to the fact that they often attempt to prevent wormholes between two legitimate nodes; however, in the participation mode, the wormhole is formed between two malicious nodes participating in the routing. Note that in this mode, all the links are valid except the one (the wormhole) between the two attackers.

Reactive countermeasures, on the other hand, do not prevent the wormhole formation. For example, the proposed source routing protocols in [58] and [59] consider the wormhole as a valid link and avoid it only if it exhibits some malicious behavior like modifying or dropping packets. This is achieved using some basic mechanisms such as packet authentication and destination acknowledgment. Unlike proactive schemes, the existing reactive countermeasures are not able to avoid the wormhole if it is employed for an invisible (passive) attack such as a traffic analysis attack. Therefore, some other techniques such as anonymous communication have to be used to defend against passive attacks.

4.3 Wormhole Attack Analysis

The wormhole attack can severely affect routing protocols based on shortest delay and shortest path by delivering packets faster and with a smaller number of hops, respectively. The common belief is that the wormhole attack can prevent nodes from discovering other nodes that are more than two hops away when it is used against on-demand routing protocols [50, 60]. We carefully analyze the effect of the wormhole attack on shortest-path routing protocols, with the objective of drawing a more precise picture of the average or maximum potential damage of wormhole attack. Throughout this section, we assume that the attackers use the same type of antenna as other nodes in the network and have the advantage of having direct links to each other. We also assume that the network is static. In mobile ad hoc networks, the number of affected communications varies over time. Moreover, a particular communication can be controlled by the attackers at a certain period of time and can become out of their control at other periods. However, a network can be seen as a sequence of many different static networks obtained by sampling the network over a given time interval. Clearly, the number of samples increases as the time interval becomes larger. Therefore, as the time interval becomes larger, the average number of affected communications over the interval will approach the average number of affected communications in static networks.

In this section, we first show how to approximate the length (number of hops) of the shortest path between two nodes in a homogeneous dense network. We then use this approximation to analytically evaluate the average effect of the wormhole attack for two different scenarios. In the first scenario, the main objective of the malicious nodes is to disrupt/control as many communications as possible in the network. We show that on average every node is still able to communicate with at least 50% of all the other nodes across the network if two malicious nodes launch a wormhole attack. However, the analysis shows that two attackers can disrupt/control about one-third of all communications if they strategically place the wormhole. In the second scenario, two malicious nodes select and attack a *target node* such as the sink node in a wireless sensor network. The main objective of the attackers in this scenario is to disrupt/control the communications between the target node and all other nodes in the network. For this scenario, we show that the attackers can control most of the communications if the target node is around the boundary of the network.

4.3.1 Approximating the Length of the Shortest Path

Consider a network consisting of a large number of nodes distributed uniformly with density δ inside a disk of radius R. Suppose each node is equipped with an omni-directional antenna with wireless transmission range of T. We assume that there is a link between two nodes if their distance is less than or equal to T. In this case, the network topology can be represented by a unit disk graph.

Let us define $d_{S,D}$ (or $\overline{SD}$) as the distance between nodes S and D, and $N_{S,D}$ as the length (#hops) of the shortest path between them. Clearly, we have

$$N_{S,D} \geqslant \frac{d_{S,D}}{T}. \tag{4.1}$$

Lemma 4.1. *In a network with limited diameter and high node density, with high probability we have*

$$N_{S,D} \leqslant 2\frac{d_{S,D}}{T}.$$

Proof. The number of nodes in a region $\mathcal{R}$ with area $\Delta_{\mathcal{R}}$ follows a Poisson distribution [61], since they are uniformly and independently distributed. It follows that

$$\mathrm{P}(\mathcal{R} \text{ contains } k \text{ nodes}) = e^{-\delta\Delta_{\mathcal{R}}}\frac{(\delta\Delta_{\mathcal{R}})^k}{k!}, \tag{4.2}$$

where δ is the density of nodes inside the network. If $d_{S,D} \geqslant \frac{T}{2}$, then there are $t = \lfloor 2\frac{d_{S,D}}{T} \rfloor - 1$ disks with radius $\frac{T}{4}$ and origins at distances $d_i = \frac{T}{2}i + \frac{T}{4}$, $1 \leqslant i \leqslant t$, from S on the line going through S and D. This is illustrated in Figure 4.1. Using (4.2), we get

$$\mathrm{P}(\text{at least one node in each disk}) = (1-\mathrm{P}(\text{no node in a disk}))^t = (1-e^{-\delta(\pi(\frac{T}{4})^2)})^t.$$

Clearly, the distance between two nodes in adjacent disks is at most T. Therefore, there is a path of length $t+1 = \lfloor 2\frac{d_{S,D}}{T} \rfloor$ with probability at least $(1 - e^{-\delta(\pi(\frac{T}{4})^2)})^t$. Consequently, we have $N_{S,D} \leqslant \lfloor 2\frac{d_{S,D}}{T} \rfloor \leqslant 2\frac{d_{S,D}}{T}$ with high probability when

$$e^{-\delta(\pi(\frac{T}{4})^2))}t \ll 1 \qquad \text{or} \qquad \delta \gg \frac{\ln(\lfloor 2\frac{d_{S,D}}{T} \rfloor - 1)}{\pi(\frac{T}{4})^2},$$

where $d_{S,D} \leqslant 2R$. Note that the assumption of uniform distribution is not a necessary condition. In other words, the same result can be obtained if

$P_0 \ll \frac{T}{4R}$, where P_0 is the probability of having no node in a disk with radius $\frac{T}{4}$ and with the origin inside the network. □

From (4.1) and Lemma 4.1, we observe that $N_{S,D}$ is a function of $\frac{d_{S,D}}{T}$ and can be approximated by

$$N_{S,D} = \beta \frac{d_{S,D}}{T}, \tag{4.3}$$

where $1 \leqslant \beta \leqslant 2$.

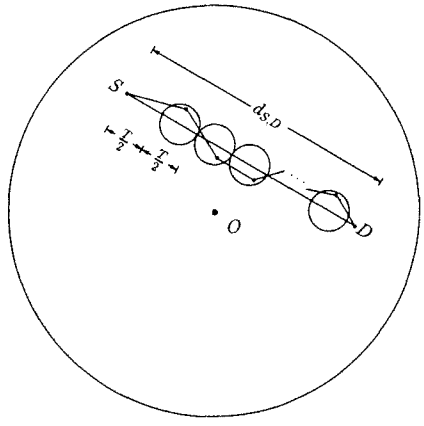

Figure 4.1: Finding a path between S and D.

4.3.2 Targeting all of the nodes in the Network

As mentioned earlier, in this scenario, the malicious nodes target all of the nodes in the network in an attempt to disrupt/control as many communications as possible. Suppose that the wormhole attack is launched in the hidden mode by two malicious nodes M_1 and M_2. Thus, there is a virtual link between every pair of nodes locating in the transmission ranges of M_1 and M_2. We assume that the malicious nodes can control the communication between two nodes S and D if and only if the shortest path between them includes a virtual link. In this case, the communication between nodes S and D goes through the malicious nodes M_1 and M_2, thus it can be disrupted or controlled by them. The following lemma shows how to model the effect of the wormhole attack using (4.3).

Lemma 4.2. *The attackers M_1 and M_2 can control the communication between the nodes S and D if*

$$\min\left(d_{S,M_1} + d_{D,M_2}, d_{S,M_2} + d_{D,M_1}\right) \leqslant d_{S,D}.$$

Proof. The path through M_1 and M_2 (or M_2 and M_1) looks shorter than the actual shortest path between S and D if and only if

$$\min\left(N_{S,M_1} + N_{D,M_2} - 1, N_{S,M_2} + N_{D,M_1} - 1\right) < N_{S,D}.$$

Using (4.3) we get

$$\min\left(\beta\frac{d_{S,M_1}}{T} + \beta\frac{d_{D,M_2}}{T}, \beta\frac{d_{S,M_2}}{T} + \beta\frac{d_{D,M_1}}{T}\right) \leqslant \beta\frac{d_{S,D}}{T},$$

thus

$$\min\left(d_{S,M_1} + d_{D,M_2}, d_{S,M_2} + d_{D,M_1}\right) \leqslant d_{S,D}.$$

When the malicious nodes are on the path, they can analyze the traffic or disrupt the communication by, for example, dropping the route reply or data packets. □

Let us assume that the malicious nodes M_1 and M_2 are located at the coordinates $(-m, 0)$ and $(m, 0)$, respectively. As shown in Figure 4.2 , the perpendicular bisector l of the line segment M_1M_2 partitions the network into two regions $\mathcal{R}_1$ and $\mathcal{R}_2$ containing M_1 and M_2, respectively.

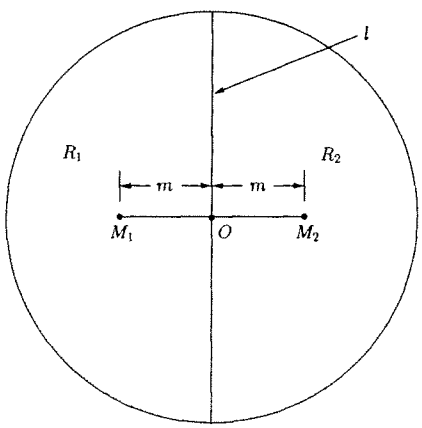

Figure 4.2: Partitioning the network into regions $\mathcal{R}_1$ and $\mathcal{R}_2$.

Lemma 4.3. *Let S and D be two nodes both inside $\mathcal{R}_1$ or $\mathcal{R}_2$. We have*

$$\min\left(d_{S,M_1} + d_{D,M_2}, d_{S,M_2} + d_{D,M_1}\right) > d_{S,D}.$$

Proof. Without loss of generality, we can assume that both S and D are in $\mathcal{R}_1$. Hence,

$$(d_{S,M_2} > d_{S,M_1}) \text{ and } (d_{D,M_2} > d_{D,M_1}).$$

Therefore, using the triangle inequality, we get

$$\min\left(d_{S,M_1} + d_{D,M_2}, d_{S,M_2} + d_{D,M_1}\right) > d_{S,M_1} + d_{D,M_1} \geqslant d_{S,D}.$$

□

From Lemmas 4.2 and 4.3 it follows that two nodes in the same region are able to communicate. The total number of possible communications between $m \gg 1$ nodes in a region $\mathcal{R}$ can be approximated as

$$\binom{m}{2} \sim \frac{m^2}{2}.$$

Let m_1 and m_2 be the number of nodes in region $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively. The attackers cannot disrupt on average more than 50% of all of the communications across the network, because

$$\frac{\binom{m_1}{2} + \binom{m_2}{2}}{\binom{m_1+m_2}{2}} \sim \frac{m_1^2 + m_2^2}{(m_1 + m_2)^2} \geqslant 0.5.$$

Definition 4.1. *A region is called unreachable for a node S if it is inside the network and if for any node D in the region*

$$\min\left(d_{S,M_1} + d_{D,M_2}, d_{S,M_2} + d_{D,M_1}\right) \leqslant d_{S,D}.$$

We denote the largest unreachable region for a node S as U_s.

From Lemma 4.2, it follows that the wormhole attack can control/disrupt the communications between a node S and any node in U_s. The following theorem gives more precise information regarding the severity of a wormhole attack initiated by two malicious nodes M_1 and M_2.

Theorem 4.1. *Let S be a randomly selected node in $\mathcal{R}_1$ and Δ_{U_s} be the area of U_s. We have*

$$\Delta_{U_s} = \frac{1}{2}\int_{\theta_{-1}}^{\theta_1} (g^2(\theta) - f^2(\theta))d\theta,$$

where

$$f(\theta) = \frac{d_{S,M_2}^2 - d_{S,M_1}^2}{2(d_{S,M_2}\cos(\theta) - d_{S,M1})},$$

$$g(\theta) = d_{S,M_1}\cos(\theta+\gamma) + \sqrt{R^2 - d_{S,M_1}^2\sin(\theta+\gamma)},$$

$\gamma = \angle OSM_2$, O is the origin of the network and the values θ_1 and θ_{-1} are the roots of the equations $(g(\theta) - f(\theta) = 0)$ and $(g(-\theta) - f(\theta) = 0)$, respectively.

Proof. Let D be a node in U_s. According to Definition 4.1,

$$\min\left(d_{S,M_1} + d_{D,M_2}, d_{S,M_2} + d_{D,M_1}\right) \leqslant d_{S,D}. \tag{4.4}$$

Since the node S is in $\mathcal{R}_1$, it follows from Lemma 4.3 that the node D is not in $\mathcal{R}_1$. Therefore,

$$\min\left(d_{S,M_1} + d_{D,M_2}, d_{S,M_2} + d_{D,M_1}\right) = d_{S,M_1} + d_{D,M_2}. \tag{4.5}$$

From (4.4) and (4.5) it follows that

$$d_{S,M_1} \leqslant d_{S,D} - d_{D,M_2}.$$

The equation $d_{S,M_1} = d_{S,D} - d_{D,M_2}$ defines a branch of the hyperbola with foci S and M_2. Therefore, as shown in Figure 4.3, U_S is a region between the network boundary (a circle of radius R) and the branch of the hyperbola. Let us consider S as the origin and SM_2 as the x-axis of a polar coordinate system. Using the cosine rule for the triangles SM_2E and SOF, we can compute $f(\theta)$ and $g(\theta)$, respectively. The area Δ_{U_s} can then be computed with

$$\Delta_{U_s} = \int_{\theta_{-1}}^{\theta_1}\int_{f(\theta)}^{g(\theta)} r dr d\theta.$$

□

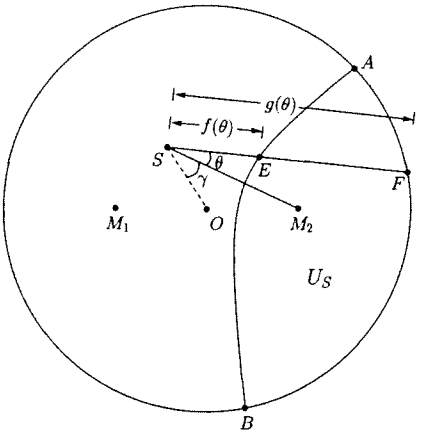

Figure 4.3: Computing the area of U_s using polar coordinates.

Consider a polar coordinate system with origin O (the origin of the network) and x-axis OM_2. Let (r, α) be the polar coordinates of a node S. The expected value of Δ_{U_s} can be calculated from

$$E[\Delta_{U_s}] = \frac{4}{\pi R^2} \int_{\alpha=\frac{\pi}{2}}^{\pi} \int_{r=0}^{R} \Delta_{U_{(r,\alpha)}} r dr d\alpha. \tag{4.6}$$

As finding a closed form for (4.6) is difficult (if not impossible), we compute it numerically. Figure 4.4 shows the numerically computed $E[\Delta_{U_s}]/(\pi R^2)$. As shown in the figure, $E[\Delta_{U_s}]/(\pi R^2)$ is maximized when the attackers M_1 and M_2 are located at the coordinates $(-0.33R, 0)$ and $(0.33R, 0)$. For the simulation, the radius of the network is set to $R = 1$ and 500 nodes are randomly put inside the network. We run the simulation for the transmission ranges $T = 0.2$ and $T = 0.15$. In the simulation, a communication is considered affected by the attack if the shortest path through the wormhole is shorter than the legitimate shortest path between the source and the destination. The simulation is then repeated a few hundred times in order to obtain the average percentage of affected communications across the network. As shown in Figure 4.4, both the simulation and the analytical results indicate that two malicious nodes can disrupt 32% of all communications across the network when they initiate a wormhole attack.

Now, consider the case where $(n \geqslant 2)$ malicious nodes attack the network by making wormholes between each other. In this case, a malicious node

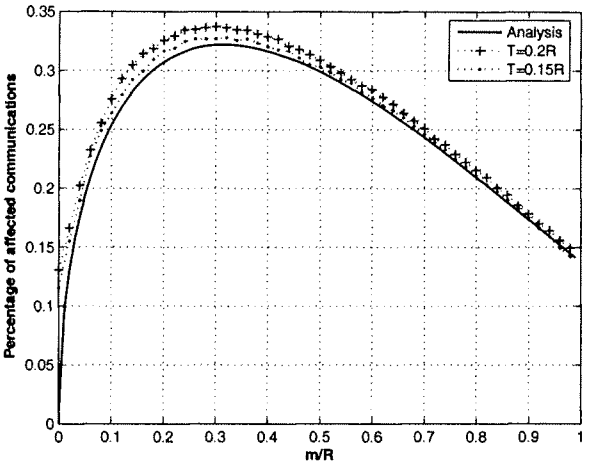

Figure 4.4: Percentage of affected communications across the network.

can send packets to any other malicious nodes using a path of wormholes. Clearly, this generalized wormhole attack can disrupt more communications across the network. The following theorem gives an upper bound on the average percentage of affected communications.

Theorem 4.2. *Let M_1, M_2, ..., M_n be ($n \geqslant 2$) malicious nodes making wormholes between each other. On average, at least $\frac{1}{n}$ of all communications across the network are not affected by the attack.*

Proof. As shown in Figure 4.5, the network can be partitioned into n regions or Voronoi cells [62] such that each cell contains exactly one malicious node and every point is closer to the malicious node in its cell than the others. Let S and D be two nodes inside a cell with the malicious node M_g. Using the Voronoi cell definition and the triangle inequality, we have

$$d_{S,D} \leqslant d_{S,M_g} + d_{D,M_g} < d_{S,M_i} + d_{D,M_j},$$

where $1 \leqslant i, j \leqslant n$ and $i \neq j$. Thus, the malicious nodes cannot disrupt the communication of two nodes inside the same cell. Suppose m_i is the number of nodes in the cell C_i. Using the Cauchy-Schwartz inequality, we get

$$\frac{\sum_{i=1}^{n} \binom{m_i}{2}}{\binom{\sum_{i=1}^{n} m_i}{2}} \sim \frac{\sum_{i=1}^{n} m_i^2}{(\sum_{i=1}^{n} m_i)^2} \geqslant \frac{1}{n}.$$

It follows that at least $\frac{1}{n}$ of all of the communications are not affected by the attackers. □

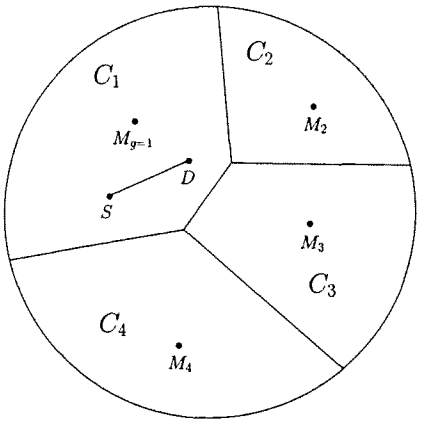

Figure 4.5: Partitioning a network with several attackers into Voronoi Cells.

4.3.3 Targeting a Particular Node in the Network

The malicious nodes may launch the wormhole attack to disrupt/control the communication between a *target node* and the other nodes in the network. For example, in wireless sensor networks, sensor devices typically collect data and send their reading to a sink node. Therefore, many-to-one communication can be a dominant flow in such networks. Using this fact, the malicious nodes can strategically choose their locations in order to maximize the damage to the sink node (the target node in this case).

Lemma 4.4. *To maximize the number of disrupted/controlled communications, one of the malicious nodes must be placed in the transmission range of the target node.*

Proof. Suppose that the malicious nodes M_1 and M_2 are located at distance d_1 and $d_2 > d_1$ from the target node. Using Lemma 4.2, the malicious nodes can control the communication between a node A and the target node S if

$$\min\left(d_{A,M_1} + d_2, d_{A,M_2} + d_1\right) \leqslant d_{A,S}.$$

Since $d_2 > d_1$, we have $d_{A,M_1} + d_2 > d_{A,M_1} + d_1$. Using the triangle inequality, we get $d_{A,M_1} + d_1 > d_{A,S}$. Therefore,

$$\min\left(d_{A,M_1} + d_2, d_{A,M_2} + d_1\right) \leqslant d_{A,S} \qquad \Longleftrightarrow \qquad d_{A,M_2} + d_1 \leqslant d_{A,S}.$$

Consequently, d_1 (or equivalently the number of hops between M_1 and the target node) has to be minimized in order to maximize the number of controlled communications. □

Now, suppose that M_1 is in the transmission range of the target node. The malicious nodes can control the communication between a node A and the target node S if $N_{A,M_2} < N_{A,S}$. In this case, the length of the shortest path though M_2 and M_1 is N_{A,M_2} which is smaller than the length of the actual shortest path, i.e., $N_{A,S}$. Using (4.3), the attackers can control the communication between A and S if

$$d_{A,M_2} < d_{A,S}. \tag{4.7}$$

As shown in Figure 4.6, the perpendicular bisector of the line segment SM_2 partitions the network into two regions $\mathcal{R}_S$ and $\mathcal{R}_M$ containing S and M_2, respectively. Clearly, Inequality 4.7 holds for any node A inside the region $\mathcal{R}_M$. Therefore, to maximize the damage, the malicious node M_2 has to be close to the target node. However, M_2 should maintain a minimum distance d_{min} from the target node S. It is because for any node A inside $\mathcal{R}_M$ we have $d_{A,S} - d_{A,M_2} \leqslant d_{S,M_2}$. Therefore, $d_{A,S} \approx d_{A,M_2}$ when d_{S,M_2} is small and thus $N_{A,S} = N_{A,M_2}$ with high probability. Note that when $N_{A,S} = N_{A,M_2}$, the malicious nodes may not take the control over the communication between A and S. For example, M_2 and S would receive all the messages at the same time if they are very close together. Clearly, the wormhole attack would not be effective in this case.

Lemma 4.5. *For the fixed location of the target node S and the fixed distance $\overline{SM_2}$, the area of $\mathcal{R}_M$ is maximized when M_2 and S are on a diagonal of the network.*

Proof. The area of circular sector (the region between a chord and a circle) can be computed as

$$\left(\frac{2\alpha - \sin(2\alpha)}{2}\right) R^2, \tag{4.8}$$

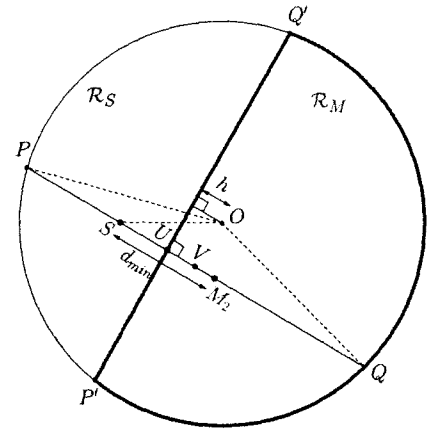

Figure 4.6: Computing the area of $\mathcal{R}_M$.

where $\alpha = \arccos(-\frac{h}{R})$ and $-R \leqslant h \leqslant R$ is the distance between the center of circle and the chord of the sector. Note that each chord partitions the circle into two sectors. Equation (4.8) gives the area of the smaller sector when h is negative and the area of the bigger sector when it is positive.

As shown in Figure 4.6, the line containing S and M_2 crosses the circle (the network boundary) at points P and Q. Let the chord $P'Q'$ be the perpendicular bisector of the line segment SM_2 and U and V be the midpoints of line segments SM_2 and PQ, respectively. The center of circle is on the perpendicular bisector of the line segment PQ. Therefore, the distance between the center of circle and the chord $P'Q'$ is equal to the distance between U and V and can be computed as

$$\begin{aligned} h = \overline{UV} &= \frac{\overline{PQ}}{2} - \left(\frac{\overline{PM_2} + \overline{PS}}{2}\right) = \left(\frac{\overline{QS} + \overline{PS}}{2}\right) - \left(\frac{\overline{PM_2} + \overline{PS}}{2}\right) \\ &= \left(\frac{\overline{QS} - \overline{PS}}{2}\right) - \left(\frac{\overline{PM_2} - \overline{PS}}{2}\right) = \left(\frac{\overline{QS} - \overline{PS}}{2}\right) - \frac{d_{min}}{2}, \end{aligned} \tag{4.9}$$

where $d_{min} = \overline{SM_2}$ is the distance between S and M_2. Using the triangle inequality for the triangles SOQ and SOP we get

$$\overline{QS} \leqslant R + \overline{OS} \qquad \text{and} \qquad \overline{PS} \geqslant R - \overline{OS}.$$

Therefore, based on (4.9), h is maximized when S and M_2 are on a diameter

of the circle. In this case, we have

$$h = \overline{OS} - \frac{d_{min}}{2}. \tag{4.10}$$

□

Using (4.8) and (4.10), we can estimate the ratio of the number of communications controlled by the attackers over the total number of communications as

$$\frac{\left(\frac{2\alpha - \sin(2\alpha)}{2}\right) R^2}{\pi R^2} = \frac{2\alpha - \sin(2\alpha)}{2\pi}, \tag{4.11}$$

where $\alpha = \arccos(-\frac{\overline{OS} - \frac{d_{min}}{2}}{R})$.

To obtain (4.11) we used Approximation (4.3). Without using (4.3) it is possible to show that (4.11) is an upper bound for the ratio of the number of communications that can be controlled on average by two malicious nodes. Suppose $\mathcal{R}_C$ is the region composed of $\mathcal{R}_S$ and its reflection in respect to $Q'P'$. Note that the region $\mathcal{R}_C$ is symmetric with respect to M_2 and S. Therefore, the malicious nodes cannot control on average more than 50% of the communications between S and the nodes inside $\mathcal{R}_C$. Suppose that the malicious nodes can control all of the communications between S and the nodes outside $\mathcal{R}_C$. Thus, the ratio of the number of communications controlled by the attackers can be bounded by

$$0.5 \times \frac{2 \times \Delta(\mathcal{R}_S)}{\pi R^2} + 1 \times \frac{\Delta(\mathcal{R}_M) - \Delta(\mathcal{R}_S)}{\pi R^2} = \frac{\Delta(\mathcal{R}_M)}{\pi R^2} = \frac{\left(\frac{2\alpha - \sin(2\alpha)}{2}\right) R^2}{\pi R^2}, \tag{4.12}$$

where $\Delta(\mathcal{R})$ denotes the area of region $\mathcal{R}$.

As shown in Figure 4.7, simulation results confirm the analytical results. For the simulation, the radius of the network is set to one and 500 nodes are randomly put inside the network. We run the simulation for the transmission ranges $T = 0.2$ and $T = 0.15$. The distance d_{min} is set to $2T$ and OS (the distance of the target node from the network center) is varied from 0 to 1 of 0.1 unit steps. As shown in Figure 4.7, both the simulation and the analytical results indicate that two malicious nodes can disrupt most of the communications between the target node and other nodes if the target node is located around the boundary of the network. Using our model, it can be easily shown that three attackers can control most of the communications

independently of the target node location. As shown in Figure 4.8, this can be done by putting one of the malicious nodes (M_1 in the figure) in the transmission range of the target node and the other malicious nodes on each side of the target node on the diagonal containing it. The shaded segments in Figure 4.8 show the unreachable region for the target node. In other words, the malicious nodes can control the communications between the target node and most of the nodes in the shaded segments, particularly the nodes that are far from the chords AB and $A'B'$.

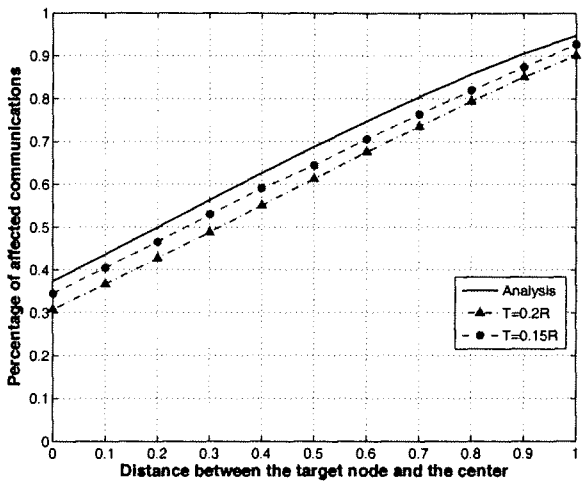

Figure 4.7: Percentage of affected communications to/from the target node.

4.4 Wormhole Attack Analysis for a Generic Network

In the previous section, we analyzed the average effect of the wormhole attack where the nodes are distributed uniformly. Clearly, the wormhole attack can have different effects on different network topologies. In this section, we first show how to compute the maximum effect of the wormhole attack on a generic network. We then use the proposed approach to evaluate the effect of the wormhole attack on grid networks.

A wireless network can be represented using a unit disk graph $G(V, T)$ with vertex set $V \in \mathbb{R}^2$ and an edge $vw \in E(G(V, T))$ whenever $\overline{vw} \leqslant T$. In

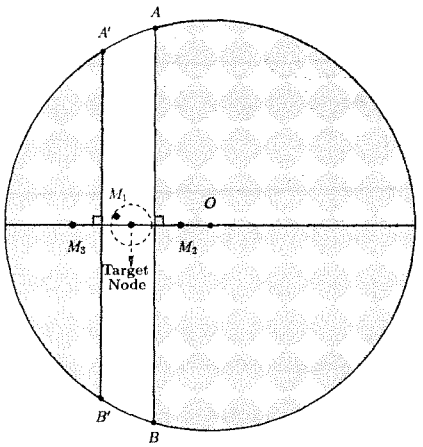

Figure 4.8: Unreachable region for the target node under attack of three malicious nodes M_1, M_2 and M_3.

this model, V corresponds to the location of nodes and T to their transmission range. Given the location of the attackers in the network, we can compute the number of affected communications by considering all the pairs of nodes and checking whether the shortest path between them contains the malicious nodes. However, without this information, we may need to consider all the possible locations of the attackers in order to compute the maximum effect of the wormhole attack.

Suppose the effect of the wormhole attack is maximized when the attackers are located at positions $P_1, P_2 \in \mathbb{R}^2$. Let us call a set of points S an *inclusive set* if there exists $P_1', P_2' \in S$ such that the effect of the wormhole attack with attackers located at P_1' and P_2' is the same as that of the case when they are located at P_1 and P_2. Clearly, it is preferable to find an inclusive set with small cardinality in order to reduce the size of the search domain. The following lemma shows how to generate an inclusive set of size $\mathcal{O}(N^2)$.

Lemma 4.6. *Let $V = \{v_1, v_2, \ldots v_N\}$ be the vertex set of a connected graph G, and $C = \{c_1, c_2, \ldots c_N\}$ be the set of circles c_i centered at v_i with radius T. Suppose S is the set of all the intersections of any two circles in the set C. The set S is an inclusive set of size $\mathcal{O}(N^2)$.*

Proof. Suppose the effect of the wormhole attack is maximized when the attackers are located at P_1 and P_2. We show that there is a point $P_1' \in S$

such that an attacker does not lose any of its neighbors if it is moved from P_1 to P_1'. There are three possible cases for the point P_1.

1. P_1 is located on the circumference of at least two circles in C;
2. P_1 is located on the circumference of exactly one circle in C;
3. P_1 is not located on circumference of any circle in C.

For the first case, we set $P_1' = P_1$. Suppose that P_1 is located on the circumference of exactly one circle $c_i \in C$ (Case 2). Since graph G is connected, c_i will intersect with at least one circle. Therefore, P_1 can be rotated on the circumference of c_i until it lies, for the first time, on an intersection of c_i and $c_j \in C$ denoted by P_1'. For the third case, we can shift P_1 towards a vertex $v \in V$ until it lies on the first circle c_i at point P_1^*. We set $P_1' = P_1^*$ if P_1^* is on the circumference of at least two circles. Otherwise, similar to the second case, we rotate P_1^* on the circumference of c_i until it lies, for the first time, on an intersection of c_i and $c_j \in C$ denoted by P_1'. Note that during the process of transferring P_1 to P_1' for all three cases, P_1 does not exit any circle. Therefore, we have

$$\forall u \in V: \quad \overline{uP_1} \leqslant T \quad \Longrightarrow \quad \overline{uP_1'} \leqslant T.$$

In other words, an attacker does not lose any of its neighbors (and thus its effect) if it is moved from P_1 to P_1'. The same process can be applied to the point P_2 to get P_2'. Note that the effect of attackers at P_1' and P_2' cannot be more than that of attackers at P_1 and P_2 since in the latter case the attackers have maximum effect. Since P_1' and P_2' are on the intersection of at least two circles, it follows that S is an inclusive set. Clearly the size of S is at most $\binom{N}{2}$, so it is of size $\mathcal{O}(N^2)$. □

Using Lemma (4.6), we can find optimal positions where the malicious nodes can maximize the damage caused by the wormhole attack. However, finding the inclusive set S introduced in Lemma (4.6) may not be practical, as in practice, we may not have the location information of the nodes or their transmission ranges. Note that having the location of the nodes is important since determining whether an arbitrary graph is a unit disk graph is $\mathcal{NP}$-complete [63]. Therefore, it may be preferable to compute an approximation of the maximum damage of the wormhole attack when the nodes' location information is not available. One possible solution, in this case, is to use the

vertex set V instead of an inclusive set to get a lower bound of the maximum damage. Using this approach, we do not need to have any information about the location of the nodes and their transmission ranges. Moreover, the size of the vertex set is less than the size of S, so the search domain is smaller. Clearly, we can always find $u, v \in V$ such that

$$\overline{uP_1'} \leqslant T \quad \text{and} \quad \overline{vP_2'} \leqslant T.$$

Therefore, using V as an inclusive set, will often lead to a good approximation of the maximum damage of the wormhole attack.

As an example, consider a simple grid topology network of $N = m \times m$ nodes. In this topology, the length of the shortest path between two nodes u and v is equal to $|u_x - v_x| + |u_y - v_y|$, where u_x and v_x denote the x-coordinates of the nodes u and v and u_y and v_y denote their y-coordinates. We evaluated the maximum damage of wormhole attack using simulation on a grid topology network of m by m. Simulation results indicate that the maximum damage occurs when the malicious nodes are located on the main diagonal of the grid. Figure 4.9 shows the percentage of maximum number of affected communications. As shown in the figure, the wormhole attack launched by two malicious nodes can affect around 40% to 50% of all of the possible communications in a grid topology network of $m \times m$, where $8 \leqslant m \leqslant 32$. The simulation results indicate that the percentage of maximum number of affected communications decreases as the total number of nodes increases.

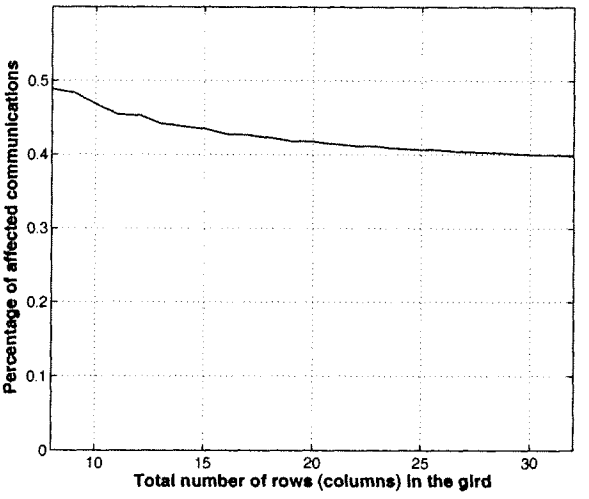

Figure 4.9: Percentage of maximum number of affected communications in a grid topology network.

4.5 Wormhole Attack Countermeasure

As mentioned in Section 4.2, the existing wormhole countermeasures can be divided into proactive and reactive countermeasures. Proactive countermeasures typically use time of flight, geographical position or signal direction information to prevent wormhole formation. These techniques often require specialized hardware and can achieve a high probability of detection. Some proactive countermeasures are able to detect wormholes in certain scenarios without using specialized hardware, For example, using the algorithm proposed in [57], each node can detect a wormhole by analyzing its local neighbor information. However, we will show in this section that this method fails if the attackers selectively forward the *Hello* messages. In contrast to proactive countermeasures, reactive countermeasures can prevent the wormhole attack launched in either hidden or participation mode. However, they do not prevent the formation of wormhole and do not avoid the wormhole if it is used for an invisible (passive) attack such as the traffic analysis attack.

In this section, we propose a proactive countermeasure based on timing analysis. Timing analysis techniques are based on the fact that a packet can travel at most at the speed of light. Therefore, a node can estimate

its distance to a sender by multiplying Packet Travel Time (PTT) by the light speed (c). The existing methods to compute PTT can be divided into two classes of synchronous and asynchronous methods. The synchronous methods are based on accurate time synchronization. For example, in the method proposed in [50], each sender stamps the packet with the time at which it is sent. The receiver can then compute PTT by comparing the time stamp with the time at which it receives the packer. Clearly, this method requires the nodes to have tightly synchronized clocks. Moreover, the sender needs to know the precise sending time which may not be possible in, for example, medium access control protocols based on CSMA [50].

Asynchronous methods do not require time synchronization. These methods typically compute PTT using a few rounds of message exchanges. For example, a node A can compute PTT by comparing the time at which it send a packet to a node B to the time at which it receives an instant response from B. In [51], PTT is computed through a series of fast one-bit exchanges. This method requires fast switching between the receive and send modes. In [54], the authors proposed a similar asynchronous method that does not need fast switching and requires only minor changes in IEEE 802.11-capable hardwares. In this method, link verification is performed in two phases. In the first phase, the nodes exchange nonces (i.e., randomly generated numbers) via a single RTS-CTS-DATA-ACK exchange. In the second phase, they mutually authenticate themselves by signing and transmitting their respective nonces.

One of the common drawbacks of the aforementioned asynchronous methods is that they need to verify the vicinity of each neighbor one-by-one. In this section, we propose a new asynchronous method that does not have the drawbacks of the previous methods. Using our proposed method, the nodes do not need to have synchronized clocks, and are not required to predict the sending time or to be capable of fast switching between the receive and send modes. Moreover, the nodes do not need to communicate with all their neighbors one-to-one. It is only assumed that each node is able to record the time at which a packet is fully sent/received. Using our scheme, each node can validate vicinity of all its neighbors in two rounds of communication. In the first round, each node sends a *signed Hello* message containing its *id* and a nonce, and records the time at which the message is fully sent. It follows that after the first round, each node has a list of all its potential neighbors. In the second round, each node signs and sends a *follow-up* packet. The follow-up packet includes the time at which the node's *Hello* message was

sent (in the first round), the list of all the *ids* in the received *Hello* messages together with their corresponding nonces and the times at which they were received. Nonces are used to prevent malicious nodes to masquerade a legitimate node. Note that neither *Hello* messages nor follow-up packets are timestamped with their transmission time. Therefore, the nodes do not require to compute a signature while having to timestamp the packet with its transmission time.

Suppose that node A receives B's *Hello* message after sending its own. When A receives a follow-up packet from B, it first checks its corresponding nonce in the packet and verifies B's signature. It then accept B as its neighbor if

$$\frac{(t_{A,B} - t_A) - (t_B - t_{B,A})}{2} \times c \leqslant T_{max}, \tag{4.13}$$

where t_x is the sending time of x's *Hello* message recorded by x, $t_{x,y}$ is the receiving time of y's *Hello* message recorded by x and T_{max} is the maximum transmission range. Node A considers B's *Hello* message as a response to its *Hello* message. The term $(t_{A,B} - t_A)$ indicates the time it takes to get the response. However, B's *Hello* message is not an immediate response. Therefore, A subtracts $(t_B - t_{B,A})$, the delay at node B, from $(t_{A,B} - t_A)$. Note that A extracts $t_{B,A}$ and t_B from B's follow-up packet.

To derive (4.13), it is assumed that node A receives B's *Hello* message after sending its own. If A sends its *Hello* message after receiving B's then A considers B as neighbor if

$$\frac{(t_{B,A} - t_B) - (t_A - t_{A,B})}{2} \times c \leqslant T_{max}. \tag{4.14}$$

It is worth mentioning that a node A can estimate the distance between its neighbors B and C using similar equations as (4.13) and (4.14). It is because all the information used by B or C to estimate $d_{B,C}$ is available in B's and C's follow-up packets.

After the second round, each node has a list of all its 2-hop neighbors. Therefore, it can employ the Maheshwari's algorithm [57] to further check the existence of a wormhole. Maheshwari's algorithm uses local neighborhood information to find a forbidden structure. The algorithm is based on the fact that inside a fixed region it is not possible to pack too many nodes without having edges between them. Using simulation, the authors show that the algorithm can detect wormhole with a 100% detection and 0% false alarm probabilities. However, Proposition 4.3 shows that the algorithm cannot

detect a wormhole if the malicious nodes only forward the *Hello* messages of those node which are at most $\frac{T}{2}$ away from an end of wormhole. The malicious node may do selection based on the power of received packet. In other words, they can only forward packets with power higher than a specific threshold. Note that selectively forwarding packets can increase the delay of transferring a packet from one end to the other end of the wormhole. Therefore, this attack can be prevented if Maheshwari's algorithm is used together with our proposed scheme.

Theorem 4.3. *Suppose that the length of wormhole is at least* $(2+\sqrt{3})T$ *and the malicious nodes only forward the Hello messages of the nodes that are at most* $\frac{T}{2}$ *away from one end of wormhole. In this case, Maheshwari's algorithm is not able to detect the wormhole if it uses 2-hop neighbor information.*

4.6 Summary

In this chapter, we investigated the effect of the wormhole attack on shortest-path routing protocols. We showed that two strategically located attackers can disrupt on average 32% of all communications across the network, when the nodes are distributed uniformly. We also studied the effect of the wormhole attack launched by $n \geqslant 2$ malicious nodes and showed that on average at least $\frac{1}{n}$ of all communications are not affected by the attack. We considered the scenario in which the malicious nodes attack a target node in the network. For this scenario, we showed that two malicious nodes can disrupt/control on average 30% to 90% (based on the location of the target node in the network) of all communications between the target node and all other nodes in the network. We also evaluated the effect of the wormhole attack on grid topology networks. In these networks, we showed that wormhole attack can disrupt/control about 40% to 50% of all communications if the wormhole is strategically placed on the diagonal of the network. Finally, We proposed a timing-based solutions to defend against the wormhole attack. We showed that, using our proposed solution, nodes need less requirements (compared to the existing timing-based countermeasures) to detect false neighbors. Moreover, we showed that the proposed solution can be used to strengthen one of the recently proposed novel countermeasures.

Bibliography

[50] Y. C. Hu, A. Perrig, and D. B. Johnson, "Packet leashes: A defense against wormhole attacks in wireless ad hoc networks," *In Proc. of IEEE INFOCOM*, 2003.

[51] S. Capkun, L. Buttyan, and J. P. Hubaux, "SECTOR: Secure tracking of node encounters in multi-hop wireless networks," *In Proc. of the first ACM workshop on Security of ad hoc and sensor networks*, 2003.

[52] L. Hu and D. Evans, "Using directional antennas to prevent wormhole attacks," *In Proc. of Network and Distributed System Security Symposium*, 2004.

[53] K. Sun, P. Ning, C. Wang, A. Liu, and Y. Zhou, "TinySeRSync: Secure and resilient time synchronization in wireless sensor networks," *In Proc. of the* 13^{th} *ACM Conference on Computer and Communications Security*, 2006.

[54] J. Eriksson, S. V. Krishnamurthy, and M. Faloutsos, "Truelink: A practical countermeasure to the wormhole attack in wireless networks," *In Proc. of IEEE International Conference on Network Protocols*, 2006.

[55] L. Lazos, R. Poovendran, C. Meadows, P. Syverson, and L. W. Chang, "Preventing wormhole attacks on wireless ad hoc networks: a graph theoretic approach," *In Proc. of IEEE WCNC*, 2005.

[56] K. Rasmussen and S. Capkun, "Implications of radio fingerprinting on the security of sensor networks," *In Proc. of IEEE SecureComm*, 2007.

[57] R. Maheshwari, J. Gao, and S. R. Das, "Detecting wormhole attacks in wireless networks using connectivity information," *In Proc. of IEEE INFOCOM*, 2007.

[58] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens, "An on-demand secure routing protocol resilient to byzantine failures," *In Proc. of ACM Workshop on Wireless Security (WiSe)*, 2002.

[59] I. Avramopoulos, H. Kobayashi, R. Wang, and A. Krishnamurthy, "Highly secure and efficient routing," *In Proc. of IEEE INFOCOM*, 2004.

[60] I. Khalil, S. Bagchi, and N. B. Shroff, "LITEWORP: A lightweight countermeasure for the wormhole attack in multihop," *In Proc. of the International Conference on Dependable Systems and Networks*, 2005.

[61] A. Papoulis, *Probability and statistics.* Prentice-Hall, Inc., 1990.

[62] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications,* 2^{nd} *ed.* New York: Springer-Verlag, 2000.

[63] H. Breu and D. Kirkpatrick, "Unit disk graph recognition is NP-hard," *Computational Geometry*, vol. 9, no. 1–2, pp. 3–24, 1998.

Chapter 5

Double Point Compression with Applications to Speeding up Random Point Multiplication

5.1 Introduction

In cryptographic protocols based on elliptic curve cryptography [64], it is often necessary to transmit or store elliptic curve points [65]. An elliptic curve point can be represented as $P = (x, y)$, where x is the x-coordinate and y is the y-coordinate. However, it is desirable to represent a point using a compact representation in order to reduce the required bandwidth/memory. One trivial way to do this is to represent a point by its x-coordinate and an additional bit. The elliptic curve equation is a quadratic equation in y, thus given x we can obtain at most two possible solutions of y. The additional bit can then be used to specify the actual y-coordinate. Solving the quadratic equation requires a square root operation. Therefore, this technique is not practical when the square root operation is computationally expensive. In this case, an elliptic curve point is typically represented by both its x-coordinate and y-coordinate.

In the first part of the chapter, we introduce a novel double point compression scheme which allows a compact representation of elliptic curve points without the computational cost associated with ordinary single point compression. We also extend double point compression to triple point compression in order to get more savings. Using the proposed double point and triple point compression schemes we can save about 25% and 33% of the bandwidth/memory, respectively. Moreover, the compression process of the proposed schemes requires almost no computational effort, and decompression

A version of this chapter has been published. M. Khabbazian, T.A. Gulliver and V.K. Bhargava, Double Point Compression with Applications to Speeding up Random Point Multiplication, *IEEE Transactions on Computers*, vol. 56, no. 3, pp. 305-313, 2007.

involves no square root operations. The proposed point compression schemes can be trivially used to compress multiple points. In this case, Montgomery's trick for simultaneous inversion [66] can be employed to significantly reduce the decompression cost by replacing almost each field inversion with three field multiplications.

The second part of the chapter deals with speeding up random point multiplication. Point multiplication, kP, is a fundamental operation which dominates the execution time of elliptic curve cryptosystems. Significant research effort has been spent on reducing the time needed to perform this operation. There are different types of point multiplication algorithms including fixed point multiplication algorithms and random point multiplication algorithms. Fixed Point Multiplication (FPM) algorithms are used when a fixed base point is multiplied several times. In this case, a lookup table can be generated once and used every time a multiplication of the base point is required. Random Point Multiplication (RPM) algorithms, on the other hand, are designed for the case where the base point is multiplied only once. To speed up the operation, these algorithms typically use low average Hamming weight integer representations such as w-NAF (see [67]) and the representation proposed in [68].

In this chapter, we extend RPM algorithms to the case where the base point is multiplied only once and we are allowed to have some limited extra information regarding the base point. We refer to the new algorithms as Partially Random Point Multiplication (PRPM) algorithms. We introduce the first PRPM algorithm which uses Möller's algorithm [69] in its evaluation stage. We speed up the precomputation stage of the algorithm using Montgomery's trick for simultaneous inversion. We then compare the performance of the proposed PRPM algorithm with a standard RPM algorithm for different integer representations and show that a significant speed up can be obtained at the cost of slightly higher required bandwidth. We also show that a substantial speed improvement can be obtained when multiple processors are available. Note that this technique can be used to speed up point multiplication in any devices, particularly those with constrained computational power or servers overloaded with elliptic curve encryption or key agreement requests. In this case, the proposed point compression techniques can be employed to reduce the required bandwidth when single point compression is not practical.

The remainder of the chapter is organized as follows. In Section 5.2, we briefly summarize elliptic curve operations. In Section 5.3, we propose a dou-

ble point compression algorithm and extend it to triple point compression. In Section 5.4, we review some of the best fixed and random point multiplication algorithms and explain how to extend random point multiplication algorithms. We introduce a partially random point multiplication algorithm in Section 5.5 and show how to accelerate its precomputation stage. We also analyze the performance of the proposed algorithm and show that a substantial performance improvement in speed can be obtained when multiple processors are available. Finally, we present a summary in Section 5.6.

5.2 Elliptic Curves: Definition and Operations

An elliptic curve E over the field $\mathbb{F}$ is a smooth curve in the so called "long Weirestraß form"

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \tag{5.1}$$

where $a_1, \ldots, a_6 \in \mathbb{F}$. Let $Char(\mathbb{F})$ denote the characteristic of the field $\mathbb{F}$. Equation (5.1) can be simplified to

$$E : y^2 = x^3 + ax + b, \tag{5.2}$$

and

$$E : y^2 + xy = x^3 + ax^2 + b, \tag{5.3}$$

when $Char(\mathbb{F}) \neq 2, 3$ and $Char(\mathbb{F}) = 2$, respectively.

Let $E(\mathbb{F})$ be the set of points $P = (x, y) \in \mathbb{F}^2$ that satisfy the elliptic curve equation (along with a "point at infinity" denoted $\mathcal{O}$). It is well known that $E(\mathbb{F})$ together with the point addition operation given in Table 5.1 form an abelian group. When the two operands are equal, i.e. $P_1 = P_2$, the operation is called point doubling. As shown in Table 5.2, point inversion in this group can be computed easily. Thus, point subtraction has virtually the same cost as point addition. Note that in Tables 5.1 and 5.2, the points $P_1 = (x_1, y_1), P_2 = (x_2, y_2)$ and $P_3 = (x_3, y_3)$ are represented by affine coordinates. Alternative coordinates such as projective coordinates or Jacobian coordinates can be used to avoid field inversion when field inversion is much more expensive than field multiplication. Point multiplication is defined as

repeated addition

$$kP = \underbrace{P + P + \ldots + P}_{k \text{ times}},$$

where P is an elliptic curve point, $k \in [1, N-1]$ is an integer, and N is the order of point P.

$Char(\mathbb{F}) = 2$	$P_1 \neq \pm P_2$	$\lambda = \frac{y_2+y_1}{x_2+x_1}$ $x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$ $y_3 = (x_1 + x_3)\lambda + y_1 + x_3$
	$P_1 = P_2$	$\lambda = \frac{y_1}{x_1} + x_1$ $x_3 = \lambda^2 + \lambda + a$ $y_3 = (x_1 + x_3)\lambda + y_1 + x_3$
$Char(\mathbb{F}) \neq 2, 3$	$P_1 \neq \pm P_2$	$\lambda = \frac{y_2-y_1}{x_2-x_1}$ $x_3 = \lambda^2 - x_1 - x_2$ $y_3 = (x_1 - x_3)\lambda - y_1$
	$P_1 = P_2$	$\lambda = \frac{3x_1^2+a}{2y_1}$ $x_3 = \lambda^2 - 2x_1$ $y_3 = (x_1 - x_3)\lambda - y_1$

Table 5.1: Elliptic curve point addition, $P_3 = P_1 + P_2$.

$Char(\mathbb{F}) = 2$	$-P_1 = (x_1, x_1 + y_1)$
$Char(\mathbb{F}) \neq 2, 3$	$-P_1 = (x_1, -y_1)$

Table 5.2: Elliptic curve point inversion.

5.3 Point Compression

Using single point compression, an elliptic curve point $P = (x, y)$ can be represented by its x-coordinate and an additional bit. This is because given x, the elliptic curve equation becomes quadratic in y. The quadratic equation has at most two solutions, so one bit is sufficient to specify y (the additional bit is not required when $Char(\mathbb{F}) = 2$ and P has odd order [70]). For example, for the case of $\mathbb{F}_p$, we have

$$y^2 = x^3 + ax + b = x(x^2 + a) + b.$$

Therefore, given x and an additional bit, y can be obtained at the cost of $1M+1S+1S_R$, where M, S and S_R denote the cost of a field multiplication, a field squaring and a square root operation, respectively. The main drawback of the single point compression scheme is that it is not practical when square root operation is expensive. For example, to obtain a square root of a modulo a prime $p \equiv 3 \mod 4$, we need to compute $a^{\frac{p+1}{4}} \mod p$, which requires $\Omega(\log p)$ field multiplications/squaring. In this case, the elliptic curve points are typically represented by both their x-coordinate and y-coordinate to avoid the overhead of computing square root operation. In this work, we present two novel point compression schemes that do not require any square root operation.

5.3.1 Double Point Compression

In some cases, such as the elliptic curve ElGamal encryption [71] and the proposed technique in Section 5.4, we are required to transmit/store multiple points. In these cases, we can use double point compression as given in Proposition 5.1 below to reduce the required bandwidth/memory by about 25%. Note that decompression does not require a square root operation. Therefore, the proposed double point compression algorithm can be used when single point compression is not practical.

Theorem 5.1. *The elliptic curve points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ can be represented by at most three field elements and an additional bit.*

Proof. Suppose $Char(\mathbb{F}) \neq 2, 3$ and $y_1 + y_2 \neq 0$. The points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ can be represented by $(x_1, x_2, y_1 + y_2)$. Given x_1, x_2 and $y_1 + y_2 = A$, we can obtain y_1 and y_2 as

$$y_1 = \frac{A}{2} + (8A)^{-1}(x_1 - x_2)\left(3(x_1 + x_2)^2 + (x_1 - x_2)^2 + 4a\right), \tag{5.4}$$

and

$$y_2 = A - y_1. \tag{5.5}$$

This is because, using (5.2), we have

$$\begin{aligned}
&\frac{A}{2} + (8A)^{-1}(x_1 - x_2)\left(3(x_1 + x_2)^2 + (x_1 - x_2)^2 + 4a\right) \\
&= \frac{A}{2} + (8A)^{-1} \times 4\left((x_1^3 + ax_1 + b) - (x_2^3 + ax_2 + b)\right) \\
&= \frac{A}{2} + (8A)^{-1} \times 4(y_1^2 - y_2^2) \\
&= \frac{y_1 + y_2}{2} + \frac{y_1 - y_2}{2} = y_1.
\end{aligned}$$

Therefore, computing both y_1 and y_2 costs $2M + 2S + I$, where I denotes the cost of a field inversion. In other words, decompressing each point costs $M + S + 0.5I$. When $y_1 + y_2 = 0$, we can represent the points simply by (x_1, x_2, y_1) as $y_2 = -y_1$. Clearly, decompression requires almost no computational effort in this case. An additional bit can be used to distinguish between the two cases $y_1 + y_2 \neq 0$ and $y_1 + y_2 = 0$.

Now, suppose that $Char(\mathbb{F}) = 2$ and $x_1 + x_2 \neq 0$. The points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ can be represented by $(x_1, x_2, y_1 + y_2)$. Given x_1, x_2 and $y_1 + y_2 = A$, we can obtain y_1 and y_2 as

$$y_1 = \left((a + x_2)(x_1 + x_2) + x_1^2\right) + A(A + x_2)(x_1 + x_2)^{-1},$$

and

$$y_2 = A - y_1 \quad (\text{or } y_2 = A + y_1 \text{ since } Char(\mathbb{F}) = 2).$$

This is because, using (5.3), we have

$$\begin{aligned}
&\left((a + x_2)(x_1 + x_2) + x_1^2\right) + A(A + x_2)(x_1 + x_2)^{-1} \\
&= (x_1^3 + x_2^3 + ax_1^2 + ax_2^2)(x_1 + x_2)^{-1} + (y_1^2 + y_2^2 + y_1x_2 + y_2x_2)(x_1 + x_2)^{-1} \\
&= \left((y_1^2 + x_1^3 + ax_1^2 + b) + (y_2^2 + x_2^3 + ax_2^2 + y_2x_2 + b) + y_1x_2\right)(x_1 + x_2)^{-1} \\
&= (y_1x_1 + y_1x_2)(x_1 + x_2)^{-1} = y_1.
\end{aligned}$$

Therefore, computing both y_1 and y_2 costs $3M + S + I$, and decompressing each point costs $1.5M + 0.5S + 0.5I$. When $x_1 + x_2 = 0$ we can represent the points simply by (x_1, y_1). This is due to the fact that $x_1 + x_2 = 0$, hence $x_1 = x_2$ as $Char(\mathbb{F}) = 2$. Consequently, we have $P_2 = \pm P_1$. An additional bit is sufficient to distinguish between the two cases $P_2 = P_1$ and $P_2 = -P_1$. For the case $P_2 = -P_1$, we have $y_2 = y_1 + x_1$ (see Table 5.2), so decompression requires almost no effort.

□

5.3.2 An Extension to Triple Point Compression

In this section, we extend double point compression to the case where three points are required to be transmitted/stored. We restrict ourselves to prime fields and show how to represent three points in a compact representation and save 33% of bandwidth/memory. A similar approach can be used to compress three points on an elliptic curve defined over a binary field.

Theorem 5.2. *Suppose* $Char(\mathbb{F}) \neq 2, 3$. *The elliptic curve points* $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ *and* $P_3 = (x_3, y_3)$ *can be represented by at most four field elements and two additional bits.*

Proof. Let

$$y_1 + y_2 + y_3 = A \qquad \text{and} \qquad A^2 + y_3^2 - y_1^2 - y_2^2 = B.$$

Suppose $y_i^2 \neq y_j^2$, $\forall i, j \in \{1, 2, 3\}, i \neq j$ and $A \neq 0$. Therefore, we have $B \neq 0$ because

$$B = A^2 + y_3^2 - y_1^2 - y_2^2 = 2(y_3 + y_1)(y_3 + y_2).$$

In this case, the elliptic curve points P_1, P_2 and P_3 can be represented as

$$(x_1, x_2, x_3, y_1 + y_2 + y_3 = A).$$

Given x_1, x_2 and x_3, we can obtain y_3 as

$$y_3 = \frac{B^2 + 4A^2y_3^2 - 4y_1^2y_2^2}{4AB}, \tag{5.6}$$

because

$$\begin{aligned}
\frac{B^2 + 4A^2y_3^2 - 4y_1^2y_2^2}{4AB} &= \frac{(B - 2Ay_3)^2 + 4ABy_3 - 4y_1^2y_2^2}{4AB} \\
&= y_3 + \frac{(A^2 + y_3^2 - y_1^2 - y_2^2 - 2Ay_3)^2 - 4y_1^2y_2^2}{4AB} \\
&= y_3 + \frac{(2y_1y_2)^2 - 4y_1^2y_2^2}{4AB} = y_3.
\end{aligned}$$

Note that y_i^2 can be obtained from x_i using (5.2) at a cost of $M + S$. Having y_3, we get

$$y_1 + y_2 = A - y_3.$$

Clearly, $y_1 + y_2 \neq 0$ as $y_1^2 \neq y_2^2$. Therefore, we can obtain y_1 and y_2 using (5.4) and (5.5), respectively.

As the second case, assume $y_i^2 \neq y_j^2$, $\forall i, j \in \{1, 2, 3\}, i \neq j$ and $A = 0$. Clearly, $y_1 + y_2 \neq 0$, thus the points can be represented as $(x_1, x_2, x_3, y_1 + y_2)$. We can use (5.4) and (5.5) to obtain y_1 and y_2, respectively. Then y_3 is

$$y_3 = -(y_1 + y_2).$$

Now suppose that $y_i^2 = y_j^2$ and $y_j^2 \neq y_k^2$ for different integers $i, j, k \in \{1, 2, 3\}$. Without loss of generality, we assume that

$$y_1^2 = y_2^2, \text{ and } y_2^2 \neq y_3^2.$$

In this case, we represent the points as $(x_1, x_2, x_3, y_2 + y_3)$. Clearly, $y_2 + y_3 \neq 0$ as $y_2^2 \neq y_3^2$. Thus, we can obtain y_2 and y_3 using (5.4) and (5.5), respectively. An additional bit can be used to distinguish between the two cases $y_1 = y_2$ and $y_1 = -y_2$.

Finally, suppose that

$$y_1^2 = y_2^2 = y_3^2.$$

In this case, the points can simply be represented as (x_1, x_2, x_3, y_1). Two additional bits would then be sufficient to differentiate the cases

$$y_2 = y_1, \quad y_2 = -y_1$$

and

$$y_3 = y_1, \quad y_3 = -y_1.$$

□

Table 5.3 summarizes all possible cases with the corresponding point representation and decompression cost. The conditions given in the second column can be verified using (5.2) when we have x_1, x_2 and x_3. Algorithm 5.1 provides pseudo code for triple point decompression. Note that in Lines 8 and 9 of the algorithm, Montgomery's trick for simultaneous inversion is used to reduce the required number of field inversions from two to one. Finally, it is worth mentioning that triple point compression requires almost no effort as the conditions given in Table 5.3 can easily be verified just by comparing the y-coordinates of the points.

Algorithm 5.1 Triple Point Decompression

Input: A 4-tuple $(x_1, x_2, x_3, \mathcal{T})$ and additional bits $b_1, b_2 \in \{0, 1\}$

Output: y_1, y_2 and y_3

Compute y_i^2 for $i = 1, 2, 3$ and determine the condition
switch (condition)
  case condition #1:
    if $(b_1 == 1)$ **then**
      $C \leftarrow B^2 + 4\mathcal{T}^2 y_3^2 - 4y_1^2 y_2^2$
      $D \leftarrow 2((\mathcal{T} + B)^2 - \mathcal{T}^2 - B^2) = 4\mathcal{T}B$
      $E \leftarrow \mathcal{T}D - C$
      $D^{-1} \leftarrow E(DE)^{-1}$
      $E^{-1} \leftarrow D(DE)^{-1}$
      $y_3 \leftarrow CD^{-1}$
      $y_2 \leftarrow \frac{1}{2}((y_2^2 - y_1^2)DE^{-1} + \mathcal{T} - y_3)$
      $y_1 \leftarrow \mathcal{T} - y_2 - y_3$
    end if
    if $(b_1 == 0)$ **then**
      $y_1 \leftarrow \frac{1}{2}((y_1^2 - y_2^2)\mathcal{T}^{-1} + \mathcal{T})$
      $y_2 \leftarrow \mathcal{T} - y_1$
      $y_3 \leftarrow -(y_1 + y_2)$
    end if
  end case {#1}:
  case condition #2:
      $y_k \leftarrow \frac{1}{2}((y_k^2 - y_j^2)\mathcal{T}^{-1} + \mathcal{T})$
      $y_j \leftarrow \mathcal{T} - y_k$
      $y_i \leftarrow (2b_1 - 1)y_j$
  end case {#2}:
  case condition #3:
      $y_2 \leftarrow (2b_1 - 1)y_1$
      $y_3 \leftarrow (2b_2 - 1)y_1$
  end case {#3}:

#	Condition	Additional Bits	Point Representation	Decompression Cost per Point
1	$y_i^2 \neq y_j^2$, where $i \neq j, 1 \leqslant i, j \leqslant 3$	$b_1 = 1$	$(x_1, x_2, x_3, y_1 + y_2 + y_3)$	$\frac{1}{3}I + 4M + 2S$
		$b_1 = 0$	$(x_1, x_2, x_3, y_1 + y_2)$	$\frac{1}{3}I + \frac{4}{3}M + S$
2	$y_i^2 = y_j^2$ and $y_j^2 \neq y_k^2$, where $1 \leqslant i, j, k \leqslant 3$	$b_1 = 1$	$(x_1, x_2, x_3, y_j + y_k)$	$\frac{1}{3}I + \frac{4}{3}M + S$
		$b_1 = 0$		
3	$y_1^2 = y_2^2 = y_3^2$	$b_1 = 1, b_2 = 1$	(x_1, x_2, x_3, y_1)	$M + S$
		$b_1 = 1, b_2 = 0$		
		$b_1 = 0, b_2 = 1$		
		$b_1 = 0, b_2 = 0$		

Table 5.3: Triple Point Compression.

5.3.3 Compressing Multiple Points

The proposed point compression schemes can be used to compress multiple points. For example, when we have $m = 2u$ points, we can divide them into u pairs and compress each pair using the double point compression scheme. We can also use triple point compression to get more bandwidth/memory savings. Suppose that we need to transmit/store $m \geqslant 2$ points. Clearly, any integer $m \geqslant 2$ can be uniquely written in the form $m = 3v+2u$, where $v \geqslant 0$ and $0 \leqslant u \leqslant 2$. Therefore, we can divide the points into v 3-tuples and u 2-tuples and compress them using the triple point and double point compression schemes, respectively. Note that decompressing each tuple requires one field inversion. Therefore, $v+u$ simultaneous field inversions are required to obtain all the y-coordinates. Montgomery's trick for simultaneous inversion can be employed to do this using one field inversion and $3(v + u - 1)$ field multiplications. This can significantly speed up the decompression process as a field inversion costs more than three field multiplications in most useful fields.

5.4 Point Multiplication

In this section, we briefly present some of the best known fixed and random point multiplication algorithms. We then show how to extend random point multiplication algorithms to the case where the base point is variable but some limited extra information is available.

5.4.1 Fixed point multiplication (FPM)

Fixed point multiplication algorithms are designed for the case where a fixed point is multiplied by different multipliers a number of times. In this case, the FPM algorithms generate a lookup table once and use it every time a multiplication of the point is required. Lim and Lee's algorithm [72] and Möller's algorithm [69] are two of the best fixed point multiplication algorithms.

In the Lim and Lee algorithm, the multiplier k is divided into hv subblocks of b bits

$$k = \sum_{u=0}^{l-1} e_u 2^u = \sum_{i=0}^{h-1} \left(\sum_{j=0}^{v-1} k_{i,j} 2^{jb} \right) 2^{ia},$$

where

$$e_u \in \{0,1\}, a = bv, b = \lceil \frac{l}{hv} \rceil, k_{i,j} = \sum_{t=0}^{b-1} e_{ia+jb+t} 2^t.$$

The points

$$\mathcal{L}[I][j] = \sum_{i=0}^{h-1} e_i 2^{ia+jb} P \text{ for } 0 \leqslant j < v, \text{ where } I = \sum_{i=0}^{h-1} e_i 2^i$$

are then precomputed and stored in a lookup table, and used to compute kP as follows

$$kP = \sum_{t=0}^{b-1} 2^t \left(\sum_{j=0}^{v-1} \mathcal{L}[I_{j,t}][j] \right), \text{ where } I_{j,t} = \sum_{i=0}^{h-1} e_{ia+jb+t} 2^i.$$

Möller's algorithm represents the multiplier k in w-NAF form (see [67]) and divides it into n subblocks of length $s = \lceil \frac{l}{n} \rceil$.

It stores the points $d(2^{sj}P)$ for $0 \leqslant j < n$ and for all odd integers $1 < d < 2^{w-1}$ in a lookup table, and uses them to compute kP

$$kP = \left(\sum_{0 \leqslant i < l} k_i 2^i \right) P = \sum_{0 \leqslant i < s} 2^i \left(\sum_{0 \leqslant j < n} k_{sj+i} (2^{sj} P) \right),$$

where $k_i \in \{\pm 1, \pm 3, \ldots, \pm(2^{w-1} - 1)\} \cup \{0\}$.

5.4.2 Random Point Multiplication (RPM)

Random point multiplication algorithms are designed for the case where the base point is multiplied only once. In this case, generating a lookup table is not practical since it is computationally expensive. There are many algorithms to accelerate the computation of random point multiplication by reducing the required number of point additions [65, 73]. These algorithms usually consist of two stages, typically a precomputation stage and an evaluation stage. Algorithm 5.2 is an example of such algorithms. The precomputation stage of this algorithm consists of two steps. In Step 1, the multiplier k is recoded to a B-representation

$$k = \sum_{0 \leqslant i < l} k_i 2^i,$$

where $k_i \in B \cup \{0\}$, $l \leqslant \lceil \log_2 N \rceil + 1$ and B is a set of nonzero integers including 1. In Step 2, point multiplications dP for the integers $d \in B, d > 1$ are computed and stored. In the evaluation stage, the information obtained from these two steps is used to compute kP. Note that Algorithm 5.2 can be considered as a sliding window algorithm. In sliding window algorithms, the recoding process may be done in the evaluation stage. This happens when the recoding and evaluating processes scan the multiplier digits in the same direction. In this case, both the recoding and evaluating processes can be carried out simultaneously.

Algorithm 5.2 Random Point Multiplication (RPM)

Input: An integer k and a point $P \in E(\mathbb{F})$
Output: kP

Precomputation Stage:

1: Compute the B-representation of the multiplier $k = \sum_{0 \leqslant i < l} k_i 2^i$, $k_i \in B \cup \{0\}$
2: Compute and store dP for all integers $d \in B$, $d > 1$

Evaluation Stage:

3: $R \leftarrow \mathcal{O}$
4: **for** i from $l - 1$ down to 0 **do**
5: $\quad R \leftarrow 2R$
6: $\quad$ **if** $(k_i > 0)$ **then**
7: $\quad\quad R \leftarrow R + k_i P$
8: $\quad$ **else if** $(k_i < 0)$ **then**
9: $\quad\quad R \leftarrow R - (-k_i)P$
10: $\quad$ **end if**
11: **end for**
12: **return** R

The integer representation of the multiplier plays an important role in the performance of the RPM algorithm. Some useful integer representations are binary, NAF, and w-NAF for which $B = \{1\}$, $B = \{\pm 1\}$, and $B = \{\pm 1, \pm 3, \ldots, \pm(2^{w-1} - 1)\}$, respectively. If a binary representation is used in Algorithm 5.2, Steps 1 and 2 of the precomputation stage are not required. However, the evaluation stage would need $(\frac{l}{2} - 1)$ point additions on average. Using the NAF representation, only Step 1 of the precomputation stage is required. In this case, the average number of point additions required in the evaluation stage is reduced to $(\frac{l}{3} - 1)$. This is because

the average Hamming weight (the number of nonzero digits in the representation) of NAF is $\frac{l}{3}$. In fact, NAF has the minimal average Hamming weight among all B-representations with $B = \{\pm 1\}$. The NAF representation can be generalized to w-NAF. The w-NAF representation has the minimal average Hamming weight of $(\frac{l}{w+1})$ among all B-representations with $B = \{\pm 1, \pm 3, \ldots, \pm(2^{w-1} - 1)\}$ [67].

When w-NAF is used in Algorithm 5.2, it is a signed sliding window algorithm. In this case, the evaluation stage would require on average $(\frac{l}{w+1} - 1)$ point additions. However, Step 2 of the precomputation stage requires $(2^{w-2} - 1)$ point additions and one point doubling. In all these cases, the evaluation stage requires about $(l-1)$ point doublings. Note that $l = \lceil \log_2 N \rceil$ for the binary representation and $l = \lceil \log_2 N \rceil + 1$ for the NAF and w-NAF representations.

5.4.3 Extending RPM Algorithms

We extend RPM algorithms to the case where we need to multiply a base point P only once but we can have some limited extra information regarding P. For example, in variants of the Diffie-Hellman key agreement protocol, an entity A receives B's public key P_B and certificate, and is required to compute kP_B. In this case, B can provide A with some limited extra information by including it in its certificate. A trivial way to do this is to add a lookup table to the certificate. The entity A can use the lookup table together with a fixed point multiplication algorithm in order to compute kP_B faster. This approach may not be practical when the table is large. A simple solution to this problem is to make a small lookup table and embed it in the certificate. However, a better solution is to use a relatively large lookup table and add only a few selected points from it to the certificate. When properly selected, the points allow A to generate the entire lookup table relatively fast, and use it to speed up point multiplication.

For instance, assume that we can add at most three points in a certificate. In addition, suppose that Möller's algorithm is used as the fixed point multiplication algorithm. We can add the points

$$3P, (2^{\lceil \frac{l}{2} \rceil}P) \text{ and } 3(2^{\lceil \frac{l}{2} \rceil}P)$$

to the certificate in order to set $w = 3$ for k's w-NAF representation and avoid using a precomputation stage. In this case, we have the entire lookup

table and we can reduce the required number of point doublings by about 1/2. However, the number of point doublings can be reduced by about 1/4 when we only add the points

$$(2^{\lceil \frac{l}{4} \rceil}P), (2^{2\times\lceil \frac{l}{4} \rceil}P) \text{ and } (2^{3\times\lceil \frac{l}{4} \rceil}P).$$

To use the same w-NAF representation ($w = 3$) as in the previous case, A must compute the points

$$3P, 3(2^{\lceil \frac{l}{4} \rceil}P), 3(2^{2\times\lceil \frac{l}{4} \rceil}P) \text{ and } 3(2^{3\times\lceil \frac{l}{4} \rceil}P)$$

in the precomputation stage using 4 point additions and 4 point doublings. This is a small price to pay for a significant reduction in the required number of point doublings.

In the next section, we use this approach by adding the set of points

$$\mathcal{I}_{n-1} = \{2^{\lceil \frac{l}{n} \rceil \times 1}P, \ldots, 2^{\lceil \frac{l}{n} \rceil \times (n-1)}P\}$$

to the certificate. This redundant information can be stored as an extension of the certificate since $X.509$ version 3 [74] supports extensions. Note that including this information into the certificate does not significantly increase the certificate size for small values of n. For example, a typical size for an $X.509$ certificate is about $1K$ bytes [75]. For a 192-bit elliptic curve (as an example), the public key P requires $\lceil \frac{192\times 2}{8} \rceil = 48$ bytes. However, using a single point compression scheme, the point P could be represented using one 192-bit value and an additional bit. It then requires $\lceil \frac{192+1}{8} \rceil = 25$ bytes. Therefore, adding an extra point to the certificate would increase its original size by less than 5% and 2.5% when the points are represented in an uncompressed and a compressed format, respectively.

5.5 Partially Random Point Multiplication (PRPM)

As shown in Algorithm 5.3, Möller's algorithm can be used to compute random point multiplication kP using the redundant information $\mathcal{I}_{n-1}$. We refer to Algorithm 5.3 as Partially Random Point Multiplication (PRPM) algorithm since it performs a precomputation stage every time it is executed.

PRPM is an extension of RPM to the case where some limited extra information is available. Therefore, PRPM is expected to be faster than RPM at the cost of extra bandwidth. In this section, we speed up the PRPM algorithm using Montgomery's trick for simultaneous inversion and show that it is significantly faster than the RPM algorithm. We also show how to compute random point multiplication using multiple processors. Note that the proposed point compression schemes can be used to reduce the required bandwidth when single point compression is computationally expensive.

Algorithm 5.3 Partially Random Point Multiplication (PRPM)

Input: An integer k, the base point P and the set $\mathcal{I}_{n-1}$
Output: kP

Precomputation Stage:

1: Compute the B-representation of the multiplier $k = \sum_{0 \leqslant i < l} k_i 2^i$, $k_i \in B \cup \{0\}$
2: Compute and store $d(2^{\lceil \frac{l}{n} \rceil \times j} P)$ for all integers $d \in B$, $d > 1$ and $0 \leqslant j < n$

Evaluation Stage:

3: $s \leftarrow \lceil \frac{l}{n} \rceil$
4: $R \leftarrow \mathcal{O}$
5: **for** i from $(s-1)$ down to 0 **do**
6: $R \leftarrow 2R$
7: **for** j from $(n-1)$ down to 0 **do**
8: **if** $(k_{sj+i} > 0)$ **then**
9: $R \leftarrow R + k_{sj+i}(2^{sj}P)$
10: **else if** $(k_{sj+i} < 0)$ **then**
11: $R \leftarrow R - (-k_{sj+i})(2^{sj}P)$
12: **end if**
13: **end for**
14: **end for**
15: **return** R

5.5.1 Speeding up PRPM

In the precomputation stage of the PRPM algorithm, we must compute $d(2^{sj}P)$ for $d \in B$, $d > 1$ and $0 \leqslant j < n$. When w-NAF is employed, this computation can be accomplished in 2^{w-2} steps by computing the point $Q = 2P_{1,j}$ in the first step and the points $P_{i,j} = P_{i-1,j} + Q$ in the i-th step

for $0 \leqslant j < n$, where $P_{1,j} = 2^{sj}P$ and $2 \leqslant i \leqslant 2^{w-2}$. In the precomputation stage, it is preferable to represent the points $P_{i,j}$ using affine coordinates in order to reduce the amount of memory required to store them and to speed up the evaluation stage of the algorithm using mixed coordinates systems [76]. Using affine coordinates, we need n simultaneous field inversions in each step. Montgomery's trick for simultaneous inversion enables us to do this using one field inversion and $3(n-1)$ field multiplications [66]. Thus, for large n, one field inversion is replaced by approximately three field multiplications, which is a significant cost saving as a field inversion costs more than three field multiplications in most useful fields. Using this approach, we still need 2^{w-2} field inversions. The number of field inversions can be further reduced to $(w-1)$ in a second approach by computing the points $2P_j$ in the first step, the points

$$(2^{i-1}+1)P_j, (2^{i-1}+3)P_j, \ldots, (2^i-1)P_j, (2^i)P_j$$

in the i-th step, $2 \leqslant i < w-1$, and the points

$$(2^{w-2}+1)P_j, (2^{w-2}+3)P_j, \ldots, (2^{w-1}-1)P_j$$

in the last step, where $P_j = 2^{sj}P$ and $0 \leqslant j < n$.

In order to analyze the effects of using these approaches in more detail, let us restrict ourselves to elliptic curves defined over prime fields. If we use affine coordinates, the cost of the precomputation stage of Algorithm 5.3 would be

$$n2^{w-2}I + n2^{w-1}M + n(2^{w-2}+1)S.$$

Using Montgomery's trick for simultaneous inversion in the first approach, we can reduce the computation cost to

$$2^{w-2}I + (5n-3)2^{w-2}M + n(2^{w-2}+1)S,$$

thus saving the cost of

$$2^{w-2}(n-1)(I-3M).$$

The second approach has a computational cost of

$$(w-1)I + (5n(2^{w-2}+w-3) - 3w + 3)M + n(2^{w-2}+2w-5)S.$$

This can result in more savings for large values of I/M.

5.5.2 Performance Analysis of the PRPM Algorithm

In this section, we compare the performance of the PRPM and RPM algorithms. To do this, we consider the binary, NAF and w-NAF integer representations. A binary representation requires no recoding, thus it can be used in devices with very limited resources. The NAF representation can speed up point multiplication at the cost of recoding the integer, so it may be used in restricted devices in order to accelerate point multiplication. Finally, w-NAF ($w > 2$) can result in more speed up at the cost of recoding and storing some multiples of the base point. Therefore, w-NAF can be employed when some storage is available.

In the PRPM algorithm, the B-representation of the multiplier k is divided into n subblocks of length $s = \lceil \frac{l}{n} \rceil$. In the evaluation stage of this algorithm, the point multiplication kP is computed as

$$kP = \left(\sum_{0 \leqslant i < l} k_i 2^i \right) P = \sum_{0 \leqslant i < s} 2^i \left(\sum_{0 \leqslant j < n} k_{sj+i} (2^{sj} P) \right),$$

Therefore, the required number of point doublings at this stage is reduced to $(\lceil \frac{l}{n} \rceil - 1)$ while the required number of point additions remains the same as that with the RPM algorithm.

Let A be the cost of a point addition and D the cost of a point doubling. If a binary representation is used, the cost of the PRPM and RPM algorithms are approximately $(\frac{l}{2}A + lD)$ and $(\frac{l}{2}A + \frac{l}{n}D)$, respectively. Therefore, we have

$$R \approx \frac{\frac{l}{2}A + lD}{\frac{l}{2}A + \frac{l}{n}D} = \frac{\frac{1}{2} + t}{\frac{1}{2} + \frac{t}{n}},$$

where $R = \dfrac{\text{cost of Algorithm 5.2}}{\text{cost of Algorithm 5.3}}$ and $t = \frac{D}{A}$. Similarly, when the NAF representation is used we have

$$R \approx \frac{\frac{l}{3}A + lD}{\frac{l}{3}A + \frac{l}{n}D} = \frac{\frac{1}{3} + t}{\frac{1}{3} + \frac{t}{n}}.$$

Figure 5.1 compares the performance of the PRPM and RPM algorithms for the cases when binary and NAF representations are used, respectively. As shown in this figure, the higher the values of t and n, the better the performance improvement achieved by replacing the RPM algorithm with the PRPM algorithm. The value of t depends on the finite field and coordinate

system employed. For example, if the binary finite field and affine coordinates are used, we would have $t = 1$ (see Table 5.1).

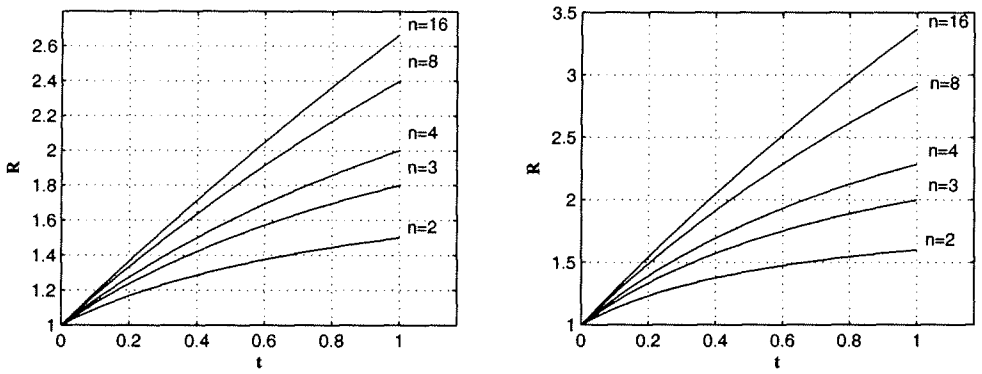

Figure 5.1: Performance comparison of Algorithms 5.2 and 5.3 when binary (left) and NAF (right) representations are used.

Now, consider the case where w-NAF ($w > 2$) is used by both the RPM and PRPM algorithms. In this case, comparing the performance of these two algorithms is more complex than the previous case since different coordinate systems may be used in the precomputation and the evaluation stages. Therefore, for a precise performance comparison of Algorithms 5.2 and 5.3, we restrict ourselves to elliptic curves defined over prime fields, particularly those recommended by the National Institute of Standards and Technology (NIST). In prime fields, the cost of a field squaring (S) can be approximated by $0.8M$ as the ratio S/M is almost independent of the implementation. In addition, I can be approximated to be between $9M$ and $30M$ for the case of $\mathbb{F}_p$, $p > 100$ [76]. Considering these assumptions, the optimum mixed coordinate systems for the RPM algorithm is to use affine coordinates in the precomputation stage, modified Jacobian coordinates (denoted by $\mathcal{J}^m$), and Jacobian coordinates (denoted by $\mathcal{J}$) in the evaluation stage [76]. The same mixed coordinate systems are used for the PRPM algorithm.

Tables 5.4 to 5.7 show the average cost of the RPM and PRPM algorithms for the NIST recommended elliptic curves. To get precise results and avoid complex equations, the average cost of the algorithms was obtained by simulation. For the PRPM algorithm, we restrict ourselves to $2 \leqslant n \leqslant 4$, as these values are more likely to be used in practice. In Table 5.4, the parameter w was selected to minimize the cost of the RPM algorithm. For the

PRPM algorithm, w was selected such that the required storage is less than or equal to that of the RPM algorithm. Therefore, the PRPM algorithm may result in more speed up when more storage is available. The fifth column of Tables 5.5 to 5.7 indicates the percentage increase in certificate size when the proposed point compression schemes are used.

	w	# stored points	Cost
P_192	5	8	$4I + 1789M$
P_224	5	8	$4I + 2084M$
	6	16	$5I + 2066M$
P_256	5	8	$4I + 2378M$
	6	16	$5I + 2351M$
P_384	6	16	$5I + 3492M$
P_521	6	16	$5I + 4714M$
	7	32	$6I + 4695M$

Table 5.4: Cost of the RPM algorithm.

	w	# Stored Points	Cost	Certificate Increase	Performance Improvement
P_192	4	8	$3I + 1180M$	2.5%	50%
P_224	4	8	$3I + 1371M$	2.8%	51%
	5	16	$4I + 1338M$		52%
P_256	4	8	$3I + 1562M$	3.2%	51%
	5	16	$4I + 1516M$		53%
P_384	5	16	$4I + 2230M$	4.6%	55%
P_521	5	16	$4I + 2994M$	6%	56%
	6	32	$5I + 2949M$		57%

Table 5.5: Cost of the PRPM algorithm for $n = 2$.

	w	# Stored Points	Cost	Certificate Increase	Performance Improvement
P_192	3	6	$2I + 1038M$	5.0%	$73 - 74\%$
P_224	4	12	$3I + 1128M$	5.6%	$81 - 83\%$
P_256	4	12	$3I + 1280M$	6.4%	$82 - 84\%$
P_384	4	12	$3I + 1888M$	9.2%	$84 - 85\%$
P_521	5	24	$4I + 2421M$	12%	$91 - 93\%$

Table 5.6: Cost of the PRPM algorithm for $n = 3$.

	w	# Stored Points	Cost	Certificate Increase	Performance Improvement
P_192	3	8	$2I + 931M$	10%	93%
P_224	3	8	$2I + 1081M$	11.2%	93%
	4	16	$3I + 1024M$		$99 - 101\%$
P_256	3	8	$2I + 1230M$	12.8%	94%
	4	16	$3I + 1155M$		$101 - 103\%$
P_384	4	16	$3I + 1682M$	18.4%	$105 - 107\%$
P_521	4	16	$3I + 2244M$	24%	$108 - 110\%$
	5	32	$4I + 2163M$		$114 - 116\%$

Table 5.7: Cost of the PRPM algorithm for $n = 4$.

5.5.3 Parallel Processing of the PRPM Algorithm

Using the redundant information $\mathcal{I}_{n-1}$, random point multiplication can be computed much faster with multiple processors. For example, assume that n processors are available and we have $\mathcal{I}_{n-1}$. Then kP can be computed as follows

$$\begin{aligned} kP &= \left(\sum_{i=0}^{l-1} k_i 2^i\right) P = \sum_{j=0}^{n-1} \left(\left(\sum_{i=0}^{s-1} k_{sj+i} 2^i\right) \left(2^{sj} P\right)\right) \\ &= \sum_{j=0}^{n-1} R_j, \end{aligned}$$

where $R_j = (\sum_{i=0}^{s-1} k_{sj+i} 2^i)(2^{sj} P)$. The j-th processor can then compute R_{j-1} using Algorithm 5.2. Hence, each processor requires s point doublings and in the worst case $((\frac{s}{w} - 1) + (2^{w-2} - 1))$ point additions when the w-NAF representation is used (note that w-NAF has at most $\frac{l}{w}$ nonzero digits). Then $\lceil \log_2 n \rceil$ point additions are required to compute the final result. Therefore the total cost of parallel computation of kP is $((\frac{s}{w} - 1) + (2^{w-2} - 1) + \lceil \log_2 n \rceil)A + sD$. Consequently, using the redundant information and parallel processing, random point multiplication can be computed about n times faster than Algorithm 5.2 when $n \ll l$ because

$$\begin{aligned} \frac{((\frac{l}{w+1} - 1) + (2^{w-2} - 1))A + lD}{((\frac{s}{w} - 1) + (2^{w-2} - 1) + \lceil \log_2 n \rceil)A + sD} &\approx \frac{\frac{l}{w+1}A + lD}{\frac{1}{n}(\frac{l}{w}A + lD)} \\ &\approx n. \end{aligned} \tag{5.7}$$

Note that for the algorithm used in parallel processing, the optimal w is generally smaller than that of Algorithm 5.2. However, we used the same w in (5.7) for computational convenience. In fact, using the optimal w for parallel processing results in a value closer to n than in (5.7).

Consider the general case where we have m processors ($m < n$) and we know $\mathcal{I}_{n-1}$. Random point multiplication can then be computed in the following (similar) way

$$\begin{aligned} kP &= \left(\sum_{i=0}^{l-1} k_i 2^i\right) P = \sum_{j=0}^{m-1} \left(\left(\sum_{i=0}^{s'-1} k_{s'j+i} 2^i\right) \left(2^{s'j} P\right)\right) \\ &= \sum_{j=0}^{m-1} R_j, \end{aligned}$$

where $s' = \lceil \frac{l}{n} \rceil \times \lceil \frac{n}{m} \rceil$ and $R_j = (\sum_{i=0}^{s'-1} k_{s'j+i} 2^i)(2^{s'j} P)$. Note that in this case, each processor uses Algorithm 5.3 instead of Algorithm 5.2 to compute R_j.

5.6 Summary

In this chapter, we proposed a novel double point compression technique and extended it to triple point compression. The compression process of the proposed schemes requires almost no computational effort, and decompression involves no square root operations. Therefore, they can be used to compress multiple points when single point compression is computationally expensive.

We introduced a new approach to compute random point multiplication in which redundant information is added to the certificate and used to speed up the multiplication. Using this approach, we showed that a significant speed up can be obtained at the cost of slightly higher bandwidth. We also showed that the required bandwidth can be reduced using the proposed point compression schemes. Note that a similar approach can be used to speed up double point multiplication [77] and exponentiation in hyperelliptic curve cryptography [78].

Bibliography

[64] N. Koblitz, "Elliptic curve cryptosystems," *Math. Comp.*, vol. 48, pp. 203–209, 1987.

[65] I. Blake, G. Seroussy, and N. Smart, *Elliptic Curves in Cryptography.* Cambridge: Cambridge University Press, 1999.

[66] H. Cohen, *A Course in Computational Number Theory, Graduate Texts in Math. 138.* Springer-Verlag, 1993, third Corrected Printing, 1996.

[67] J. A. Muir and D. R. Stinson, "Minimality and other properties of the with-w nonadjacent form," *Preprint*, 2004, Available from http://www.cacr.math.uwaterloo.ca/tech_reports.html.

[68] M. Khabbazian, T. A. Gulliver, and V. K. Bhargava, "A new minimal average weight representation for left-to-right point multiplication methods," *IEEE Trans. Computers*, vol. 54, pp. 1454–1459, 2005.

[69] B. Möller, "Improved techniques for fast exponentiation," *Springer-Verlag Lecture Notes in Computer Science*, vol. 2587, pp. 298–312, 2003.

[70] G. Seroussi, "Compact representation of elliptic curve points over $\mathbb{F}_{2^n}$," *Hewlett-Packard Laboratories Technical Report No. HPL-98-94R1*, 1998.

[71] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography.* CRC Press, 1997.

[72] C. H. Lim and P. J. Lee, "More flexible exponentiation with precomputation," *Springer-Verlag Lecture Notes in Computer Science*, vol. 839, pp. 95–107, 1994.

[73] D. Gordon, "A survey of fast exponentiation methods," *J. Algorithms*, vol. 27, pp. 129–146, 1998.

[74] ITU-T, *ITU-T Recommendation X.509 version 3 (1997)*, Information Technology - Open Systems Interconnection - The Directory Authentication Framewordk, ISO/IEC 9594-8, 1997.

[75] R. Zuccherato, "Elliptic curve cryptography support in entrust," Tech. Rep., May 9, 2000.

[76] H. Cohen, A. Miyaji, and T. Ono, "Efficient elliptic curve exponentiation using mixed coordinates," *Springer-Verlag Lecture Notes in Computer Science*, vol. 1514, pp. 51–65, 1998.

[77] M. Khabbazian, T. A. Gulliver, and V. K. Bhargava, "A new technique for improving the speed of double point multiplication," *IEEE Pacific Rim Conf. on Commun., Computers and Signal Processing*, pp. 653–656, 2005.

[78] N. Koblitz, "Hyperelliptic cryptosystems," *Journal of Cryptology*, vol. 1, pp. 139–150, 1989.

Chapter 6

Summary and Conclusion

This work consists of four major breakthroughs in the area of wireless ad hoc networks. The first two, presented in Chapters 2 and 3, deal with broadcasting, a fundamental operation in wireless ad hoc networks. In Chapter 2, we present two efficient broadcast algorithms based on 1-hop neighbor information. The first algorithm offers great improvements over one of the best neighbor-designating broadcast algorithms based on 1-hop neighbor information [79]. In their paper [79], Liu *et al.* proposed a neighbor-designating broadcast algorithm that can achieve local optimality by selecting the minimum number of forwarding nodes in the lowest computational time complexity $O(n \log n)$, where n is the number of neighbors. We show that this optimality only holds for a subclass of neighbor-designating algorithms. In fact, we prove that our first broadcast algorithm can reduce the time complexity of computing forwarding nodes to $O(n)$. The second improvement of our first broadcast algorithm is on reducing the maximum number of selected nodes. In algorithm of Liu *et al.*, n nodes are selected to forward the message in the worst case, whereas in our proposed algorithm, the number of forwarding nodes in the worst case is 11. This nice feature provides the capability of bounding packets overhead. Our second broadcast algorithm, presented in Chapter 2, was proposed with the objective of significantly reducing the number of transmissions. We show that our second proposed broadcast algorithm is very successful in achieving this goal for the case where nodes are distributed randomly. However, we show that the algorithm cannot provide a constant approximation ratio to the optimal solution (minimum number of required transmissions) in the worst case. This directed us to our next work, presented in Chapter 3, in which we investigated whether localized broadcast algorithms are able to provide both full delivery and a good approximation ratio to the optimal solution.

In Chapter 3, we first show that many of the existing neighbor-designating broadcast algorithms based on 1-hop neighbor information are not able to provide both full delivery and a good bound on the optimal solution. We also

show that this is true for many self-pruning broadcast algorithms based on 1-hop neighbor information as well. These results are in favor of a common belief that localized broadcast algorithms for ad hoc networks are not able to achieve full delivery and guarantee a bounded number of transmissions in the network. One of the breakthrough results, presented in Chapter 3, is showing for the first time this belief is unfounded by designing an efficient broadcast algorithm that satisfies both aforementioned properties. In fact, we prove that an extension of our second algorithm proposed in Chapter 2 can achieve both full delivery and a constant approximation ratio to the optimum solution. In addition, we prove that the message complexity of the extended algorithm is $O(N)$, where N is the total number of nodes in the network. This is also a very interesting result because the lower message complexity bound of finding CDS, a closely related problem, is proven to be $\Omega(N \log N)$. Chapter 3 also includes other interesting contributions such as:

- Proposing a nearly optimal algorithm to verify the responsibility condition
- Removing nodes' IDs from the broadcast messages in order to reduce bandwidth overhead
- Representing location information using only a constant number of bits
- Replacing the list of neighbors of the broadcasting node (piggybacked in the packet) with a significantly smaller subset of it with the objective of further reduction in bandwidth overhead
- Relaxing several system model assumptions or replacing them with practical ones:
  - Distributing nodded in 3-D instead of 2-D
  - Broadcasting under uncertain position information
  - Relaxing the Homogeneous Network Assumption

The other two major breakthrough of this work, presented in Chapters 4 and 5, deal with security issues in wireless ad hoc networks. In Chapter 4, we first analyze the effect of wormhole attack, one of the most severe attacks in ad hoc networks. We first draw a precise picture of the effect of the wormhole attack in shortest path routing protocols. We then show, in contrast to the

common belief, the wormhole attack launched by two malicious nodes cannot disrupt/control the majority of communications, on average. However, we show that the attackers can disrupt/control about one-third of all the communications if the wormhole is placed strategically in the network. We also analyze a scenario in which several attackers make wormholes among each other and a case where two malicious nodes attack a target node in the network. We show how to evaluate the maximum effect of the wormhole attack on a given network topology. Then, we compute the maximum effect of the wormhole attack on grid topology networks and show that the attackers can disrupt/control around 40% to 50% of all communications when the wormhole is strategically placed in the network. To defend against the wormhole attack, we propose a timing-based countermeasure. The existing timing-based solutions typically suffer from some of the following shortcomings:

- The nodes require tightly synchronized clocks
- Each node has to be capable of fast switching between the receive and send modes
- Each node needs one-to-one communication with all its neighbors
- Each node requires predicting the sending time

We show that our proposed timing-based countermeasure does not have any of these shortcomings, thus is more suitable for practical implementation of solutions based on packet travel time measurements.

As the continuation of our work on security of ad hoc networks, we studied elliptic curve cryptography, which is one the primary public-key candidates to be used in ad hoc networks. We propose two point compression schemes, in Chapter 5, that can be used to reduce memory/bandwidth required to store/transmit elliptic curve points. The compression using our schemes can be done with almost no effort. In contrast to single point compression scheme, patented by Certicom Corporation, decompression of our schemes does not require any square root operation. Therefore, our schemes can be used, rather than single point compression scheme, when square root operation is computationally expensive.

All the results presented in this work can be used to improve the efficiency of the wireless ad hoc network operations or to make the network

more secure. As future work, it would be interesting to show whether localized broadcast algorithms can achieve both full delivery and a good bound on the optimal solution without using location information. Also, it is worth analyzing the effect of the wormhole attack in routing protocols not based on shortest path. In Chapter 5, we showed that the wormhole attack countermeasure presented in [80] is not able to detect the wormhole if the attackers selectively forward the packets. We claimed that this problem can be mitigated when our proposed timing-based solution is used. It is of interest to improve the proposed countermeasure in [80] to resolve this issue without using any specialized hardware such as accurate clocks (note that timing-based solutions require accurate clocks).

Bibliography

[79] H. Liu, P. Wan, X. Jia, X. Liu, and F. Yao, "Efficient flooding scheme based on 1-hop information in mobile ad hoc networks," *In Proc. of IEEE INFOCOM*, 2006.

[80] R. Maheshwari, J. Gao, and S. R. Das, "Detecting wormhole attacks in wireless networks using connectivity information," *In Proc. of IEEE INFOCOM*, 2007.

Appendix A

List of Publications

Journal papers (accepted/published)

- **M. Khabbazian**, Md. Jahangir Hossain, and V. K. Bhargava, "Exact Method for the Error Probability Calculation of Three-Dimensional Signal Constellations," *accepted for publication IEEE Transactions on Communications*, 2008.

- H. Mercier, **M. Khabbazian** and V. K. Bhargava, "On the Number of Subsequences when Deleting Symbols from a String," *accepted for publication in IEEE Transactions on Information Theory*, 2008.

- **M. Khabbazian** and V.K. Bhargava, "Localized Broadcasting with Guaranteed Delivery and Bounded Transmission Redundancy," *accepted for publication in IEEE Transactions on Computers*, 2008.

- **M. Khabbazian**, H. Mercier and V. K. Bhargava, "Severity Analysis and Countermeasures for the Wormhole Attack in Wireless Ad Hoc Networks," *accepted for publication in IEEE Transactions on Wireless Communications*, 2008.

- **M. Khabbazian** and V.K. Bhargava, "Highly Efficient Broadcasting in Mobile Ad Hoc Networks," *accepted for publication in IEEE Transactions on Mobile Computing*, 2007.

- **M. Khabbazian**, T. A. Gulliver and V. K. Bhargava, "Double Point Compression with Applications to Speeding up Random Point Multiplication," *IEEE Transactions on Computers*, vol. 56, no. 3, pp. 305-313, Mar. 2007.

- **M. Khabbazian**, T. A. Gulliver and V. K. Bhargava, "A New Minimal Average Weight Representation for Left-to-Right Point Multiplication Methods," *IEEE Trans. Computers*, vol. 54, No. 11, pp. 1454-1459, Nov. 2005.

Journal papers (submitted)

- K.K. Leung, C. W. Sung, **M. Khabbazian** and M. A. Safari, "Optimal Phase Control in MIMO Systems with Quantized Feedback," *submitted to IEEE Transactions on Information Theory,* Mar. 2008.

Selected conference papers

- P. Kaligineedi, **M. Khabbazian**, and V. K. Bhargava, "Secure Cooperative Sensing Techniques for Cognitive Radio Systems," *IEEE International Conference on Communications (ICC),* May 2008.

- **M. Khabbazian** and V.K. Bhargava, "Reducing Broadcast Redundancy in Wireless Ad Hoc Networks," *IEEE Global Communications Conference*, Nov. 2007.

- **M. Khabbazian**, H. Mercier and V. K. Bhargava, "Wormhole Attack in Wireless Ad Hoc Networks: Analysis and Countermeasures," *in Proc. IEEE Global Communications Conference*, Nov. 2006.

- **M. Khabbazian**, K-K. Leung and M. A. Safari, "On the Optimal Phase Control in MIMO Systems with Phase Quantization," *in Proc. IEEE International Communications Conference*, vol. 9, pp. 4272-4277, Jun. 2006.

- M. M. Rashid, E. Hossain, **M. Khabbazian**, and V. K. Bhargava, "On Access-Based Self-Organized Clustering in Ad Hoc Mobile Wireless Networks," *in Proc. IEEE Conference on Communication System Software and Middleware (COMSWARE)*, Jan. 2006.