Efficient neighbour discovery and route request flooding schemes for ad hoc networks with directional antennas

Santosa, Rully Adrian

2009

Santosa, R. A. (2009). Efficient neighbour discovery and route request flooding schemes for ad hoc networks with directional antennas. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/14960>

<https://doi.org/10.32657/10356/14960>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**EFFICIENT NEIGHBOR DISCOVERY AND
ROUTE REQUEST FLOODING SCHEMES
FOR AD HOC NETWORKS
WITH DIRECTIONAL ANTENNAS**

**RULLY ADRIAN SANTOSA
SCHOOL OF COMPUTER ENGINEERING
2009**

EFFICIENT NEIGHBOR DISCOVERY AND ROUTE REQUEST FLOODING SCHEMES FOR AD HOC NETWORKS WITH DIRECTIONAL ANTENNAS

RULLY ADRIAN SANTOSA

SCHOOL OF COMPUTER ENGINEERING

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Master of Engineering

2009

Acknowledgements

I would like to thank The Lord for His abundant blessings that have given me strength and perseverance to complete this research project.

I would like to sincerely thank my supervisor Assoc. Prof. Lee Bu Sung, Francis for his invaluable guidance and constant supervision throughout the course of my master study, for providing me with opportunities to develop my knowledge and research skills that will certainly be beneficial for my future career.

I also would like to express my gratitude to my co-supervisor Assoc. Prof. Yeo Chai Kiat for introducing me to the research project, for her guidance, advice, patience, and encouragements during the sunny days as well as the rainy days.

I am also grateful to Dr. Lim Teck Meng and Dr. Juki Wirawan Tantra for their assistance, knowledge sharing and supports, as well as for the discussions that have always inspired me to make progress in my research.

Finally, I would like to express my gratitude to my family and friends for their endless supports, encouragements, and prayers.

Contents

Acknowledgements	3
Contents	4
List of Figures	7
List of Tables	10
Abstract	11
1 Introduction	13
1.1 Background	13
1.2 Research Objectives	15
1.3 Thesis Organization	15
2 Literature Review	17
2.1 Directional Antennas	17
2.2 Neighbor Discovery in Ad Hoc Networks with Directional Antennas	22
2.3 Routing in Ad Hoc Networks with Directional Antennas	25
2.4 Broadcasting in Ad Hoc Networks with Directional Antennas	29
3 Distributed Neighbor Discovery in Ad Hoc Networks with Directional Antennas	32
3.1 Introduction	32
3.2 Definitions	33

3.3	Neighbor Discovery Schemes	35
3.3.1	Active Neighbor Discovery Schemes	35
3.3.2	Passive Neighbor Discovery Schemes	36
3.4	The Proposed Scheme	37
3.4.1	Overview	37
3.4.2	The Algorithm	39
3.4.3	Example	43
3.5	Coverage Analysis	46
3.6	Coverage Evaluation	49
3.7	Comparative Performance Evaluation	50
4	DSR Route Request Flooding in Ad Hoc Networks with Direc-	
	tional Antennas	58
4.1	Introduction	58
4.2	The Proposed Algorithm	60
4.3	Fixed Branching Strategy	66
4.4	Honeycomb Strategy	73
4.4.1	Hexagon Overview	73
4.4.2	Honeycomb Selection Algorithm	75
4.4.3	Example	82
4.4.4	Why Hexagon?	90
4.5	Performance Comparison	93
4.5.1	Simulation Setup	94
4.5.2	Simulation Results	95
5	Conclusions and Future Works	101
5.1	Conclusions	101
5.2	Future Works	104
	Author's Publications	107

Glossary	108
Bibliography	110

List of Figures

3.1	Range and area definitions	33
3.2	Minimum coverage	41
3.3	Example neighbor discovery scenario	45
3.4	Expected coverage $c(n)$ against the number of omnidirectional neighbors	47
3.5	Expected area not covered by the proposed algorithm against node density	48
3.6	Actual not-covered area obtained from simulations vs theoretical not-covered area	50
3.7	Average number of representative nodes against node density . . .	51
3.8	Scenario 1 for comparative performance evaluation	52
3.9	Scenario 2 for comparative performance evaluation	52
4.1	A scenario where the destination node is a direct neighbor of the source node	63
4.2	Initial representative node selection	64
4.3	The source node chooses a representative node in one of the predefined directions. Node B is the farthest neighbor of Node A that falls in the sector corresponding to the predefined direction, and therefore is chosen as the representative node for the direction. . .	66

4.4	The intermediate node determines the forwarding directions. The forwarding directions are calculated with respect to the direction from the source node to the intermediate node, not to the incoming direction of the route request packet.	68
4.5	Fixed branching strategy with $TTL = 3$	70
4.6	Fixed branching strategy with $TTL = 8$	71
4.7	An example of poor coverage. The initial representative nodes selection uses 3 directions. The next representative nodes selection uses relative directions: -30° , 0° , and 30°	72
4.8	An example of poor coverage. The initial representative nodes selection uses 4 directions. The next representative nodes selection uses relative directions: -15° , 0° , and 15°	72
4.9	A regular hexagon with 30° , 90° , 150° , 210° , 270° , or 330° orientation	74
4.10	An example showing how to locate the hexagon with hexagonal coordinate $(3, 2)$ in the hexagonal tiling.	75
4.11	Virtual hexagonal tiling constructed over an area. The dot in the center is the source node.	76
4.12	Dividing the virtual hexagonal tiling into 6 zones, each approximately forms a 60° sector.	78
4.13	Flow directions assigned to the hexagon edges (left). The simplified representation of the left hexagon using a single arrow inside the hexagon (right).	78
4.14	The assignment of the hexagonal flow direction to the hexagons. All hexagons in the same zone are assigned the same hexagonal flow direction.	79
4.15	The pattern of the route request flow	80
4.16	Route request flooding using the honeycomb strategy ($TTL = 5$) .	83
4.17	Route request flooding using the honeycomb strategy ($TTL = 10$)	83
4.18	Initial representative nodes selection	85

4.19 Mapping the ideal locations of the representative nodes to the ac-	
tual neighbor nodes	86
4.20 Node A in the virtual hexagonal tiling	87
4.21 Ideal location of node A	88
4.22 Next representative nodes selection	89
4.23 Mapping the ideal locations of the representative nodes to the ac-	
tual neighbor nodes	90
4.24 Octagon-and-square tiling	91
4.25 Triangular tiling and square tiling	92
4.26 Coverage overlaps for different types of polygons	93
4.27 The route discovery success rate	95
4.28 The average number of hops in the route	97
4.29 The average number of nodes involved in the route discovery process	98
4.30 The average number of transmissions during the route discovery	
process	99

List of Tables

3.1	Performance comparison of neighbor discovery algorithms	54
4.1	Percentage of the area inside the polygon covered by certain number of representative nodes	92

Abstract

The use of directional antennas in ad hoc networks gains more popularity in recent years. Directional antenna provides a node with the ability to transmit over longer distance with a transmission beam that is narrowly focused in one particular direction. When used properly, directional antennas offer the potential of increasing the network connectivity and the network bandwidth, while reducing the number of hops and the interference region.

Neighbor discovery plays a significant role in ad hoc networks with directional antennas. To efficiently use the directional antenna, a node needs to know the direction of the intended recipient to where it will form the transmission beam. To fully benefit from the longer transmission range of the directional antenna, the neighbor discovery scheme should be able to provide the node with the information about the neighbors beyond the omnidirectional transmission range.

In this thesis, an efficient distributed neighbor discovery scheme for ad hoc networks with directional antennas is proposed. The proposed scheme aims to discover neighbors beyond the omnidirectional transmission range by querying a small subset of the neighbors in the omnidirectional transmission range. It provides a fast neighbor discovery for a node that is joining an existing system. The use of high power broadcast is not required in the discovery process.

The information provided by the neighbor discovery scheme can also be exploited by higher layer protocols for other purposes such as for making a better routing decision. In ad hoc networks, many reactive routing protocols perform the route discovery by flooding the route request packets throughout the network.

This flooding process usually consumes the network bandwidth considerably and involves a lot of nodes within certain hops from the source node.

In this thesis, an efficient scheme for flooding the route request packets in ad hoc networks with directional antennas is also proposed. The scheme exploits the local neighbor information provided by the neighbor discovery scheme at the lower layer. In the proposed scheme, route request packets are forwarded only to selected representative neighbors using directional transmissions instead of being broadcast to all neighbors. Two strategies are proposed to effectively select the representative neighbors. The fixed branching strategy selects one representative neighbor in each predefined forwarding direction, while the honeycomb strategy selects the representative neighbors to create a hexagonal tiling pattern. Both strategies are able to achieve a high route discovery success rate while keeping the number of involved nodes and the number of transmissions low.

Chapter 1

Introduction

1.1 Background

Ad hoc network has become one of the major research interest in the wireless communication area. Its ability to form a self-organizing network without the existence of a central controller has made it attractive for different applications that require fast deployment at low cost. Due to the lack of central controller, ad hoc networks have to be able to work and organize themselves in a distributed manner.

The use of directional antennas in ad-hoc networks has received great attention in recent years. Directional antennas are able to focus the transmission power in one particular direction, resulting in longer transmission range and localized interference region. The benefits of directional antennas include reducing the number of hops to reach the destination node, reducing the occurrences of network partitions and increasing the network bandwidth due to spatial reuse.

With the use of directional antennas in wireless ad hoc networks, the role of neighbor discovery becomes significant. Unlike omnidirectional antennas that simply transmit into all directions irrespective of the location of the recipient, a node with directional antenna has to know the location of the intended recipient to form the transmission beam in the direction of the recipient. Neighbor discov-

ery provides the node with the information about the neighbor nodes around it. The neighbor information typically includes the identifier and the location of the neighbor nodes. Basic neighbor discovery schemes that are originally designed to support omnidirectional transmissions are not sufficient when directional transmissions are used. Typically, these neighbor discovery schemes are only able to discover neighbors within the omnidirectional transmission range. To fully benefit from the longer transmission range of directional antennas, the neighbor discovery schemes should be able to provide the node with the information about the neighbor nodes beyond the omnidirectional transmission range.

A node joining an existing system initially needs to discover other nodes in the system before establishing communication with them in the future. Information about the existence and the location of neighbors is important to facilitate a node in making routing and other decisions. It is therefore important for a node to get as much neighbor information as possible shortly after the node joins the existing system. The faster the node gets the neighbor information, the sooner the node is able to start communicating with other nodes in the network efficiently.

When performing its function, neighbor discovery consumes a considerable amount of resources. Therefore, it is highly desirable if the information provided by the neighbor discovery can also be exploited by higher layer protocols, e.g. the routing protocols, in the hope of reducing the resource consumption by those higher layer protocols.

Reactive routing protocols such as the Dynamic Source Routing (DSR) protocol perform route discovery as needed by the source node. During the route discovery, route request packets are flooded throughout the network from the source node using omnidirectional broadcast. Each node receiving the route request packet broadcasts it further using omnidirectional transmission as well. Effectively, all nodes within certain numbers of hops from the source node, depending on the TTL value in the route request packets, are involved in the route discovery process. Therefore, route request flooding can be considered as one of the

processes in wireless ad hoc networking that consume a considerable amount of resources.

1.2 Research Objectives

The general objective of this research project is to explore the use of directional antennas to increase the efficiency of wireless ad hoc networks. This general objective is achieved through the following specific objectives:

- to explore and develop a fast neighbor discovery scheme that exploits the benefits of directional antennas
- to explore and develop a route request flooding scheme that makes use of the information provided by the neighbor discovery scheme and exploits the benefits of directional antennas

Directional MAC protocol is currently a rather hot research area. Therefore, this research project does not attempt to jump directly into this hot area. Instead, this research project tries to differentiate itself by focusing on other aspects of exploiting directional antennas in ad hoc networks that are as important but are currently less researched.

1.3 Thesis Organization

The rest of the thesis is organised into the following chapters.

Chapter 2 presents an overview of directional antennas, followed by a discussion on the existing works in the area of neighbor discovery in ad hoc networks with directional antennas. The discussion continues with the existing works in the area of routing in ad hoc networks with directional antennas. The last section of the chapter discusses the existing works in the area of broadcasting/flooding in ad hoc networks with directional antennas.

Chapter 3 presents a detail discussion on the proposed neighbor discovery schemes in ad hoc networks with directional antennas. The proposed algorithm together with some examples will be explained. The coverage of the proposed neighbor discovery scheme will be analysed, and the analysis will be compared with the simulation results. A comparative performance evaluation of the proposed scheme against other schemes will also be presented.

Chapter 4 presents a detail discussion on the proposed route request flooding mechanism in ad hoc networks with directional antennas. The general algorithm will be explained, followed by two specific strategies that can be used in the algorithm. The comparative performance evaluation of the proposed algorithm against other schemes will be presented.

Finally, chapter 5 summarizes the concepts, the analysis, and the results that have been proposed in this thesis. The second part of the chapter discusses some possible future works to develop the ideas proposed in this thesis further.

Chapter 2

Literature Review

2.1 Directional Antennas

Directional antennas are broadly defined as antennas that are capable to maximize the transmission power in one particular direction and minimize the transmission power in the other directions. Unlike omnidirectional antennas that have a uniform radiation pattern in all directions, directional antennas have a higher gain in one particular direction. Directional antennas form transmission beams that allow nodes in the direction of the beam to receive the transmission while nodes in the other direction remain unaware of the ongoing transmission.

One specific kind of directional antennas with additional intelligence embedded in it is known as smart antennas. Smart antennas have the capability to dynamically adjust the antenna radiation pattern to form a beam towards a specific direction. Smart antennas consist of a number of radiating elements that are geometrically arranged, e.g. linear, circular, or grid, and controlled by a digital signal processing unit. The digital signal processing unit controls the phase and the relative amplitude of the signals transmitted by each radiating element. When the signals from different radiating elements are combined together, the result is a radiation pattern with a high gain in one particular direction and a low gain in the other directions.

Based on its capability and complexity, there are three types of smart antennas [1]:

- *Switched Lobe*

The beam directions are predefined. The smart antenna switches between these predefined beams to choose the one yielding the maximum received power. This type of smart antenna is the least complex and the easiest to implement.

- *Dynamically Phased Array*

The direction of arrival algorithm is used to continuously track the received signal and dynamically steer the beam into that direction that maximize the received power.

- *Adaptive Array*

The directions of arrival of the interference sources are also tracked. The antenna gain pattern is adjusted to null out the interferers. The pattern can also be adapted to receive multipath signals. The Signal to Interference and Noise Ratio (SINR) is maximized. This type of smart antenna is the most complex.

There are several advantages of directional antennas that make them attractive for use in the wireless ad-hoc networks:

- *spatial reuse*

Omnidirectional antennas spread the transmission power to all directions although only a small portion of the transmitted power is received by the intended destination. As a result, neighbor nodes within the range of the omnidirectional transmissions must keep silent in order not to interfere with the ongoing transmission. Using directional antennas, the sender node transmits the power only in the direction of the destination node and causes little interference in the other directions. Neighboring nodes can simultaneously

transmit in different directions as long as the transmissions do not interfere with each other. As the number of possible concurrent transmission increases, the capacity of the overall system also increases.

- *higher transmission range*

As directional antenna concentrates the transmission power in a specific direction, the gain of the antenna in that particular direction increases. As a result, the transmission is able to reach far beyond the omnidirectional transmission range with the same transmission power. This brings about the possibility to communicate with distant nodes using fewer number of hops, and thereby reducing the end-to-end delay. In addition, network partitions are also reduced.

- *higher link quality*

As more power is transmitted in the direction of the receiver node, the link quality is expected to be higher.

- *reduced chances of eavesdropping*

With the smaller transmission angle of directional antennas, the chances of eavesdropping are reduced. One attempting to eavesdrop on an ongoing communication must be located in the direction where the transmission beams are pointing to.

Some issues found in omnidirectional antennas due to the restrictions and impairments of the wireless environment also apply to directional antennas. These issues include the half duplex operation of the transceivers, the time-varying, unpredictable and unreliable wireless channel, frequent and bursty channel errors, and location dependent carrier sensing [2]. In addition to these issues, there are some new challenges arising from the use of directional antennas.

- *neighbor discovery*

The role of neighbor discovery becomes more significant with the use of

directional antennas. A node intending to transmit data to one of its neighbors must know the location of the intended recipient in order to form the transmission beam in the direction of the recipient. Some proposed directional MAC protocols include the neighbor discovery scheme as part of the protocol.

- *directional hidden terminal*

One of the classic problems in wireless network is the hidden terminal problem. The IEEE 802.11 DCF resolves this problem by the exchange of RTS/CTS control packets. The use of smart antennas introduces two new types of hidden terminal problem [3].

Hidden terminal due to asymmetry in gain occurs when the transmission antenna gain is not the same as the reception antenna gain. This may occur when transmissions are performed directionally while receptions are performed omnidirectionally. As the directional antenna gain is usually much higher than the omnidirectional antenna gain, the asymmetry in gain exists. A node may sense the channel as idle when it is in omnidirectional mode. But when it transmits in directional mode, its transmission may reach some communicating nodes and interfere with the ongoing communication.

Hidden terminal due to unheard RTS/CTS occurs when a node does not hear the RTS/CTS transmission from other nodes because it is currently engaged in a communication with another node. When the node finishes communicating with the other node, it may attempt to initiate a transmission in the direction of the nodes the RTS/CTS of which were unheard since its DNAV table indicates that it is free to transmit in that direction. Consequently, as soon as the node transmits, a collision occurs.

- *deafness*

The problem of deafness occurs when the transmitter fails to communicate with the intended receiver because the antenna beam of the receiver

is pointing towards a direction away from the transmitter [4]. The receiver may be pointing to that direction due to it being engaged in a communication with another node. Upon realizing the communication failure, the transmitter backs off and retries the transmission after a random interval chosen from the range $[0, CW]$. The transmitter may attempt multiple retransmissions while the receiver is still engaged in the communication with the other node. Each retransmission is preceded by an exponentially longer backoff interval. When finally the receiver node finishes its communication with the other node, it is very likely that the transmitter is counting down a large backoff value now, thus wasting the channel bandwidth.

The situation gets worse if now the receiver node has a packet to send. The receiver will choose a random backoff value before transmitting the packets, and this backoff value is highly probable to be smaller than the backoff value of the transmitter that attempts to communicate with this receiver. As a result, the transmitter may fail to communicate with the receiver again. After a number of unsuccessful attempts, the transmitter will eventually drop the packet. Higher layer may react to the packet drops by triggering the congestion control algorithm or running the rerouting algorithm. This unnecessary congestion control and rerouting will deteriorate the performance significantly.

The deafness problem leads to excessive packet drops, large delay variances, and unfairness of the channel use.

MAC protocols originally designed for omnidirectional antennas are not suitable for ad hoc networks with directional antennas. To fully exploit the benefits of directional antennas and to overcome the new challenges introduced by directional antennas, new directional MAC protocols have been proposed by various researchers. Most of the proposed directional MAC protocols are extensions to the existing MAC protocols for omnidirectional antennas. However, a small number

of proposals try to develop novel MAC protocols that are not based on existing protocols. As the development of MAC protocols for directional antennas is not the main focus of this thesis, the topic will not be discussed further. A good survey and overview of recently proposed MAC protocols for directional antennas can be found in [2].

2.2 Neighbor Discovery in Ad Hoc Networks with Directional Antennas

Periodic HELLO protocol can be considered as the fundamental and commonly used neighbor discovery scheme in wireless networks. It has been used as part of several common protocols for ad hoc networks. HELLO broadcast is used in the AODV routing protocol [5] for maintaining network connectivity. In AODV, a node may broadcast a HELLO message if it has not sent any broadcast message within a specified period. In [6], an implementation of the AODV protocol is utilized to study the effectiveness of HELLO messages for determining local connectivity and to examine different approaches for improving the accuracy of HELLO message as an indicator of local connectivity.

The fundamental idea of periodic HELLO protocol is the broadcasting of HELLO messages by nodes in the system periodically. A node that receives HELLO message from its neighbor is said to have *discovered* the neighbor, and thus records the information about the neighbor in its neighbor list or updates the information if it has already have one. Each record in the neighbor list is associated with a timeout, after which the record will be removed from the neighbor list.

Performance analysis of the basic periodic HELLO protocol has been conducted by researchers and improvements to the basic periodic HELLO protocol have been proposed. One of the main issues with the basic periodic HELLO protocol is the network congestion created by the HELLO packets. In [7], three

variations of the basic periodic HELLO protocol are proposed to reduce network congestion. The main idea is to beacon as minimum as possible without compromising the accuracy of the neighbor list. In the first variation, a node broadcasts HELLO messages with a rate proportional to the speed of the node, thus it is suitable for mobility scenario. In the second variation, a node broadcasts HELLO request message and the neighbors respond with HELLO reply messages. As the node only establishes the neighbor list when necessary, message overhead, and therefore congestion, is reduced. In the third variation, HELLO messages are sent only when the node detects activity within a time period.

In [8], the impact of interferences on the neighbor discovery process using HELLO protocol is studied. On average, in the presence of interferences, a node discovers only a subset of its neighbors. An improvement of the HELLO protocol by introducing sleep periods to reduce the energy consumption and the interference is proposed.

A variation of HELLO protocol inspired by the *birthday paradox* is proposed in [9]. It is claimed that the proposed probabilistic protocol saves a great deal of energy during deployment while allowing high probability of discovering neighbors. In [10], other probabilistic protocols for node discovery in single broadcast channel networks are proposed. This work is extended in [11] to study node discovery in multichannel broadcast networks.

Some neighbor discovery algorithms that exploit the benefit of directional antennas have been proposed. Most of the proposed neighbor discovery algorithms run at the MAC layer.

A polling-based neighbor discovery algorithm is proposed in [12]. A node sends polling message in a specific direction and waits for replies from the neighbors in that direction. Polling can be contention-based or contention-free. The space is successively scanned until the entire space has been covered. The node will have complete knowledge of its neighbors after it finishes polling in all directions.

A similar scheme based on the sequential scanning of space is proposed in [13].

Nodes are synchronized with each other and periodically perform neighbor discovery. During neighbor discovery, a node can either be in scanning or in listening mode. The selection of the mode can be stochastic or deterministic. A scanning node scans the space in a predetermined sequence and transmits advertisements. A listening node listens in the directions opposite to the scanning directions and discovers a scanning node when both nodes form beams towards each other.

In [14], a node transmits its identifier and location in random direction with probability p and listens for transmissions with probability $1 - p$. A neighbor node is discovered upon receiving a successful transmission from that neighbor node. Depending on p , the algorithm needs varying time duration to obtain complete knowledge of the neighbors. Neighbor discovery can be synchronous (time-slotted) or asynchronous (non time-slotted). An enhancement based on gossip is also proposed. In this algorithm, a node also transmits the identifiers and locations of the neighbors it has accumulated so far.

An integrated neighbor discovery and MAC protocol is proposed in [15]. Time is segmented into consecutive frames and nodes are synchronized with each other. Neighbor discovery is performed in the search slot at the start of each frame. The search slot is divided into two sub-slots. In the first sub-slot, each node randomly chooses whether to transmit pilot tone or to receive, together with the direction for transmission/reception. If the node chooses to transmit pilot tone in the first sub-slot, it will receive in the second sub-slot, and vice versa. Two nodes discover each other upon successful exchange of pilot tones.

In [16], a synchronized directional neighbor discovery is proposed. It aims to get two nodes that do not know each other's existence to beamform towards each other simultaneously. A direction is chosen based on the synchronized time. Each node alternates randomly between transmitting in that direction and listening in the opposite direction. When the transmit beamform of a node aligns with the receive beamform of another node, the two nodes can discover each other. A complete knowledge of the neighbors is available after one such cycle covering all

directions.

2.3 Routing in Ad Hoc Networks with Directional Antennas

A number of works in the area of improving the performance of the routing protocols for ad-hoc networks have previously been carried out by various researchers. Each research work focused on different areas of improvements. Of various routing protocols for ad-hoc networks, the Dynamic Source Routing (DSR) [17] is one of the protocol being most extensively researched. Being an on-demand protocol, DSR discovers and maintains route reactively as needed by the data packets. It has a lower routing overhead as compared with the shortest-path based protocols that need to update the routing tables periodically by exchanging control packets. However, as the route is discovered only when needed, there is some delay incurred before the data packet can be sent to the destination.

The DSR protocol discovers routes by flooding route request packets throughout the network. If the route request packet is received by a non-destination node, the node adds its identity to the route request packet and rebroadcasts it further. The Time-To-Live (TTL) value is included inside the route request packet to limit the flooding area. If the route request packet is received by the destination node, the node constructs the route reply packet and sends it back to the source node following the route that has just been discovered.

In [18], variations of the DSR protocol that exploit the benefits of directional antennas were proposed. The main idea being proposed is to reduce the search space by restricting the flooding of the route request packets to the area where the destination node is likely to be found. Two different approaches are proposed to restrict the flooding area. In the first approach, the source node transmits the route request packet only in the direction that it had been using earlier to send packets to the destination node. Each non-destination node that receives

the route request packet rebroadcasts it further using the same direction. In the second approach, the source node takes into account the directions that have been used earlier by the nodes in the last valid route to the destination node to roughly estimate the direction where the destination node is likely to be found. In either approach, a network-wide flooding is performed if the restricted flooding failed to give a result after a certain timeout period. Note that the proposed restricted flooding is only good if a route to the destination has been discovered earlier, not too long ago. For example, when the route to the destination is broken due to the node mobility and therefore a new route needs to be discovered. The protocol works under the assumption that the destination node is likely to be found close to the place it was found earlier.

The work presented in [19] attempts to achieve routing improvement by exploiting the capability of directional antennas to transmit over longer distance. In particular, the work focused on bridging permanent network partitions and repairing route breakages caused by the movement of intermediate nodes. During the route discovery, route request packets are flooded throughout the networks as in the normal DSR. However, if the source node does not receive the route reply packet after a certain timeout period, it reinitiates the route discovery, this time with the trigger partition bridging flag set in the route request packet. This will trigger the intermediate nodes to send the route request packets using directional transmissions over a longer distance and hence bridging network partitions. During the data transfer, when an intermediate node detects that the next-hop node becomes unreachable, possibly due to the node movement, it attempts to bridge over the node to reach the following next-hop node using the long range directional transmissions, thus avoiding route disruptions.

Another work that attempts to take the benefit of directional antennas is presented in [20]. This work focused on exploiting the directionality of the transmissions to minimize co-channel interference from neighboring nodes. The proposed protocol selects the best route by considering the number of overlaps between

beams in the route. Nodes in the route are able to calculate the beam overlaps by including positional information of the nodes in the route request packets. In addition to the number of beam overlaps, the proposed scheme also takes into account the total power loss when transmitting a packet from the source to the destination via the route.

A directional routing protocol proposed in [21] attempts to enhance the performance by enabling close cross-layer interactions between the MAC layer and the network layer (routing). The network layer is aware of the different antenna beams at the MAC layer. The information in the Directional Neighbor Table is shared between the MAC layer and the network layer. During the route discovery, the protocol employs the directional broadcast optimizations proposed in [22] to reduce packet redundancy and route discovery latency. When broadcasting the route request packet, the node transmits the packet only using the beams other than those from which the packet arrived. The node initiates the broadcast process using the beam that is opposite to the beam from which the packet was received, followed by the beams adjacent to the first beam. This process continues until the packet has been transmitted using all selected beams.

In [23], a study on the impact of directional antennas on the performance of the reactive routing protocol is presented. In the study, the DSR protocol was used over a directional MAC protocol. The tradeoff between the higher transmission range of directional antennas and the sweeping delay was evaluated. The higher transmission range of directional antennas leads to the discovery of the routes with lower number of hops. However, the route request packet is broadcast by having the antenna system sweep its transmitting beam sequentially over multiple directions. This introduces a sweeping delay during broadcasting. Neighbor nodes at different directions receive the broadcast packet at a slightly different time. Other factors that also impact the performance when directional antennas are used include the spatial reuse, deafness and directional interference. The study proposed some optimizations to DSR over directional MAC. The first optimization

is to delay the sending of the route reply packet to allow the destination node to choose the best route among all the routes that arrive. The next optimization is to forward the route request packet using fewer number of antenna elements that are diagonally opposite to the element from which the packet was received.

A few routing protocols that make use of the local neighbor information have been proposed. In [24], a routing protocol based on greedy forwarding is proposed. Each packet being routed to the destination includes the location of the destination node itself. Each intermediate node receiving such packet searches its local neighbor information for the node closest to the destination node and forward the received packet to the node. As this process continues, the packet gets successively closer to the destination until it finally reaches the destination node. The protocol is enhanced with perimeter forwarding which will be used in the situation where greedy forwarding is not possible, that is when the intermediate node receiving the packet has no neighbor that is closer to the destination node. One assumption made by this routing protocol is that the source node is able to determine the location of the destination node. Thus, it is assumed that a location registration and lookup service that maps node address to location is available.

A similar geographical routing protocol that uses the information about a subset of the nodes in the network is proposed in [25]. Each node in the network maintains a routing table that maps a geographical location to one of its neighbors. When an intermediate node receives a packet for a destination node, it finds an entry in the routing table which has the closest geographical location to the destination node. Then, the intermediate node forwards the received packet to the neighbor node corresponding to the entry. If the intermediate node cannot find an entry with the geographical location closer to the destination node than itself, the intermediate node will trigger the route discovery mechanism and update the routing table accordingly. This routing protocol also makes the assumption that the source node is able to determine the location of the destination node, and

thus the existence of a geographical location service is required.

In [26], a mechanism to improve the performance of the route request flooding during the route discovery is proposed. Upon receiving the route request packet, intermediate nodes identifies itself as a good or bad candidate for the next-hop node based on attributes such as the relative distance from the previous-hop node, estimated link lifetime, transmission power consumption, or residual battery capacity. The algorithm runs in a fully distributed manner as the decision is based only on the local information. The intermediate nodes will then adjust the rebroadcast delay accordingly. A good candidate will use a smaller rebroadcast delay while a bad candidate will use a bigger rebroadcast delay. By allowing good candidates to go first, it is expected that a better route to the destination will be discovered.

2.4 Broadcasting in Ad Hoc Networks with Directional Antennas

Broadcasting/flooding is an integral part of the DSR protocol. During the route discovery process, DSR floods the route request packet throughout the network. It is therefore necessary to review some of the previous works in the broadcasting/flooding area.

There are a number of schemes proposed for enhancing the performance of broadcasting/flooding in ad hoc network with omnidirectional antennas. But when it comes to broadcasting/flooding with directional antennas, the number of existing schemes is quite limited. Of this limited number of existing schemes, some are based on probability, some others are based on areas/zones, and only a few are based on knowledge of neighbor locations.

One of the earliest works in broadcasting using directional antennas can be found in [27]. Three directional broadcasting schemes are proposed. In the first scheme, a node receiving the broadcast packet will mark the incoming direction

of the packet as passive and wait for some random delay. If the same packet is received from other directions before the delay expires, those directions will be marked as passive as well. Upon the expiry of the delay, the node will forward the packet in the directions that are not marked as passive. The second scheme is similar to the first scheme, but the rebroadcast delay varies for different directions. The delay is inversely proportional to the extra coverage that can be made if the packet is forwarded along the direction. The larger the extra coverage is, the smaller the delay will be. In the third scheme, a node will designate the farthest neighbor node in each direction as the relay node, which will rebroadcast the packet further.

Some enhancements to the broadcasting schemes above are proposed in [22]. It is generally assumed that directional antennas transmit into one direction at a time, especially when the longer transmission range capability of the directional antenna is used. Under this assumption, broadcasting is achieved by sequentially transmitting into each direction, which introduces a sweeping delay. The enhancement proposed in [22] aims to reduce the overall latency by transmitting first in the direction with maximum uncovered nodes, i.e. the direction that is vertically opposite to the incoming direction of the packet. This will be followed by transmitting in the directions adjacent to this first direction. The process continues until all directions are covered.

An enhancement to the relay node based scheme is also proposed in [22]. A node receiving the broadcast packet will rebroadcast the packet with some probability. The probability is calculated from the ratio of the power received by the node to the power received by the farthest neighbor of the broadcast source. Therefore, nodes which are close to the broadcast source have lower probability to rebroadcast, while nodes which are far from the broadcast source have higher probability to rebroadcast.

A significant directional broadcast protocol called directional self-pruning (DSP) is proposed in [28]. The protocol extends the idea from the self-pruning broadcast

protocol for omnidirectional antennas. DSP requires the knowledge of neighbor information. In DSP, each node v that receives the broadcast packet will compute the set of covered neighbors. According to vs local view, a neighbor node w is covered if:

1. w is a known forward node
2. w is a neighbor of a known forward node u , and w is within one of us forward directions, or
3. w is a neighbor of a covered node with a higher id than v .

If all of vs neighbors are covered, v becomes a non-forward node. Otherwise, v becomes a forward node and rebroadcasts the packet to the directions of the uncovered neighbors.

Another significant directional broadcast protocol similar to DSP called the Link-Reduction-based protocol (LR) is proposed in [29]. LR also requires the knowledge of neighbor information. In LR, each node v that receives the broadcast packet will first compute a localized broadcasting tree by applying Prim's Algorithm. According to vs local view, the link to a neighbor node w will be reduced if:

1. the link is not found in the localized broadcasting tree
2. w is a neighbor of a known forward node u , and w is within one of us forward directions, or
3. w is the parent of v (the broadcast packet comes from w)

If all links to vs neighbors are reduced, v becomes a non-forward node. Otherwise, v becomes a forward node and rebroadcast the packet to the directions of the unreduced links.

Chapter 3

Distributed Neighbor Discovery in Ad Hoc Networks with Directional Antennas

3.1 Introduction

In this chapter, an efficient distributed neighbor discovery scheme for ad hoc networks with directional antennas is proposed. The proposed scheme aims to discover neighbors beyond the omnidirectional transmission range without high-power broadcast. The discovery process is expected to be fast and involves low number of neighbor nodes.

The rest of the chapter is organized as follows. Section 3.2 introduces a few definitions that will be used in the subsequent sections. A brief discussion about the active and the passive neighbor discovery schemes is presented in Section 3.3. The proposed neighbor discovery scheme is discussed in detail in Section 3.4. Section 3.5 presents an analysis of the coverage of the proposed scheme. The analysis is compared with the actual coverage obtained from simulations in Section 3.6. A comparative performance evaluation of the proposed scheme against other neighbor discovery algorithms is presented in Section 3.7.

3.2 Definitions

This section introduces a few definitions that will be used in the discussions in the subsequent sections.

Figure 3.1: Range and area definitions

Omnidirectional transmission range of node x The furthest distance from node x that is reachable by the omnidirectional transmission of node x . The omnidirectional transmission range of node x will be denoted as r_x^o .

Omnidirectional area of node x The area around node x that is reachable by the omnidirectional transmission of node x . This area is circular with node x at its center point (eq. 3.1). The omnidirectional area of node x will be denoted as A_x^o .

$$A_x^o = \{p | dist_x(p) \leq r_x^o\} \quad (3.1)$$

where $dist_x(p)$ denotes the distance of point p from node x .

Omnidirectional neighbors of node x Nodes that are located inside the omnidirectional area of node x (eq. 3.2). The omnidirectional neighbors of node x will be denoted as N_x^o .

$$N_x^o = \{n | loc_n \in A_x^o\} \quad (3.2)$$

where loc_n denotes the location of node n .

Directional transmission range of node x The furthest distance from node x that is reachable by the directional transmission of node x . The directional transmission range of node x will be denoted as r_x^d .

Directional area of node x The area around node x that is reachable by the directional transmission of node x , but is not reachable by the omnidirectional transmission of node x . This area is donut-shaped with node x at its center point. The directional area of node x will be denoted as A_x^d .

$$A_x^d = \{p | r_x^o < \text{dist}_x(p) \leq r_x^d\} \quad (3.3)$$

where $\text{dist}_x(p)$ denotes the distance of point p from node x .

Directional neighbors of node x Nodes that are located inside the directional area of node x (eq. 3.4). The directional neighbors of node x will be denoted as N_x^d .

$$N_x^d = \{n | \text{loc}_n \in A_x^d\} \quad (3.4)$$

where loc_n denotes the location of node n .

Fig. 3.1 shows the relationship between r_x^o , A_x^o , r_x^d , and A_x^d .

Based on the definitions above, it is clear that the omnidirectional area of node x does not overlap with the directional area of node x (eq. 3.5). Consequently, the omnidirectional neighbors of node x and the directional neighbors of node x are mutually exclusive (eq. 3.6). A neighbor of node x is either an omnidirectional neighbor or a directional neighbor, but not both.

$$A_x^o \cap A_x^d = \emptyset \quad (3.5)$$

$$N_x^o \cap N_x^d = \emptyset \quad (3.6)$$

Neighbor area of node x The area around node x that is reachable by either the omnidirectional or the directional transmission of node x (eq. 3.7 and

eq. 3.8). This area is circular with node x at its center point. The neighbor area of node x will be denoted as A_x .

$$A_x = A_x^o \cup A_x^d \quad (3.7)$$

$$A_x = \{p | dist_x(p) \leq r_x^d\} \quad (3.8)$$

Neighbors of node x All omnidirectional and directional neighbors of node x (eq. 3.9 and eq. 3.10). The neighbors of node x will be denoted as N_x .

$$N_x = \{n | loc_n \in A_x\} \quad (3.9)$$

$$N_x = N_x^o \cup N_x^d \quad (3.10)$$

3.3 Neighbor Discovery Schemes

In general, neighbor discovery schemes can be categorized into two groups: active neighbor discovery schemes and passive neighbor discovery schemes. In the active neighbor discovery schemes, a node initiates the discovery process by explicitly sending a request message to some other nodes. Upon receiving such request message, these nodes will send neighbor information to the requesting node. In the passive neighbor discovery schemes, a node intending to perform the discovery process will not explicitly send a request message. Instead, it will listen to the transmissions from other nodes to obtain information about its neighbors.

3.3.1 Active Neighbor Discovery Schemes

Simple Request-Reply Scheme

A simple active neighbor discovery scheme is described as follows. A node x intending to perform the neighbor discovery broadcasts a request message using omnidirectional transmission. Upon receiving the request message, each omnidirectional neighbor of x sends a reply message containing the information about

itself (node ID, location information, etc.) to x . In this scheme, only omnidirectional neighbors can be discovered by x since the request message only reaches omnidirectional neighbors of x . The number of reply transmissions is equal to the number of omnidirectional neighbors of x . This scheme will be referred in subsequent sections as the *simple request-reply* scheme.

High-Power Request-Reply Scheme

Extending the idea of the simple request-reply scheme, directional neighbors can also be discovered by utilizing high-power omnidirectional transmission to broadcast the request message. Upon receiving the request message, the directional neighbors of x send reply messages back to x , either using directional transmissions or using high-power omnidirectional transmissions. In this scheme, the number of transmissions is equal to the total number of neighbors of x , both directional neighbors and omnidirectional neighbors. Since high power is used for the request message, it may cause interference to larger area and may reduce the performance of the overall system. Moreover, in the case where power is limited, it is usually not desirable to use high-power transmissions. This scheme will be referred in subsequent sections as the *high-power request-reply* scheme.

3.3.2 Passive Neighbor Discovery Schemes

Simple HELLO Scheme

Passive neighbor discovery schemes are commonly realized by the use of periodic HELLO messages. Each node in the system periodically broadcasts HELLO messages using omnidirectional transmission. Each HELLO message contains information about the node sending the HELLO message. A node intending to perform the neighbor discovery listens to the HELLO messages sent by its omnidirectional neighbors. A node will discover its omnidirectional neighbors completely only after it has received HELLO messages from all its omnidirectional neighbors. The requesting node may need to wait as long as the interval between HELLO message

broadcasts after it starts the discovery process. Similar to the simple request-reply scheme, the requesting node can only discover omnidirectional neighbors due to the use of omnidirectional transmissions for broadcasting the HELLO messages. This scheme will be referred in subsequent sections as the *simple HELLO* scheme.

High-Power HELLO Scheme

The simple HELLO scheme can also be extended by utilizing high-power transmissions to broadcast the HELLO messages. Despite the ability to discover directional neighbors, this scheme penalizes the system performance due to the larger interference region. This scheme will be referred in subsequent sections as the *high-power HELLO* scheme.

Enhanced HELLO Scheme

The simple HELLO scheme can be enhanced by including the information about the neighbors discovered so far inside the HELLO message. This way, a node performing the neighbor discovery process will be able to gather information about both its omnidirectional and directional neighbors without penalizing the system with a larger interference region. This scheme will be referred in subsequent sections as the *enhanced HELLO* scheme.

3.4 The Proposed Scheme

3.4.1 Overview

In this section, an efficient neighbor discovery scheme is proposed. The proposed neighbor discovery scheme attempts to avail itself with the advantages of the basic neighbor discovery schemes as discussed in the previous section. It belongs to the category of active neighbor discovery schemes as the node intending to perform the neighbor discovery will explicitly send a request message to initiate the discovery process.

The proposed neighbor discovery scheme aims to achieve the following goals:

- **Discovery of both directional and omnidirectional neighbors**

As the main purpose of this neighbor discovery scheme is to provide support for communications using directional antennas, it is required that the scheme is able to discover not only omnidirectional neighbors but also directional neighbors.

- **No need for omnidirectional high-power broadcast**

Omnidirectional high power broadcast causes more interference and collisions, and is not desirable in a power-limited systems.

- **Fast discovery**

The neighbor discovery process should be as fast as possible

- **Minimise number of neighbor nodes involved in the discovery process**

The neighbor discovery process should involve only a subset of the neighbors to reduce the load at each node. It also implies fewer number of transmissions, and hence fewer number of collisions.

- **Distributed algorithm**

The neighbor discovery should work in a distributed manner.

The proposed neighbor discovery scheme is inspired by the principle of induction. Consider a system of static nodes at stable state. In this discussion, the stable state of a system is defined as no new nodes joining the system and no existing nodes leaving the system. In addition, at stable state, each node in the system knows its own neighbors, both omnidirectional and directional.

When a new node joins the system, it initiates the neighbor discovery process using the proposed algorithm. Some neighbor nodes will help the new node to gather information about its omnidirectional and directional neighbors. At the end of the neighbor discovery process, it is expected that the new node obtains

complete information about its omnidirectional and directional neighbors. Hence, at the end of the discovery process, the system will once again be in the stable state: no new nodes joining the system, no existing nodes leaving the system, and each node in the system knows its own neighbors, both omnidirectional and directional.

3.4.2 The Algorithm

Let S be a system of static nodes at stable state. Let x be a new node that is joining S and initiating the neighbor discovery process to quickly find its neighbors, both omnidirectional and directional (N_x). x initiates the neighbor discovery process by broadcasting information about itself using omnidirectional transmission. The information broadcast includes the node identifier and the location of node x . The broadcast message is received by the omnidirectional neighbors of node x (N_x^o). Upon receiving the broadcast message, the omnidirectional neighbors of node x update their discovered neighbor lists with information about x . At this stage, omnidirectional neighbors of node x learn about the existence of node x in S .

Assume all nodes use the same omnidirectional range r^o , as well as the same directional range r^d , and the directional range is at least twice the omnidirectional range. With this assumption, together with the assumption made previously that S is at stable state, two things can be noted:

- omnidirectional neighbors of x know each other (eq. 3.11)

$$\forall i \in N_x^o [N_x^o \subset N_i] \quad (3.11)$$

Proof:

Let $j \in N_x^o, j \neq i$. By definition, $dist_x(i) \leq r^o$ and $dist_x(j) \leq r^o$. The distance between i and j , $dist_i(j)$, is therefore bounded by $2r^o$. As r^d is assumed to be at least $2r^o$, $dist_i(j)$ is also bounded by r^d . By eq. 3.8 and

eq. 3.9, it can be concluded that $j \in N_i$, and thus $N_x^o \subset N_i$.

- omnidirectional neighbors of x may know some directional neighbors of x .

Specifically, $i \in N_x^o$ knows the nodes that belong to the set $N_x^d \cap N_i$.

From the first point above, it can be deduced that any omnidirectional neighbor of x is sufficient to help x gather information about all its omnidirectional neighbors (N_x^o). However, from the second point above, it can be deduced that a single omnidirectional neighbor of x is usually not sufficient to help x gather information about all its directional neighbors (N_x^d). An exception to this can be found if i , the omnidirectional neighbor of x , is located extremely close to x such that A_i almost fully overlaps with A_x . In this particular case, i alone is sufficient to help x gather information about all its omnidirectional and directional neighbors. In normal situations, however, nodes are not usually located extremely close to another node. Thus, more than one omnidirectional neighbors of x are needed to help x gather information about all its directional neighbors.

The subset of omnidirectional neighbors of x (N_x^o) that helps x gather information about all its neighbors (N_x) are called the **representative nodes for x** , denoted as R_x . R_x should be chosen such that

- the aggregated neighbor areas of the nodes in R_x cover the whole neighbor area of x (eq. 3.12).

$$A_x \subset \bigcup_{i \in R_x} A_i \quad (3.12)$$

- the number of nodes in R_x is minimized. With minimal number of nodes involved in helping x to gather information about its neighbors, the number of transmissions and the interference caused are expected to be reduced.

With the two goals above in mind, the question that arises next is how to choose the nodes to be included in R_x among the omnidirectional neighbors of x . It is desired that the selection process should not impose too much overhead, particularly in terms of the number of transmissions. Even better if the selection

Figure 3.2: Minimum coverage

process can be carried out independently at each neighbor node without the need to communicate with each other. This can be made possible when omnidirectional neighbors of x knows each other as discussed earlier (eq. 3.11). At the end of the selection process, it is expected that all omnidirectional neighbors of x agree on the same set R_x .

Different algorithms can be used for the selection process in order to achieve the most optimal R_x . A simple selection algorithm based on heuristic is proposed here. The simple selection algorithm starts choosing the first representative node, i_1 , by picking the node in N_x^o that is closest to x . This way, N_x will overlap substantially with N_{i_1} . If there are more than one nodes with the same distance from x , a tie-breaker rule is used, for example by scanning in clockwise direction starting from the north and picking the first node encountered. After choosing the first representative node, i_1 , at least 68.4% of A_x will have been covered. This minimum coverage of 68.4% occurs when i_1 lies at the border of A_x^o , i.e. when $dist_x(i_1) = r^o$ (see Fig. 3.2). The minimum coverage percentage can be found easily using simple geometry calculation, and hence the proof is omitted here.

To cover the remaining area, $A_x - A_{i_1}$, the second representative node, i_2 , is chosen. The simple selection algorithm chooses the node in N_x^o that is second closest to x as the second representative node. A smarter selection algorithm will

choose the node in N_x^o that can cover the largest portion of $A_x - A_{i_1}$ as the second representative node.

The selection process continues with choosing i_3, i_4 , and so on, using the same way as above until the whole neighbor area of x (A_x) is covered, or until there are no more omnidirectional neighbors to choose from. Although each omnidirectional neighbors of x computes R_x independently, they will agree on the same R_x because each node runs the same selection algorithm to determine R_x .

After R_x has been computed, nodes that do not belong to R_x ceases to be further involved in the neighbor discovery process. On the other hand, nodes that belong to R_x will need to send neighbor information to x . Each node in R_x is responsible for sending a unique subset of the neighbors of x (N_x). The unique subset that a node in R_x is responsible for depends on the order the node was chosen to be included in R_x during the selection process. The first representative node, i_1 , is responsible for sending the information about the nodes that belong to the set $N_x \cap N_{i_1}$. Note that $N_x \cap N_{i_1}$ includes all omnidirectional neighbors of x (N_x^o). i_1 is the only node in R_x that sends information about omnidirectional neighbors of x .

The second representative node, i_2 , is responsible for sending the information about the nodes that belong to the set $N_x \cap N_{i_2}$, minus the nodes that has been covered by the first representative node, i_1 . Similarly, the third representative node, i_3 is responsible for sending the information about the nodes that belong to the set $N_x \cap N_{i_3}$, minus the nodes that has been covered by the first and the second representative nodes, i_1 and i_2 . In general, the k^{th} representative node is responsible for sending the information about the nodes that belong to the set $N_x \cap N_{i_k}$, minus the nodes that has been covered by the previous representative nodes (eq. 3.13).

$$\left(N_x \cap N_{i_k}\right) - \bigcup_{j=1}^{k-1} N_{i_j} \quad (3.13)$$

Note that the neighbor information sent by different representative nodes do not

overlap with each other. There is no redundant neighbor information being sent, and hence less traffic is generated.

The pseudocode of the steps performed by the joining node x is presented in Algorithm 3.1, while the pseudocode of the steps performed by each omnidirectional neighbor of x is presented in Algorithm 3.2.

Algorithm 3.1 The joining node x

- 1: Let N_x be the neighbor list of node x
 $N_x \leftarrow \emptyset$
 - 2: broadcast request message using omnidirectional transmission
 - 3: wait for response messages from representative nodes
 - 4: **for all** response messages received from representative nodes **do**
 - 5: **for all** neighbor information Z contained in the received response messages **do**
 - 6: $N_x \leftarrow N_x \cup Z$
 - 7: **end for**
 - 8: **end for**
-

3.4.3 Example

To illustrate the proposed algorithm, an example will be discussed in this section.

Consider the scenario in Fig. 3.3. There are currently 7 nodes in the system, nodes 1 to 7. A new node x joins the system and initiates the neighbor discovery process by broadcasting information about itself using omnidirectional transmission. Nodes 1, 2, 3, and 4 in the omnidirectional area of x receive the broadcast message and update their discovered neighbor lists with information about x . At this stage, nodes 1, 2, 3, and 4 learn about the existence of x in the system.

After receiving the broadcast message from x , nodes 1, 2, 3, and 4 compute R_x independently, and determine whether they are one of the representative nodes in R_x that are responsible for sending neighbor information to x . Node 1 finds that it is the closest node to x among other omnidirectional neighbors of x . Hence, it becomes the first representative node for x . Node 1 sends the information about all omnidirectional neighbors of x (nodes 1, 2, 3, and 4), plus the information about a directional neighbors of x that it knows (node 5).

Algorithm 3.2 Each omnidirectional neighbor of x

```

1: receive broadcast request message from  $x$ 
2: Let  $N_i$  be the neighbor list of node  $i$ 
    $N_i \leftarrow N_i \cup \{x\}$ 
3:  $W \leftarrow (N_i \cup \{i\}) - \{x\}$ 
4: for all  $j \in W$  do
5:   if  $\text{dist}_x(j) > r^d$  then
6:      $W \leftarrow W - \{j\}$ 
7:   end if
8: end for
9:  $Z \leftarrow W$ 
10: sort all  $j \in W$  in ascending order based on  $\text{dist}_x(j)$ , then based on  $\text{angle}_x(j)$ 
11: for all  $j \in W$  do
12:   if  $j = i$  then
13:     exit for
14:   end if
15:    $Z \leftarrow Z - \{j\}$ 
16:   for all  $k \in W, k \neq j$  do
17:     if  $\text{dist}_j(k) \leq r^d$  then
18:        $Z \leftarrow Z - \{k\}$ 
19:     end if
20:   end for
21: end for
22: if  $Z \neq \emptyset$  then
23:   send  $Z$  to  $x$ 
24: else
25:   do nothing
26: end if

```

Figure 3.3: Example neighbor discovery scenario

Node 2 finds that it is the second closest node to x . Node 2 has the information about all omnidirectional neighbors of x (nodes 1, 2, 3, and 4), plus the information about some directional neighbors of x (nodes 5 and 6). However, node 2 computes that the information about nodes 1, 2, 3, 4, and 5 has been covered by node 1. Thus, node 2 only sends the information about the node that has not been covered (node 6).

Similarly, node 3 finds that it is the third closest node to x . Node 3 has the information about nodes 1, 2, 3, 4, 5, and 6, all of which have been covered previously by nodes 1 and 2. Thus, node 3 will not send any neighbor information to node x .

Finally, the last closest node to x is node 4. Among all nodes that are known to node 4, only node 7 has not been covered by previous representative nodes. Thus, node 4 sends the information about node 7 to x .

At this point, there are no more omnidirectional neighbors of x that may send additional neighbor information to x . The neighbor discovery process ends here.

3.5 Coverage Analysis

In this section, an analysis of the coverage of the proposed algorithm will be presented. The coverage analysis finds the percentage of the neighbors of the joining node x that can be discovered by performing the proposed algorithm.

When a new node x joins the system and performs the proposed algorithm, it may be able to obtain a complete or partial information about its neighbors, depending on the number and placement of its omnidirectional neighbors. Having more omnidirectional neighbors increases the chance of discovering all neighbors. The placement of the omnidirectional nodes is important as well. Having lots of omnidirectional neighbors that are clustered at one spot does not help the discovery process much. It is as good as having only one omnidirectional neighbor located at the spot.

The first step of the analysis is to find how many percents of the neighbor area of x are on average covered when there are n omnidirectional neighbors of x . Calculating this percentage is not easy as it requires finding the area of intersecting circles. Finding the area of intersecting circles purely by geometry calculation is tedious and difficult. Therefore, the analysis does not use geometry calculation to find the area of intersecting circles. Instead, the analysis uses a simple monte carlo method to approximate the expected coverage.

A large number of sample points (≈ 10000 points) are scattered randomly inside the neighbor area of x . n omnidirectional neighbors of x ($i_1, i_2, i_3, \dots, i_n$) are placed randomly. Then, each sample point is evaluated on whether it falls inside the neighbor area of any of $i_1, i_2, i_3, \dots, i_n$. If it does, the sample point is marked as covered. Otherwise, it is marked as not covered. After all sample points have been evaluated, the percentage of the covered sample points is calculated. The procedure is repeated a large number of times (> 1000 times) with different placements of sample points and neighbor nodes. The expected coverage when the number of omnidirectional neighbors of x is n , denoted as $c(n)$, can be

Figure 3.4: Expected coverage $c(n)$ against the number of omnidirectional neighbors

found by averaging over all runs. The expected coverage, $c(n)$, for n from 0 to 20 is shown in Fig. 3.4. The directional transmission ranges, r^d , assumed in the calculation are twice and 4 times the omnidirectional transmission range ($2r^o$ and $4r^o$). This expected coverage, $c(n)$, is used to obtain the expected coverage of the proposed algorithm.

Assume nodes are uniformly distributed in a large area. Let ρ denote the node density, i.e. the average number of nodes per unit area. The expected number of omnidirectional neighbors of x , denoted as λ , can be found as shown in eq. 3.14.

$$\lambda = \rho\pi(r^o)^2 \quad (3.14)$$

The probability of finding n omnidirectional neighbors follows the Poisson distribution with mean λ as shown in eq. 3.15.

$$p(n) = \frac{\lambda^n e^{-\lambda}}{n!} \quad (3.15)$$

Figure 3.5: Expected area not covered by the proposed algorithm against node density

Thus, the expected coverage of the proposed algorithm, $E[c(n)]$, can be calculated as shown in eq. 3.16.

$$E[c(n)] = \sum_{k=0}^{\infty} p(k)c(k) \quad (3.16)$$

Eq. 3.16 is computed numerically for node density ρ from 10 to 100 nodes/km². The omnidirectional transmission range r^o used is 200 m. The directional transmission ranges used are $2r^o = 400$ m and $4r^o = 800$ m. The expected area not covered by the proposed algorithm, $1 - E[c(n)]$, is computed and shown in Fig. 3.5.

It can be seen in Fig. 3.5 that in a sparse population, a considerable percentage of the directional transmission range is not covered by the proposed algorithm. However, as the node density increases, the expected not-covered area decreases rapidly. When the node density is greater than 50 nodes/km², the percentage of the not-covered area drops to less than 1%, and hence becomes insignificant. As a conclusion, the proposed algorithm works better in a dense population.

3.6 Coverage Evaluation

Simulations were carried out to find the actual coverage of the proposed algorithm and to compare the result with the expected coverage as discussed in section 3.5. The coverage simulations were performed using a custom-tailored program as well as using the QualNet Network Simulator 3.8 [30].

n nodes were randomly scattered inside a $1500 \text{ m} \times 1500 \text{ m}$ square area according to the uniform distribution, such that the node density equaled to ρ . These nodes were static and constituted the system S in the stable state. The omnidirectional transmission range used was 200 m , while the directional transmission range used was 400 m .

A newly joining node x was placed at a random position inside the square area. The proposed neighbor discovery algorithm was performed for node x . At the end of the neighbor discovery process, the number of the undiscovered neighbors of x was compared with the actual number of neighbors of x .

The simulations were performed with different values of ρ , from 4 nodes/km^2 to 44 nodes/km^2 . For each ρ , the simulation was repeated 1000 times. By averaging over all simulation runs for each ρ , the average area not-covered by the proposed algorithm can be obtained. The simulation results were then compared with the analytical result as shown in Fig. 3.6. It can be seen that the simulation results tracks closely the analytical result.

In addition, the number of representative nodes for x was also recorded in each simulation run. By averaging over all simulation runs for each ρ , the average number of representative nodes can be obtained. The result was compared with the average number of omnidirectional neighbors of x for each ρ as shown in Fig. 3.7. It can be seen that the average number of representative nodes is lower than the average number of omnidirectional neighbors. The difference between the two increases as the node density increases. This shows the efficiency of the proposed algorithm since only a small subset of the omnidirectional neighbors are involved

Figure 3.6: Actual not-covered area obtained from simulations vs theoretical not-covered area

in the discovery process.

3.7 Comparative Performance Evaluation

To further evaluate the performance of the proposed algorithm, a comparative performance evaluation was performed. The performance of the proposed algorithm was compared with the performance of other basic neighbor discovery algorithms. Both active and passive algorithms were used in the comparison. As the proposed algorithm utilizes omnidirectional broadcast, the comparison was focused on comparing it with other omnidirectional-broadcast based algorithms.

The time taken to perform the discovery was used as the performance metric. As in the coverage evaluation, the comparative performance evaluation was carried out by using simulation. The QualNet Network Simulator 3.9 [30] was again used to perform the simulation. The neighbor discovery algorithms were implemented at the network layer, while the IEEE 802.11 was used at the MAC layer.

Simulations were carried out using two different scenarios. Each scenario cor-

Figure 3.7: Average number of representative nodes against node density

responds to a particular node placement. In each scenario, 135 nodes were placed randomly in an area of size $1500 \text{ m} \times 1500 \text{ m}$. This corresponds to a node density of about 60 nodes/km^2 . One of the 135 nodes in the area was randomly chosen to be the node performing neighbor discovery. The chosen node had its entire omnidirectional and directional areas inside the $1500 \text{ m} \times 1500 \text{ m}$ area (they were not clipped by the area boundaries).

In the first scenario, the chosen node had a total of 26 neighbors, consisting of 8 omnidirectional neighbors and 18 directional neighbors (Fig. 3.8). In the second scenario, the chosen node had a total of 36 neighbors, consisting of 6 omnidirectional neighbors and 30 directional neighbors (Fig. 3.9). The omnidirectional transmission range used was 200 m, while the directional transmission range used was 400 m. In simulating other algorithms that used HELLO broadcasts, the broadcast interval was set to 2 seconds.

For each scenario, we run 6 different neighbor discovery algorithms on the chosen node, namely the simple request-reply scheme, the high-power request-reply scheme, the simple HELLO scheme, the high-power HELLO scheme, the enhanced

Figure 3.8: Scenario 1 for comparative performance evaluation

Figure 3.9: Scenario 2 for comparative performance evaluation

HELLO scheme, and the proposed algorithm. The simulation was repeated 100 times for each neighbor discovery algorithm. The average result of the 100 runs was taken. In each run, the chosen node started the discovery process at time 0. The interval since the beginning of the discovery process until the time the last neighbor was discovered (measured in seconds) was recorded. In addition, the number of neighbors discovered at the end of each simulation run was also recorded. The results are presented in Table 3.1.

Table 3.1: Performance comparison of neighbor discovery algorithms

Scheme	Type	Scenario 1		Scenario 2	
		Number of neighbors discovered	Time last neighbor discovered	Number of neighbors discovered	Time last neighbor discovered
Simple Request-Reply	Active	8	0.013831	6	0.010224
High-power Request-Reply	Active	26	0.049102	35.98	0.076050
Simple HELLO	Passive	8	1.782262	6	1.762605
High-power HELLO	Passive	25.84	2.040947	35.88	2.024939
Enhanced HELLO	Passive	26	0.345972	36	1.063491
Proposed algorithm	Active	26	0.007595	36	0.008695
Actual number of neighbors		26		36	

From the table, it can be seen that the proposed algorithm was superior to other algorithms with respect to the time the last neighbor being discovered. The proposed algorithm was able to discover both omnidirectional and directional neighbors in the lowest amount of time.

The simple request-reply scheme was the next fastest algorithm. However the scheme was only able to discover omnidirectional neighbors. When the node performing the neighbor discovery broadcast the request message, it used omnidirectional transmission. As the consequence, only neighbors located inside its omnidirectional area received the request message and reacted to it by sending back the information about itself to the requesting node. Each omnidirectional neighbor made one reply transmission to the requesting node, and hence the number of reply transmissions was equal to the number of omnidirectional neighbors (8 reply transmissions in the first scenario and 6 reply transmissions in the second scenario).

The high-power request reply scheme took longer amount of time to discover the last neighbor than the simple request-reply scheme. As high-power transmission was used to broadcast the request message, neighbors located inside the directional area of the requesting node were able to receive the request message and reacted to it by sending back information about itself to the requesting node. The number of reply transmissions was equal to the number of omnidirectional and directional neighbors (26 reply transmissions in the first scenario and 36 reply transmissions in the second scenario). With the increase of the number of reply transmissions, the time taken to discover the last neighbor increased accordingly. Additionally, with larger number of reply transmissions, there was a higher chance for the reply transmissions to collide with each other and got retransmitted at the MAC layer, thus increasing the discovery time further. In the second scenario, the average number of discovered neighbors was slightly smaller than the actual number of neighbors. This was because in some simulation runs, the reply transmissions of some neighbor nodes were not received by the requesting node due to

collisions.

Passive neighbor discovery schemes generally take longer amount of time to discover the last neighbor as compared to active neighbor discovery schemes. In the worst case, passive neighbor discovery schemes may need to wait as long as the interval between HELLO broadcasts to discover the last neighbor. This worst case situation occurs when the node performing discovery starts the discovery process just after one of its neighbors broadcasts the HELLO message. In this case, the node performing discovery needs to wait for the neighbor to broadcast its next HELLO message. This will take an amount of time close to the interval between HELLO broadcasts (2 seconds in the simulation). In the existence of collisions among broadcasts from different neighbors, the time taken to discover the last neighbor may get longer than the interval between HELLO broadcasts because the node performing discovery will need to wait for another period to receive the next HELLO message. The best case for passive neighbor discovery scheme occurs when the neighbors broadcast the HELLO messages one after another, just after the node performing discovery starts the discovery process. However, this is a rare occasion.

The simple HELLO scheme was able to discover only omnidirectional neighbors due to the reason similar to that of the simple request-reply scheme. The use of omnidirectional transmission for broadcasting the HELLO message made the node performing discovery only able to hear HELLO messages from its omnidirectional neighbors.

The high-power HELLO scheme performance was the worst among the 6 neighbor discovery schemes. Not only the time taken to discover the last neighbor was the longest, but also the average number of neighbors discovered was lower than the actual number of neighbors. The main cause was the higher number of HELLO message collisions due to the larger interference region of the high-power broadcast. The node performing discovery may need to wait for more than one period of HELLO broadcast before it discovered the last neighbor. This situation was

found in both the first scenario and the second scenario. The average times to discover the last neighbor in both scenarios were greater than the interval between HELLO broadcasts.

The enhanced HELLO scheme improved the performance of the passive neighbor discovery scheme. By including information about discovered neighbors inside the HELLO message, the node performing discovery may be able to discover its neighbors faster as it did not need to wait until all neighbors broadcast their HELLO messages. In addition, this scheme made it possible to also discover directional neighbors without the need to use high-power transmission. Despite these improvements, the issue inherent in passive neighbor discovery schemes still applied. The node performing discovery may not receive the HELLO messages from its neighbors right after it started the discovery process. It may need to wait for some random amount of time before it received the first HELLO message from its neighbor.

The proposed algorithm, being an active neighbor discovery scheme, did not need to wait for some random amount of time to receive the replies from its neighbors. As soon as the requesting node broadcast the request message, its representative nodes started sending replies. As the number of representative nodes was less than the number of omnidirectional neighbors, the time taken by the proposed algorithm to discover the last neighbor was lower than the time taken by the simple request-reply scheme.

Chapter 4

DSR Route Request Flooding in Ad Hoc Networks with Directional Antennas

4.1 Introduction

In this chapter, we will focus on the issue of enhancing the DSR route request flooding mechanism in ad hoc networks with directional antennas. The proposed enhancement exploits the neighbor information provided by the neighbor discovery. The main idea of the proposed enhancement is to strategically select a small subset of neighbor nodes as representative nodes to forward the DSR route request. The DSR route request is forwarded to each representative node using directional transmission.

Existing directional broadcast protocols have one common main goal: full delivery of the broadcast packet to all nodes in the system. In other words, it is highly desired that all nodes in the system receive the broadcast packet at least once. This makes those protocols suitable for scenarios where dissemination of information to all nodes in the system is very important.

The main goal of route request flooding in the DSR protocol, however, is to

discover the route to the destination node, and not so much to make sure that every node in the system receives the route request packet. In fact, if there is a flooding scheme that does not require each and every node in the system to receive the route request packet but is still able to discover the route to the destination node with the same route discovery success rate, it is certainly desirable. The proposed route request flooding scheme exactly does this.

The proposed scheme does not require each and every node in the system to receive the route request packet. Only the representative nodes are expected to receive and process the route request packet. The rest of the nodes in the system do not need to know the route request packet. In fact, if the directional transmission beam width is narrow, some non-representative nodes may not hear the transmissions of the route request packet at all.

The expected benefits of the proposed enhancement include:

- lower number of nodes involved in the route discovery process as compared to the traditional DSR route request flooding mechanism. Only neighbor nodes selected as representative nodes are involved in the route discovery process.
- lower number of directional transmissions made by each representative node involved in the route discovery process. The proposed enhancement does not need to send out the route request packets to all directions when forwarding the route requests. It only forwards the route request in the directions of the selected representative nodes. This also implies that lower amount of the node available time is taken by the route discovery process.
- lower amount of interference and collisions during the route discovery process due to the above two points.
- shorter route length (number of hops) due to the use of directional transmissions as compared with the traditional DSR that uses omnidirectional

transmissions. Directional transmissions have a longer range than omnidirectional transmissions.

The rest of the chapter is organized as follows. The proposed algorithm is presented in Section 4.2, followed by the two proposed selection strategies: the Fixed Branching Strategy in Section 4.3 and the Honeycomb Strategy in Section 4.4. Section 4.5 presents the performance comparison of the proposed strategies with other strategies.

4.2 The Proposed Algorithm

In this section, the proposed algorithm for efficiently flooding the DSR route request in ad-hoc networks with directional antennas will be discussed in detail.

The proposed algorithm makes the following assumptions:

- each node in the system is equipped with a directional antenna
- the directional transmission range is longer than the omnidirectional transmission range.
- a neighbor discovery protocol is running at each node in the system. The neighbor discovery is able to provide the node with the information about its omnidirectional neighbors as well as its directional neighbors. The neighbor information includes both the identity and the location of the neighbors.
- each node knows its own location

In source routing, the route a packet takes to reach the destination node is specified by the source node and embedded in the packet itself. When a node has a packet to send, it checks whether it has already got the route to the destination node in its route cache. If the route is not in the route cache, the source node initiates the route discovery process.

In the proposed algorithm, the source node begins the route discovery process by checking the neighbor list provided by the neighbor discovery to find the destination node. If the destination node exists in the neighbor list, a direct route to the destination node is found and the route discovery process immediately terminates here.

Otherwise, the source node selects a small subset of the neighbor nodes as the representative nodes. There are different strategies that can be used to select the representative nodes. Two strategies will be discussed in detail in the subsequent sections. The selection of the representative nodes by the source node will subsequently be called the *initial representative nodes selection*.

The source node constructs the route request packet to be sent to the representative nodes. The route request packet is similar to the traditional DSR route request packet, with the addition of the location of the source node. The request packet is then transmitted to the representative nodes using directional transmissions. The pseudo-code of the algorithm used by the source node to initiate the route discovery process is presented in Algorithm 4.1.

Algorithm 4.1 Source node initiates the route discovery process

```

1: let  $L_{src}$  be the neighbor list of the source node  $n_{src}$ 
2: if  $n_{dest}$  in  $L_{src}$  then
3:   return  $n_{dest}$  is a direct neighbor of  $n_{src}$ 
4: end if
5: construct request packet  $P_{req}$ 
6: append  $n_{src}$  to  $P_{req}.route$ 
7:  $N_{rpr} \leftarrow$  select initial representative nodes
8: for all  $n$  in  $N_{rpr}$  do
9:   send  $P_{req}$  to  $n$  using directional transmission
10: end for

```

Upon receiving the route request packet, each representative node checks its own neighbor list to find the destination node. If the destination node exists in the neighbor list, a route from the source node to the destination node is found. In this case, the representative node constructs the route reply packet and transmits it back to the source node following the route that has just been

found. This is similar to the way the traditional DSR does to send the reply packet back to the source node, except that directional transmissions are used instead of omnidirectional transmissions. Note that the route reply packet is not constructed by the destination node as in the traditional DSR, but by the last node in the route just before the destination node.

In the case the destination node does not exist in the neighbor list, the representative nodes will have to forward the route request packet further. For this, each representative node selects a small subset of its neighbor nodes as the next representative nodes. Just like the initial representative nodes selection by the source node, there are different strategies that can be used to select the next representative nodes. Two strategies will be discussed in detail in the subsequent sections. The selection of the next representative nodes will subsequently be called the *next representative nodes selection*.

This process of forwarding the route request packet continues until the Time To Live (TTL) value in the route request packet reaches zero. The pseudo-code of the algorithm used by the intermediate node when it receives the route request packet is presented in Algorithm 4.2.

Algorithm 4.2 Intermediate node receives the route request packet

```

1: receive request packet  $P_{req}$  from previous hop node  $n_{prev}$ 
2: let  $L_{int}$  be the neighbor list of the intermediate node  $n_{int}$ 
3: append  $n_{int}$  to  $P_{req}.route$ 
4:  $P_{req}.TTL \leftarrow P_{req}.TTL - 1$ 
5: if  $n_{dest}$  in  $L_{int}$  then
6:   construct reply packet  $P_{rep}$ 
7:    $P_{rep}.route \leftarrow P_{req}.route$ 
8:   append  $n_{dest}$  to  $P_{rep}.route$ 
9:   send  $P_{rep}$  to  $n_{prev}$  using directional transmission
10: else if  $P_{req}.TTL > 0$  then
11:    $N_{rpr} \leftarrow$  select next representative nodes
12:   for all  $n$  in  $N_{rpr}$  do
13:     send  $P_{req}$  to  $n$  using directional transmission
14:   end for
15: else
16:   do nothing
17: end if

```

Figure 4.1: A scenario where the destination node is a direct neighbor of the source node

Figure 4.1 shows a scenario where the destination node is a direct neighbor of the source node. In the figure, Node A is the source node, while Node B is the destination node. The circle with Node A as its center point indicates the directional transmission range of Node A. All nodes inside the circle are the neighbors of Node A. As the first step of the route discovery, Node A checks its own neighbor list, which is provided by the neighbor discovery running at Node A, and finds that the destination node is one of its neighbors. In this trivial case, a direct route to the destination is found, and the route discovery process immediately ends here. Notice that in this case there is no route request packet being transmitted.

Figure 4.2 shows a scenario where the destination node is not a direct neighbor of the source node. In the figure, Node A is the source node initiating the route discovery process. Checking its neighbor list, Node A finds that the destination node is not one of its neighbors. Therefore, Node A performs the initial representative node selection. It happens that the selection strategy used in this scenario

Figure 4.2: Initial representative node selection

chooses Nodes B, C, and D among all neighbors of Node A as the representative nodes. Node A constructs the route request packet and sends it to Nodes B, C, and D using 3 directional transmissions (one directional transmission for each representative node).

Upon receiving the route request packet, each of the representative nodes checks its own neighbor list to see if the destination node is one of its neighbors. If the destination node happens to be the neighbor of any of these representative nodes, a route to the destination node is found. The representative node having the destination node as its neighbor will construct the route reply packet and send it back to the source node following the route that has just been found.

The four circles shown in figure 4.2 indicate the directional transmission ranges of the source node and the representative nodes. The nodes inside these circles are the neighbors of the source node or the representative nodes. The area inside these circles is considered as *covered* by the route discovery, meaning that if the destination node is located inside any of these circles, the route to the destination

will be found.

If the representative node finds that the destination node is not one of its neighbors, it has the responsibility to forward the route request further, unless the TTL value has reached 0. For this, the representative node chooses the next representative nodes among its neighbors using certain selection strategy and forwards the route request packet to them.

The proposed algorithm for the route discovery described above leaves open the strategies to use for the *initial representative nodes selection* and the *next representative nodes selection*. In fact, the success of the route discovery depends very much on the right strategies being used in those two decision points. In general, there are some desirable characteristics of a good selection strategy:

- *small set of representative nodes*: small number of neighbor nodes selected as representative nodes. The eventual goal is to minimize the number of transmission directions as well as the number of transmissions performed by a node when forwarding the route request packet.
- *total coverage*: each node in the system should be covered by (be reachable from, be a neighbor of) at least one representative node.
- *minimal overlap*: each node in the system should be covered by as few representative nodes as possible to avoid transmission redundancy.

Based on the total coverage and the minimal overlap characteristics, in the ideal case, each node in the system should be covered by one and only one representative node.

In the subsequent sections, two different strategies will be discussed: the simple and intuitive *fixed branching strategy*, and the more complex *honeycomb strategy*.

Figure 4.3: The source node chooses a representative node in one of the predefined directions. Node B is the farthest neighbor of Node A that falls in the sector corresponding to the predefined direction, and therefore is chosen as the representative node for the direction.

4.3 Fixed Branching Strategy

In the fixed branching strategy, the source node chooses one neighbor node in each of the predefined directions as the representative node. For each predefined direction, a sector with a fixed width is defined. The width of the sector is chosen to be the same as the beam width of the directional antenna. The neighbor node farthest from the source node that falls in the sector is chosen as the representative node for that particular direction (Figure 4.3).

Generally, the source node does not have any clue about where the destination node is likely to be found. It is therefore reasonable for the source node to divide the full circle evenly into a number of predefined directions such that the angles between adjacent directions are equal (Eq. 4.1).

$$\alpha_n = n \times \frac{360^\circ}{N} + C, \text{ for } 0 \leq n < N \quad (4.1)$$

where α_n is the n^{th} predefined direction, N is the number of directions and C is an arbitrary constant. For example, if 3 directions are to be used, it is reasonable to choose the directions 0° , 120° , and 240° . The angles between adjacent directions are equal to 120° . If 4 directions are to be used, the following directions can be chosen: 5° , 95° , 185° , and 275° . The angles between adjacent directions are equal

to 90°.

The initial representative nodes selection in the fixed branching strategy is presented in Algorithm 4.3.

Algorithm 4.3 Initial representative nodes selection in the fixed branching strategy

```

1: let  $D_{pre}$  be the list of predefined directions
2: let  $L_{src}$  be the neighbor list of the source node  $n_{src}$ 
3:  $N_{rpr} \leftarrow \emptyset$ 
4: for all  $\alpha$  in  $D_{pre}$  do
5:   let  $S_\alpha$  be the fixed-width sector in the direction  $\alpha$ 
6:    $r \leftarrow null$ 
7:   for all  $n$  in  $L_{src}$  do
8:     if  $n$  falls in  $S_\alpha$  then
9:       if  $distance(n_{src}, n) > distance(n_{src}, r)$  or  $r$  is null then
10:         $r \leftarrow n$ 
11:       end if
12:     end if
13:   end for
14:    $N_{rpr} \leftarrow N_{rpr} \cup \{r\}$ 
15: end for
16: return  $N_{rpr}$ 

```

The next representative nodes selection performed by the intermediate nodes is slightly more complex than the initial representative nodes selection performed by the source node. Upon receiving the route request packet, the first thing the intermediate node needs to do is to calculate the direction from the source node to itself. The direction is calculated based on the location of the source node and its own location. Remember that the location of the source node is included inside the route request packet.

The next step is to determine the directions that will be used by the intermediate node to forward the route request. The forwarding directions are calculated from a set of predefined relative directions, with respect to the direction from the source node calculated in the first step, as shown by Eq. 4.2.

$$\beta_n = D + \gamma_n, \text{ for } 0 \leq n < N \quad (4.2)$$

Figure 4.4: The intermediate node determines the forwarding directions. The forwarding directions are calculated with respect to the direction from the source node to the intermediate node, not to the incoming direction of the route request packet.

where β_n is the n^{th} forwarding direction, D is the direction from the source node to the intermediate node, γ_n is the n^{th} predefined relative direction, and N is the number of directions.

Generally, it is desired that the route request packet be forwarded away from the source node. Therefore, it is recommended to choose the predefined relative directions between -90° and 90° . The relative directions are usually chosen to be symmetrical about 0° , although this is not a requirement. For example, the following relative directions can be chosen: -30° , 0° , and 30° .

Figure 4.4 shows how the intermediate node determines the forwarding directions. In the figure, the intermediate nodes uses the relative directions -30° and 30° . Notice that the forwarding directions are calculated relative to the direction from the source node (Node A) to the intermediate node (Node B), not relative to the direction from which the route request packet was received by the intermediate node.

Similar to the initial representative nodes selection, a sector with a fixed width is defined for each forwarding direction. The width of the sector is chosen to be the same as the beam width of the directional antenna. The neighbor node farthest

from the intermediate node that falls in the sector is chosen as the representative node for that particular direction.

The next representative nodes selection in the fixed branching strategy is presented in Algorithm 4.4.

Algorithm 4.4 Next representative nodes selection in the fixed branching strategy

```

1: let  $D_{pre}$  be the list of predefined relative directions
2: let  $L_{int}$  be the neighbor list of the intermediate node  $n_{int}$ 
3:  $N_{rpr} \leftarrow \emptyset$ 
4:  $D \leftarrow direction(n_{src}, n_{int})$ 
5: for all  $\gamma$  in  $D_{pre}$  do
6:    $\beta \leftarrow D + \gamma$ 
7:   let  $S_\beta$  be the fixed-width sector in the direction  $\beta$ 
8:    $r \leftarrow null$ 
9:   for all  $n$  in  $L_{int}$  do
10:    if  $n$  falls in  $S_\beta$  then
11:      if  $distance(n_{int}, n) > distance(n_{int}, r)$  or  $r$  is null then
12:         $r \leftarrow n$ 
13:      end if
14:    end if
15:  end for
16:   $N_{rpr} \leftarrow N_{rpr} \cup \{r\}$ 
17: end for
18: return  $N_{rpr}$ 

```

Overall, this strategy will result in a flooding pattern that looks like the branching of a tree.

Figure 4.5 shows a scenario that uses the fixed branching strategy with $TTL = 3$. In the figure, Node A is the source node, while Node B is the destination node. The initial representative nodes selection chooses nodes in the following predefined directions: 60° , 180° , and 300° . The next representative node selection uses the following predefined relative directions: -60° , 0° , and 60° . In this scenario, the route from the source node to the destination node is successfully found. The green nodes in the figure indicate the representative nodes. The yellow circles indicate the directional transmission ranges of the source node and the representative nodes. The area inside these yellow circles are covered by the route discovery,

Figure 4.5: Fixed branching strategy with $TTL = 3$

meaning that a route from the source node to any nodes inside the yellow circles can be discovered.

Figure 4.6 shows a scenario that uses the fixed branching strategy with $TTL = 8$. The predefined directions for the initial representative nodes selection and the next representative nodes selection are the same as in the previous scenario. It can be seen from the figure that the whole area is covered by yellow circles, meaning that a route from the source node to any nodes in the system should be able to be discovered. Notice that as the route request packet is forwarded to the further nodes, the density of the nodes selected as the representative nodes (shown in the figure in green) increases, and so does the density of the transmission branches. In the area where the density of the representative nodes is high, yellow circles overlap substantially with each other. According to the minimal overlap characteristic of a good selection strategy, this is not desirable as it corresponds to many transmission redundancies.

Choosing the right number of directions for the initial representative nodes

Figure 4.6: Fixed branching strategy with $TTL = 8$

selection as well as the suitable relative directions for the next representative nodes selection is important to achieve total coverage and minimal overlap. Figure 4.7 and Figure 4.8 show two examples of poor coverage due to the bad choice of the parameters.

The scenario in Figure 4.7 uses 3 directions for the initial representative nodes selection. The next representative nodes selection uses the relative directions: -30° , 0° , and 30° . The figure shows that these relative directions are not spread wide enough, resulting in the route discovery failing to cover the entire area. The white areas around the directions 0° , 120° , and 240° are not covered by any representative nodes. Therefore, if the destination node falls in these white areas, the route to it will not be discovered.

The scenario in Figure 4.8 uses 4 directions instead of 3 for the initial representative nodes selection. Despite this increase in number of directions, the next representative nodes selection uses even narrower relative directions: -15° , 0° , and 15° . This results in areas that are not covered by the route discovery around

Figure 4.7: An example of poor coverage. The initial representative nodes selection uses 3 directions. The next representative nodes selection uses relative directions: -30° , 0° , and 30° .

Figure 4.8: An example of poor coverage. The initial representative nodes selection uses 4 directions. The next representative nodes selection uses relative directions: -15° , 0° , and 15° .

the directions 45° , 135° , 225° , and 315° .

4.4 Honeycomb Strategy

4.4.1 Hexagon Overview

In this section, the terms and the conventions that will be used in the subsequent discussions in relation to hexagons will be defined.

A *hexagon* is a polygon with six edges and six vertices. A *regular hexagon* is a hexagon where all edges are equal in length and all internal angles are all 120° . Subsequent discussions will deal only with regular hexagons.

The *location of a regular hexagon* in the cartesian coordinate system is specified by the location of its center point. The *center point of a regular hexagon* is the point inside the hexagon that has the same distance to every vertex of the hexagon.

The *orientation of a regular hexagon* is specified by the direction from the center point of the hexagon to any vertex of the hexagon. Directions are gauged with respect to the positive Y axis, with positive angles indicate clockwise rotation from the axis. Due to the rotational symmetry of a regular hexagon, the same orientation of a regular hexagon can be specified by six different directions that are 60° apart. For example, a regular hexagon with the orientation of 30° is the same as a regular hexagon with the orientation of 90° , 150° , 210° , 270° , or 330° (Figure 4.9).

Hexagonal tiling is an arrangement of regular hexagons to cover a plane with no overlaps and no gaps between the hexagons. A point on the plane is contained by at least one hexagon. If the point lies on a hexagon edge, it is contained by two hexagons that share the edge. If the point lies on a hexagon vertex, it is contained by three hexagons that share the vertex. The hexagon that contains the point on the plane is called the *container hexagon of the point*.

To refer to a specific hexagon in a hexagonal tiling, a hexagonal coordinate system is introduced. The hexagonal coordinate system specifies two axes that

Figure 4.9: A regular hexagon with 30° , 90° , 150° , 210° , 270° , or 330° orientation

form a 60° angle. The hexagon that is located at the intersection of the two axes is called the *origin hexagon*. The first axis passes through the center of the origin hexagon and the midpoints of a pair of parallel edges of the origin hexagon. The second axis passes through the center of the origin hexagon and the midpoints of another pair of parallel edges of the origin hexagon, forming a 60° angle with the first axis in the counter-clockwise rotation. The first axis will be called the *hexagonal X axis*, and the second axis will be called the *hexagonal Y axis*.

A hexagon in the hexagonal tiling is uniquely identified by an ordered pair of integers (x, y) – its coordinate in the hexagonal coordinate system. A hexagon with coordinate (x, y) in the hexagonal coordinate system can be located by moving x hexagons from the origin hexagon in the positive hexagonal X axis direction, then moving y hexagons in the positive hexagonal Y axis direction. Figure 4.10 shows an example of locating a hexagon in a hexagonal tiling using the hexagonal coordinate system.

Figure 4.10: An example showing how to locate the hexagon with hexagonal coordinate $(3, 2)$ in the hexagonal tiling.

4.4.2 Honeycomb Selection Algorithm

One problem with the fixed branching strategy is that the density of the representative nodes gets higher at a far distance from the source node, and so does the density of the transmission branches. Nodes in the area far away from the source node are covered by many representative nodes simultaneously. This is not desirable according to the minimal overlap characteristics of a good selection strategy.

The honeycomb strategy aims to achieve a constant low density of the representative nodes irrespective of the distance from the source node. The main idea of the strategy is to choose representative nodes and shape the transmissions between representative nodes in such a way that the overall route request flow forms a honeycomb (hexagonal tiling) pattern. The representative nodes form the hexagon vertices while the transmissions between representative nodes form the hexagon edges.

Figure 4.11: Virtual hexagonal tiling constructed over an area. The dot in the center is the source node.

Suppose a source node intends to discover the route to a destination node. A virtual hexagonal tiling can be constructed over the area as shown in Figure 4.11. The length of the hexagon edges is a certain percentage of the directional transmission range, and the orientation of the hexagons is 0° . The virtual hexagonal tiling is constructed in such a way that the source node lies on a hexagon vertex. There are two possible ways to do this. The source node can lie on the hexagon vertex that is incident with the hexagon edges at the directions 0° , 120° , and 240° . Alternatively, the source node can lie on the hexagon vertex that is incident with the hexagon edges at the directions 60° , 180° , and 300° . The second way will be used in the subsequent discussions.

Given the location of the source node and the hexagonal tiling parameters above, there is only one unambiguous way to construct the virtual hexagonal tiling over the area.

The constructed virtual hexagonal tiling gives a hint on the ideal locations of the representative nodes. The representative nodes are ideally located on the

vertices of the hexagons. The edges of the hexagons represent the ideal paths to forward the route request from one representative node to the next representative nodes.

Hexagon edges are directed. The route request packet flows along each hexagon edge in one direction. From the point of view of a hexagon vertex, incident hexagon edges can be categorized as incoming edge, if the direction of the edge is coming towards the vertex, or outgoing edge, if the direction of the edge is going away from the vertex.

As each hexagon vertex is incident with 3 hexagon edges, there are at most 3 transmissions made by each representative node to forward the route request. However, one of the 3 hexagon edges is the incoming edge from where the representative node receives the route request. Since the route request should not be retransmitted in the direction where it comes from, the maximum number of transmissions made by each representative node to forward the route request is reduced to 2. As discussed later, some representative nodes even only need to make a single transmission to forward the route request.

Having the virtual hexagonal tiling constructed, the next step is to determine the direction of the route request flow at each hexagon edge. For this purpose, the virtual hexagonal tiling is divided into 6 zones with the source node at the center as shown in Figure 4.12. Each zone approximately forms a 60° sector.

The zone to which a hexagon belongs can be determined by first calculating the direction from the source node to the center point of the hexagon. If the calculated direction is between the left bound and the right bound of a particular zone, the hexagon belongs to the zone. The left bound and the right bound of a zone are calculated as follows:

$$Bound_{left}(zone) = zone * 60^\circ - 30^\circ \quad (4.3)$$

$$Bound_{right}(zone) = (zone + 1) * 60^\circ - 30^\circ \quad (4.4)$$

Figure 4.12: Dividing the virtual hexagonal tiling into 6 zones, each approximately forms a 60° sector.

Figure 4.13: Flow directions assigned to the hexagon edges (left). The simplified representation of the left hexagon using a single arrow inside the hexagon (right).

For each hexagon, directions are assigned to its edges as shown by the left hexagon in Figure 4.13. The flow originates from one vertex, then is split into two flows. Finally, the two flows merge again at the opposite vertex. To simplify the diagram, the flow around a hexagon will be represented by a single arrow inside the hexagon as shown by the right hexagon in Figure 4.13. The arrow starts from the vertex where the flow originates and points to the vertex where the two flows merge again. The direction represented by the arrow will be called the *hexagonal flow direction*.

Figure 4.14: The assignment of the hexagonal flow direction to the hexagons. All hexagons in the same zone are assigned the same hexagonal flow direction.

All hexagons in the same zone are assigned the same hexagonal flow direction. The hexagonal flow directions corresponding to zones 0, 1, 2, 3, 4, and 5 respectively are 0° , 60° , 120° , 180° , 240° , and 300° . The assignment of the hexagonal flow direction to the hexagons is shown in Figure 4.14.

Finally, replacing the simplified representation of the hexagonal flow with the normal representation, the overall pattern of the route request flow is as shown in Figure 4.15.

Based on the pattern of the route request flow, the source node is the only node transmitting the route request into 3 different directions. Representative nodes are only required to transmit the route request into either 1 or 2 different directions.

The source node performs the initial representative nodes selection in a relatively simple way. Referring to the constructed virtual hexagonal tiling, the source node calculates the ideal locations of the 3 representative nodes to which it will forward the route request. Then, the source node looks into its neighbor

Figure 4.15: The pattern of the route request flow

list to find the neighbor nodes that match the calculated ideal locations. Most likely, neighbor nodes that match the ideal locations perfectly will not be found. Therefore, the source node picks the neighbor nodes that are closest to the ideal locations as the representative nodes.

The initial representative nodes selection in the honeycomb strategy is presented in Algorithm 4.5.

The next representative nodes selection performed by the intermediate nodes is more complex. Upon receiving the route request, the intermediate node constructs the virtual hexagonal tiling in the same way as performed by the source node. The virtual hexagonal tiling is constructed based on the source node location (found in the route request) and the agreed hexagonal tiling parameters (hexagon edge length, hexagon orientation, etc.). As there is only one unambiguous way to construct the virtual hexagonal tiling, all representative nodes and the source node will agree on the same virtual hexagonal tiling.

The intermediate node determines its container hexagon by referring to the

Algorithm 4.5 Initial representative nodes selection in the honeycomb strategy

```

1: let  $L_{src}$  be the neighbor list of the source node  $n_{src}$ 
2: let  $l_{dir}$  be the directional transmission range
3:  $l_{hex} \leftarrow C \times l_{dir}$   $\{C \text{ is some constant scaling factor}\}$ 
4:  $D \leftarrow \{60^\circ, 180^\circ, 300^\circ\}$ 
5:  $N_{rpr} \leftarrow \emptyset$ 
6: for all  $\alpha$  in  $D$  do
7:    $r_{ideal}.location \leftarrow n_{src}.location + (l_{hex}, \alpha)$ 
8:    $r \leftarrow null$ 
9:   for all  $n$  in  $L_{src}$  do
10:    if  $distance(r_{ideal}, n) < distance(r_{ideal}, r)$  or  $r$  is null then
11:       $r \leftarrow n$ 
12:    end if
13:  end for
14:   $N_{rpr} \leftarrow N_{rpr} \cup \{r\}$ 
15: end for
16: return  $N_{rpr}$ 

```

constructed virtual hexagonal tiling. The zone to which the container hexagon belongs can be easily determined using the method described earlier. Following this, the intermediate node calculates the distance from itself to each vertex of the container hexagon. The vertex closest to the intermediate node is considered as the ideal location of the intermediate node. Subsequent calculations to select the representative nodes will be based on this ideal location of the intermediate node rather than the actual location of the intermediate node.

The constructed virtual hexagonal tiling shows the outgoing edges of the vertex representing the intermediate node. These outgoing edges lead to the vertices corresponding to the ideal locations of the representative nodes to be selected. The intermediate node looks into its neighbor list to find the neighbor nodes that match the ideal locations. As a perfect match is unlikely to be found, the intermediate node picks the neighbor nodes that are closest to the ideal locations as the representative nodes.

The next representative nodes selection in the honeycomb strategy is presented in Algorithm 4.6.

It has been mentioned earlier that the length of the hexagon edges is chosen to

Algorithm 4.6 Next representative nodes selection in the honeycomb strategy

```

1: let  $L_{int}$  be the neighbor list of the intermediate node  $n_{int}$ 
2: let  $l_{dir}$  be the directional transmission range
3:  $l_{hex} \leftarrow C \times l_{dir}$   $\{C$  is some constant scaling factor $\}$ 
4:  $N_{rpr} \leftarrow \emptyset$ 
5:  $h_{container} \leftarrow$  the container hexagon of  $n_{int}$ 
6:  $n_{int\_ideal} \leftarrow$  the vertex of  $h_{container}$  closest to  $n_{int}$ 
7:  $D \leftarrow$  the directions of the outgoing edges of  $n_{int\_ideal}$ 
8: for all  $\alpha$  in  $D$  do
9:    $r_{ideal}.location \leftarrow n_{int\_ideal}.location + (l_{hex}, \alpha)$ 
10:   $r \leftarrow null$ 
11:  for all  $n$  in  $L_{src}$  do
12:    if  $distance(r_{ideal}, n) < distance(r_{ideal}, r)$  or  $r$  is null then
13:       $r \leftarrow n$ 
14:    end if
15:  end for
16:   $N_{rpr} \leftarrow N_{rpr} \cup \{r\}$ 
17: end for
18: return  $N_{rpr}$ 

```

be a certain percentage of the directional transmission range rather than equal to the full directional transmission range. The reason is because the actual locations of the representative nodes do not usually match the ideal locations perfectly. Under this condition, the distance from one representative node to the next representative node may be longer than the length of the hexagon edges. Therefore, if the hexagon edges is chosen to be equal to the full directional transmission range, a representative node may not be able to reach the next representative node.

Figure 4.16 and Figure 4.17 show two scenarios of route request flooding using the honeycomb strategy. The first scenario uses TTL value of 5, while the second scenario uses TTL value of 10.

4.4.3 Example

To illustrate how the Honeycomb selection strategy works, an example will be discussed in this section.

Assume a source node S has a packet to send to the destination node D . As the source routing is used in the system, node S is required to attach the complete

Figure 4.16: Route request flooding using the honeycomb strategy ($TTL = 5$)

Figure 4.17: Route request flooding using the honeycomb strategy ($TTL = 10$)

route from node S to node D in the packet header. For this, node S checks its own route cache to find the route to node D . If the route is found in the cache, node S can immediately use the route.

However, if the route is not found in the cache, probably because this is the first time node S has a packet to send to node D or because the previously cached route to node D has expired, node S will now check its own neighbor list. The neighbor list is provided by the neighbor discovery scheme running at node S , possibly at the lower layer, and contains both the omnidirectional and directional neighbors of node S . If node D is found in the neighbor list, a direct route to node D has been found and can be immediately used.

However, if node D is not found in the neighbor list, node S will now need to initiate a route discovery process by sending out a route request packet. The route request packet will not be sent to all its neighbors, but only to some representative nodes. The question now is how to select the representative nodes using the Honeycomb selection strategy.

Assume node S is located at the coordinate $(100, 200)$. The Honeycomb selection strategy begins with node S constructing a virtual hexagonal tiling. There are certain rules that must be followed when constructing the virtual hexagonal tiling such that there is only one unambiguous way to do it. First, the length of the hexagon side has been agreed by all nodes in the system. In this example, assume that the agreed length of the hexagon side is 200 m. Second, the orientation of the hexagon is 0° . Third, node S as the source node must be laid on the hexagon vertex that is incident with the hexagon edges at the directions 60° , 180° , and 300° . Finally, flow directions are assigned to the hexagon edges by following the standard flow pattern as shown in Figure 4.15. The result is shown in Figure 4.18.

The virtual hexagonal tiling acts like a map that will help a node determine the location of the next representative nodes to select. By consulting the virtual hexagonal tiling and by simple geometric calculations, node S finds that

Figure 4.18: Initial representative nodes selection

Figure 4.19: Mapping the ideal locations of the representative nodes to the actual neighbor nodes

the ideal locations of the 3 representative nodes to select are at the coordinates $R_1(-73.21, 300)$, $R_2(273.21, 300)$, and $R_3(100, 0)$, as shown in Figure 4.18.

The next thing node S must do is to map these ideal locations of the representative nodes to the actual neighbor nodes. For this, node S looks into its own neighbor list to see if there is a match between the ideal locations and the actual locations of the neighbor nodes. There is a high chance that an exact match will not be found. Therefore, node S picks the neighbor nodes that are closest to these ideal locations as shown in Figure 4.19. Neighbor nodes R'_1 , R'_2 , and R'_3 are the closest neighbor nodes to the ideal locations R_1 , R_2 , and R_3 , respectively.

Finally, node S sends the route request packet to the selected representative nodes using 3 separate directional transmissions.

When a representative node receives the route request packet, the first thing it needs to do is to check its own neighbor list to see if node D is in the list. If node D is found in the neighbor list, the representative node will immediately construct the route reply packet and send it back to node S following the route that has just been discovered.

However, if node D is not found in the neighbor list, the representative node will need to select the next representative nodes and forward the route request

Figure 4.20: Node A in the virtual hexagonal tiling

packet to them, provided that the TTL value of the route request packet has not reached 0. Again, the question now is how to select the next representative nodes using the Honeycomb selection strategy.

In this example, assume that node A , located at the coordinate $(570, -110)$, has been chosen as the representative node by some previous hop node, and therefore, receives the route request packet from the previous hop node. Now, node A has to forward the route request packet further.

Node A begins the next representative nodes selection by constructing the virtual hexagonal tiling in exactly the same way as node S did. By following the same rules, complemented by the coordinate of node S found in the route request packet, node A is able to reconstruct the virtual hexagonal tiling just like what node S constructed. The result is shown in Figure 4.20.

Node A maps its own location on the constructed virtual hexagonal tiling, and then finds its container hexagon. The container hexagon of node A is shown as

Figure 4.21: Ideal location of node A

the shaded hexagon in Figure 4.20.

Having found the container hexagon, node A calculates the distance from itself to each vertex of the container hexagon. The vertex of the container hexagon that is closest to node A is considered as the ideal location of node A , as shown in Figure 4.21. In this example, the ideal location of node A turns out to be at the coordinate $(619.62, -100)$.

Similar to what node S did, node A uses the virtual hexagonal tiling as a map to help it determine the locations of the representative nodes to select. By consulting the virtual hexagonal tiling, node A finds that there are 2 flows going out from the ideal location of node A , leading to the locations of the 2 representative nodes to select. By simple geometric calculations, node A finds that the ideal locations of the 2 representative nodes to select are at the coordinates $R_1(792.82, 0)$ and $R_2(619.62, -300)$, as shown in Figure 4.22.

The next thing node A must do is to map these ideal locations of the representative nodes to the actual neighbor nodes. For this, node A looks into its own neighbor list to see if there is a match between the ideal locations and the actual locations of the neighbor nodes. There is a high chance that an exact match will not be found. Therefore, node A picks the neighbor nodes that are closest to

Figure 4.22: Next representative nodes selection

Figure 4.23: Mapping the ideal locations of the representative nodes to the actual neighbor nodes

these ideal locations as shown in Figure 4.23. Neighbor nodes R'_1 and R'_2 are the closest neighbor nodes to the ideal locations R_1 and R_2 , respectively.

Finally, node A sends the route request packet to the selected representative nodes using 2 separate directional transmissions.

4.4.4 Why Hexagon?

There are a couple of reasons why regular hexagon is the most suitable for the proposed route request flooding scheme over other types of regular polygons. Note that only regular polygons are considered here, i.e. polygons where all edges are equal in length and all internal angles are also equal. Using irregular polygons will make the algorithm more complex as there will be different cases that need to be handled separately due to the irregularity of the polygon shapes, and the patterns may not be so easy to recognize and formulate.

1. not all regular polygons can be tiled

Regular polygons such as pentagons (5 edges) and heptagons (7 edges) are obviously impossible to tile, not to mention other higher-order polygons such as dodecagons (12 edges). In fact, only 3 types of regular polygons can be

Figure 4.24: Octagon-and-square tiling

tiled, i.e. triangles, squares, and hexagons.

Tilings using a combination of two or more types of regular polygons are possible, for example, the tiling made up of octagons (8 edges) and squares as shown in Figure 4.24. However, this kind of tiling is generally not desirable since the coverage of the representative nodes is not uniform over the entire area. This will be elaborated more in point 3 below.

2. small number of representative nodes is preferred

It has been mentioned that one characteristic of a good selection strategy is small number of neighbor nodes selected as representative nodes, because it translates directly to the number of transmissions required to forward the route request packets. In hexagonal tiling, each vertex is adjacent with 3 other vertices. This means, each node will select at most 3 neighbor nodes as representative nodes. Compare this with the triangular tiling and the square tiling as shown in Figure 4.25. In the triangular tiling, each vertex is adjacent with 6 other vertices, and in the square tiling, each vertex is adjacent with 4 other vertices. This means, each node will select up to 6 neighbor nodes in the triangular tiling, and up to 4 neighbor nodes in the square tiling as representative nodes.

3. minimal overlap is preferred

Figure 4.25: Triangular tiling and square tiling

Table 4.1: Percentage of the area inside the polygon covered by certain number of representative nodes

Num of overlaps	Triangle	Square	Hexagon
2	-	17.36%	58.16%
3	37.24%	51.13%	41.84%
4	62.76%	31.51%	-
Total	100%	100%	100%

Another characteristic of a good selection strategy is minimal overlap, i.e. each node in the system should be covered by as few representative nodes as possible to avoid transmission redundancy. Figure 4.26 shows the coverage overlaps for different types of polygons. Based on the figure and using simple geometric calculation, the percentage of the area inside the polygon that is covered by certain number of representative nodes can be found. The calculation results are shown in Table 4.1. From the table, it can be seen that a large percentage of the area inside the triangle is covered by 4 representative nodes, while a large percentage of the area inside the hexagon is covered only by 2 representative nodes.

Yet another characteristic of a good selection strategy is total coverage, i.e. each node in the system should be covered by at least one representative node. Based on this characteristic, octagon can immediately be excluded from further consideration since the area around the center of the octagon is not covered by any representative nodes as shown in Figure 4.26. Moreover,

Figure 4.26: Coverage overlaps for different types of polygons

if octagons and squares are tiled together as shown in Figure 4.24, most of the area inside the square will be covered by the 4 representative nodes at the vertices of the square since these representative nodes are relatively close to each other. This will cause non-uniform coverage of the representative nodes over the entire area. The area inside the square will be heavily covered while the area inside the octagon will only be slightly covered (and even completely not covered at the center of the octagon).

4.5 Performance Comparison

Simulations were carried out to compare the performances of the proposed route request flooding strategies with other strategies. In these simulations, the performances of the fixed branching strategy and the honeycomb strategy were compared against the performances of the traditional DSR flooding strategy [17], the directional sweeping flooding strategy [23], the Directional Self-Pruning (DSP) strategy, and the Link-Reduction-based (LR) strategy.

4.5.1 Simulation Setup

The simulated system area was a square of size $3 \text{ km} \times 3 \text{ km}$. 800 static nodes were randomly distributed inside the area such that the node density was approximately 88.89 nodes/km^2 . The omnidirectional transmission range used was 200 m, while the directional transmission range used was 400 m. The beam width of the directional transmission was 30° .

Simulations were repeated 100 times where each simulation run differs in the positions of the nodes inside the area. In each simulation run, a pair of source-destination nodes were chosen. The source node was instructed to initiate the route request flooding process. This process was repeated for each of the 4 route request flooding strategies being studied.

At the end of each simulation run, a number of data were collected for the purpose of analysis and comparison. These include:

- whether the route from the source node to the destination node was found
- the number of hops in the found route
- the number of nodes involved in the route request flooding process
- the number of transmissions made by the nodes involved in the route request flooding process

The fixed branching strategy used the following predefined directions for the initial representative nodes selection: 60° , 180° , and 300° . While for the next representative nodes selection, the following predefined relative directions were used: -60° , 0° , and 60° .

The length of the hexagon sides used in the honeycomb strategy was 85% of the directional transmission range. That was about 340 m.

The directional sweeping flooding strategy was a direct extension of the traditional DSR flooding strategy. It behaves in the same way as the traditional DSR

Figure 4.27: The route discovery success rate

flooding strategy except that the route request packets were broadcast using directional transmissions. In order to cover all directions, a node broadcasts the route request packets using a number of directional transmissions sequentially (sweeping). As the beam width of the directional transmissions was 30° , a node needs $360^\circ/30^\circ = 12$ directional transmissions to broadcast a route request packet.

4.5.2 Simulation Results

The first performance metric being studied was the route discovery success rate. The route discovery success rate was defined as the number of simulation runs in which the route from the source to the destination was successfully found. Figure 4.27 shows the route discovery success rates of the 6 strategies being studied for 3 different TTL values: 5, 10, and 15.

The traditional DSR flooding strategy had the lowest route discovery success rate for all of the TTL values used. This was expected since the traditional DSR was the only strategy that used omnidirectional transmissions of the 6 strategies being studied. As the omnidirectional transmission range was shorter than the directional transmission range, the route request flooding by the traditional DSR

covered a smaller area for the same TTL values. Network partitions were also more likely to happen with the omnidirectional transmissions due to the shorter transmission range.

The directional sweeping strategy scored the highest for all of the 3 TTL values used. This was due to the fact that the directional sweeping strategy broadcast the route request packet indiscriminately to virtually all of its directional and omnidirectional neighbors. This flooding strategy effectively behaved as if the route request packets had been transmitted by the high-power omnidirectional transmissions with the same transmission range as the directional transmissions. Nevertheless, the high route discovery success rates of the directional sweeping strategy came with a number of costs as discussed below in this section.

The fixed branching, honeycomb, and DSP strategies achieved comparable route discovery success rates, while the LR strategy achieved lower rates.

At TTL value of 15, the fixed branching, honeycomb, directional sweeping, DSP, and LR strategies exhibited similar route discovery success rates. At this TTL value, the 5 strategies were able to discover the routes in almost all simulation runs, while the traditional DSR was only able to discover the routes in about 83% of the simulation runs.

Figure 4.28 shows the average number of hops in the route. Note that only those simulation runs in which the route to the destination was successfully found were included in the calculation of the average. This means the number of simulation runs included in the calculation of the average could be different for each strategy. Therefore, the average number of hops in the route should be interpreted with the route discovery success rate in mind.

At TTL value of 5, the average numbers of hops in the route for all of the 6 strategies were similar, at about 4-5 hops. The result at this TTL value did not give much information about the relative performance of the strategies. The only valuable information that can be obtained from this result was that TTL value of 5 was not enough to cover the system area since the average number of

Figure 4.28: The average number of hops in the route

hops was just slightly below TTL value.

At TTL values of 10 and 15, the relative performance of the strategies exhibited similar pattern. The traditional DSR needed the largest number of hops on average, again due to the use of omnidirectional transmissions. The average numbers of hops for the other 5 strategies are comparable, with the directional sweeping strategy showing a slightly lower average number of hops. The LR strategy needed a slightly higher average number of hops, possibly because the routes found using LR had to follow the localized branching tree computed using the Prim's Algorithm. The route between two nodes in the localized branching tree was not always close to the shortest path between the two nodes.

The more interesting results to discuss are the average number of nodes involved in the route discovery process as shown in Figure 4.29. A node was considered to be involved in the route discovery process if the routing protocol running at the node received and processed the route request packets (and optionally retransmitted the route request packets). Therefore, if a node received the route request packets, but the packets were discarded at the layer below the routing protocol (e.g. dropped at the MAC layer), the node was not considered to be

Figure 4.29: The average number of nodes involved in the route discovery process

involved in the route discovery process.

The figure clearly shows that the directional sweeping strategy involved significantly high number of nodes in the route discovery process. At TTL value of 5, it involved about 500 nodes or about 62.5% of the total number of nodes in the simulation system. At TTL values of 10 and 15, almost all nodes in the simulation system were involved in the route discovery process. These results were expected since the directional sweeping strategy broadcast the route request packets to all neighbors. As the result, all neighbors received and must process the route request packets. Since the broadcast reached omnidirectional neighbors as well as directional neighbors, the number of involved nodes increased drastically.

Similarly, the average numbers of nodes involved in the route discovery process for DSP and LR were significantly high as well. The main reason for this was because DSP and LR aim to achieve full delivery of the broadcast packet to all nodes in the system. On the other hand, fixed branching strategy and honeycomb strategy were able to achieve low average numbers of nodes involved in the route discovery process because these strategies only forwarded the route request packet to the representative nodes.

Figure 4.30: The average number of transmissions during the route discovery process

The traditional DSR strategy and the fixed branching strategy involved comparable number of nodes in the route discovery process at TTL values of 5 and 15. However, at TTL value of 15, the traditional DSR involved significantly higher number of nodes.

The honeycomb strategy achieved the best results for all the 3 TTL values by having significantly lower number of nodes involved in the route discovery process. The strategy involved only about 5%, 12%, and 14% of the total number of nodes in the simulation system for TTL values of 5, 10, and 15, respectively.

The last result worth discussing is the average number of transmissions during the route discovery process. The result is shown in Figure 4.30. Note that for the traditional DSR strategy, the figure shows the number of omnidirectional transmissions, while for the other strategies, the figure shows the number of directional transmissions.

It is obvious that the directional sweeping strategy had significantly higher number of transmissions as compared with the other strategies. The reason for this is also obvious: nodes broadcast the route request packets by using sequential directional transmissions, sweeping into all directions. In the simulation, each

node had to broadcast sequentially by using 12 directional transmissions since the beam width used was 30° . This, coupled with the high number of nodes involved in the route discovery process, explained the significantly high number of transmissions during the route discovery process.

The honeycomb strategy achieved the lowest number of transmissions for all of the 3 TTL values. It was about 62% lower than the number of transmissions for the traditional DSR strategy at the TTL value of 5, about 54% lower at the TTL value of 10, and about 42% lower at the TTL value of 15. The honeycomb strategy was able to achieve such low numbers of transmissions despite the fact that some of the representative nodes had to transmit the route request packets into more than 1 directions and that it covered larger areas than the traditional DSR strategy did.

Chapter 5

Conclusions and Future Works

5.1 Conclusions

The use of directional antennas in ad hoc networks has opened up a wide range of new possibilities in enhancing the efficiency of the network. Some of the many benefits that directional antennas have to offer include the increase in the overall network bandwidth, the reduction in the interference level and the power consumption, the lower occurrences of network partitions and the lower number of hops to reach the destination node. These benefits may be attained provided that the antennas are used properly. Despite the benefits, directional antennas have also brought new challenges that need to be addressed in order to use them efficiently.

One of the challenges that are inherent in the use of directional antennas in ad hoc networks is neighbor discovery. A node needs to know the location of the destination node in order to form the transmission beam in the direction of the destination node. In this thesis, an efficient neighbor discovery scheme for ad hoc networks with directional antennas has been proposed. The proposed scheme is able to discover neighbors beyond the omnidirectional transmission range without the use of high-power broadcast. The main idea of the proposed scheme is to select a small subset of the omnidirectional neighbors as the representative nodes

that will help the requesting node gather information about its neighbors. The neighbor discovery algorithm works in a distributed manner.

The proposed scheme exhibits a number of benefits. First, the proposed scheme does not require all neighbors to be involved in the discovery process. Only the representative nodes are required to send neighbor information to the requesting node. Second, there is no overlap of information being sent by the different representative nodes. Each representative node sends only the neighbor information that has not been covered by the previous representative nodes. As the result, the neighbor discovery process is fast and efficient.

Coverage analysis and simulations show that the proposed neighbor discovery scheme works better in a dense population. As the population density increases, the area covered by the scheme also increases. In addition, the percentage of neighbors involved in the discovery process drops as the population density increases. Comparative performance evaluation shows that the proposed scheme is able to perform neighbor discovery faster than other basic neighbor discovery schemes while still being able to obtain complete neighbor information. Hence, the proposed neighbor discovery scheme is most suitable for use in the situation where a node has to be able to perform its function efficiently as soon as it joins an existing system.

The neighbor information provided by neighbor discovery scheme is not only good for assisting the directional MAC protocol to form the transmission beam in the direction of the destination node. The same neighbor information may also be exploited by higher layer protocols to help them make better decisions and perform their tasks more efficiently.

In this thesis, the use of the neighbor information to assist the routing protocol has been explored. An efficient scheme for flooding the route request packets in ad hoc networks with directional antennas has been proposed. The neighbor information is utilized by the source node to strategically select a small subset of neighbors as representative nodes that will help to forward the route request

packets to farther nodes. The route request packets are forwarded to the selected representative neighbors using directional transmissions instead of being broadcast to all neighbors. Upon receiving the route request packet, each representative node searches for the destination node in its own neighbor list. If the search fails, the representative node selects the next representative neighbors and forward the route request packet further.

The effectiveness of the flooding scheme depends on the strategies being used to select the representative nodes. A good selection strategy should involve minimal number of representative nodes that are sufficient to cover the entire system. In this thesis, two strategies have been proposed for the selection of the representative nodes.

The Fixed Branching Strategy works by choosing one neighbor in each of the predefined forwarding directions as the representative node. This strategy creates a flooding pattern that looks like the branching of a tree. One drawback of this strategy is that the density of representative nodes gets higher at a far distance from the source node, and so does the density of the transmission branches.

The Honeycomb Strategy aims to achieve a constant low density of the representative nodes irrespective of the distance from the source node. The strategy works by choosing representative nodes and shaping the transmissions between representative nodes in such way that the resulting flooding pattern forms a honeycomb (hexagonal tiling) pattern.

Simulation results show that both strategies are able to achieve a high route discovery success rate comparable to the one exhibited by the directional sweeping strategy, while at the same time maintaining a lower number of nodes involved in the discovery process and a lower number of transmissions during the discovery process. In addition, the average number of hops achieved by both strategies is comparable to the one achieved by the directional sweeping strategy.

5.2 Future Works

Moving forward, there are a number of research areas that can be explored further based on the work presented in this thesis:

- *Extending the neighbor discovery scheme for more challenging scenarios*

There are a number of more challenging scenarios in which the proposed neighbor discovery scheme may not be able to perform well. One such scenario is the mobility scenario. Choosing the representative nodes efficiently and accurately under mobility condition proves to be a challenging task. Studying the impact of mobility on the performance of neighbor discovery may give some insights on the directions to take to design a reliable neighbor discovery scheme for mobility condition.

The proposed neighbor discovery scheme is also not suitable for use during the system start up. The node performing neighbor discovery relies on the help of the representative nodes that will provide the neighbor information. During the system start up, all nodes in the system have minimal knowledge about their own neighbors. Therefore, they are less capable to give neighbor information to other nodes that require it. Even worse, the nodes may depend on each other to provide neighbor information during the system start up. A way to perform neighbor discovery quickly and efficiently during the system start up can be explored.

- *Maintaining the neighbor information*

The proposed neighbor discovery scheme particularly focused on a way to quickly and efficiently discover neighbors when a node joins an existing system such that the node can immediately start communicating with other nodes in the system. After the initial neighbor discovery, the neighbor information has to be maintained and kept updated. Future research may explore various ways to maintain the neighbor information in a quick and efficient manner. The concept of representative nodes, which has been used exten-

sively in this thesis, may be used again to efficiently maintain the neighbor information.

- *Optimizing the discovered path*

Route discovery using the proposed route request flooding scheme does not always result in the optimized route (the route with the lowest number of hops). The reason is because the route request packets are only forwarded to the representative nodes. As the consequence, only representative nodes can be included in the discovered route. However, there may exist other routes to the destination node that have lower number of hops, but include some non representative nodes.

One possible approach to optimize the discovered path is by using two-phase route discovery. In the first phase, the proposed route request flooding scheme is used. This will allow an initial route to the destination, possibly not optimized, to be discovered efficiently. In the second phase, a kind of guided route request flooding is performed. This time, the route request packets are flooded only about the route discovered in the first phase. The expectation is that another more optimized route to the destination can be found in the second phase.

Another possible approach is to perform the route optimization when the route reply packet is being forwarded to the source node. Each node in the route once again uses its local neighbor information to see if there exists another node that may improve the performance or the efficiency of the route based on some criteria. If such node is found, the current node in the route may choose to pass its duty to the better node.

Other optimization goals besides reducing the number of hops in the route may also be explored. Some examples include load balancing, avoiding congestions, and reducing co-interference between different routes. Neighbor information may be used again to assist the optimization process.

- *Enhancing the proposed representative node selection strategies*

The proposed fixed branching strategy can be evolved into the dynamic branching strategy where the forwarding directions are adjusted dynamically based on certain criteria. For example, as the route request packets are forwarded farther from the source node, the number of forwarding directions can be reduced and the spread angle between the forwarding directions can be made narrower. The goal is to reduce the density of the representative nodes and the transmission branches at a far distance from the source node.

In the proposed honeycomb strategy, there are some representative nodes that receive the same route request packet from two different neighbors. These are the nodes at which two flows merge as shown in the hexagonal route request flow pattern. The strategy can be improved by reshaping the route request flow pattern such that fewer or even none of the representative nodes receive the same route request packet more than once. This will reduce the number of transmissions further. The impact of this reduction on the route discovery success rate may also need to be studied.

Author's Publications

R. A. Santosa, B.-S. Lee, C. K. Yeo, and T. M. Lim, "Distributed neighbor discovery in ad hoc networks using directional antennas," in *IEEE International Conference on Computer and Information Technology*, Sep. 2006, pp. 97-98.

R. A. Santosa, B.-S. Lee, C. K. Yeo, and T. M. Lim, "An efficient scheme to discover neighbors beyond omnidirectional transmission range," in *IEEE Consumer Communications and Networking Conference*, Jan. 2008.

R. A. Santosa, B.-S. Lee, and C. K. Yeo, "Efficient DSR Route Request Flooding with Directional Antennas", submitted to *Elsevier Computer Networks Journal*, Jul. 2008.

Glossary

Directional area of node x The area around node x that is reachable by the directional transmission of node x , but is not reachable by the omnidirectional transmission of node x . This area is donut-shaped with node x at its center point.

Directional neighbors of node x Nodes that are located inside the directional area of node x .

Directional transmission range of node x The furthest distance from node x that is reachable by the directional transmission of node x .

Hexagon A polygon with six edges and six vertices.

Hexagonal tiling An arrangement of regular hexagons to cover a plane with no overlaps and no gaps between the hexagons.

Initial representative nodes selection The selection of representative nodes by the source node.

Neighbor area of node x The area around node x that is reachable by either the omnidirectional or the directional transmission of node x . This area is circular with node x at its center point.

Neighbors of node x All omnidirectional and directional neighbors of node x .

Next representative nodes selection The selection of representative nodes by the intermediate node.

Omnidirectional area of node x The area around node x that is reachable by the omnidirectional transmission of node x . This area is circular with node x at its center point.

Omnidirectional neighbors of node x Nodes that are located inside the omnidirectional area of node x .

Omnidirectional transmission range of node x The furthest distance from node x that is reachable by the omnidirectional transmission of node x .

Regular hexagon A hexagon where all edges are equal in length and all internal angles are all 120° .

Representative nodes of node x (neighbor discovery) A subset of omnidirectional neighbors of node x that helps node x gather information about all its neighbors.

Representative nodes of node x (route request flooding) A subset of neighbors of node x that are chosen to forward the route request packet from the source node further.

Bibliography

- [1] P. H. Lehne and M. Pettersen, “An overview of smart antenna technology for mobile communications systems,” *IEEE Communications Surveys*, vol. 2, no. 4, pp. 2–13, Fourth Quarter 1999.
- [2] R. Vilzmann and C. Bettstetter, “A survey on MAC protocols for ad hoc networks with directional antennas,” in *Proc. EUNICE Open European Summer School*, Colmenarejo, Spain, Jul. 2005.
- [3] R. R. Choudhury, X. Yang, R. Ramanathan, and N. H. Vadya, “Using directional antennas for medium access control in ad hoc networks,” in *Proc. ACM MobiCom*, Sep. 2002, pp. 59–70.
- [4] R. R. Choudhury and N. H. Vaidya, “Deafness: A MAC problem in ad hoc networks when using directional antennas,” in *Proc. IEEE ICNP*, 2004, pp. 283 – 292.
- [5] C. Perkins, E. Belding-Royer, and S. Das, “Ad hoc On-Demand Distance Vector (AODV) Routing,” RFC 3561 (Experimental), Jul. 2003. [Online]. Available: <http://www.ietf.org/rfc/rfc3561.txt>
- [6] I. D. Chakeres and E. M. Belding-Royer, “The Utility of Hello Messages for Determining Link Connectivity,” in *Proceedings of the 5th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, Honolulu, HI, October 2002, pp. 504–508.

- [7] V. C. Giruka and M. Singhal, "Hello protocols for ad-hoc networks: Overhead and accuracy tradeoffs," in *Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM'05)*, Jun. 2005.
- [8] E. B. Hamida, G. Chelius, and E. Fleury, "Revisiting neighbor discovery with interferences consideration," in *PE-WASUN '06: Proceedings of the 3rd ACM international workshop on Performance evaluation of wireless ad hoc, sensor and ubiquitous networks*. New York, NY, USA: ACM, 2006, pp. 74–81.
- [9] M. J. McGlynn and S. A. Borbash, "Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks," in *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*. New York, NY, USA: ACM, 2001, pp. 137–145.
- [10] G. Alonso, E. Kranakis, R. Wattenhofer, and P. Widmayer, "Probabilistic protocols for node discovery in ad-hoc, single broadcast channel networks," *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, pp. 8 pp.–, 22-26 April 2003.
- [11] G. Alonso, E. Kranakis, S. C., R. Wattenhofer, and P. Widmayer, "Randomized protocols for node discovery in ad-hoc multichannel broadcast networks," in *2nd Annual Conference on Adhoc Networks and Wireless (ADHOC-NOW'03)*, Oct. 2003, pp. 104–115.
- [12] T. Ren, I. Koutsopoulos, and L. Tassiulas, "Efficient media access protocols for wireless LANs with smart antennas," in *Wireless Communications and Networking, 2003*, vol. 2, Mar. 2003, pp. 1286–1290.
- [13] M. E. Steenstrup, "Neighbor discovery among mobile nodes equipped with smart antennas," in *Proc. 3rd Scandinavian Workshop on Wireless Ad-hoc Networks*, May 2003.

- [14] S. Vasudevan, J. Kurose, and D. Towsley, "On neighbor discovery in wireless networks with directional antennas," in *Proc. IEEE INFOCOM 2005*, vol. 4, Mar. 2005, pp. 2502–2512.
- [15] G. Jakllari, W. Luo, and S. V. Krishnamurthy, "An integrated neighbor discovery and MAC protocol for ad hoc networks using directional antennas," in *Proc. IEEE WoWMoM*, Jun. 2005, pp. 11–21.
- [16] R. Ramanathan, J. Redi, C. Santivanez, D. Wiggins, and S. Polit, "Ad hoc networking with directional antennas: A complete system solution," *IEEE J. Sel. Areas Commun.*, vol. 23, pp. 496–506, Mar. 2005.
- [17] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, Imielinski and Korth, Eds. Kluwer Academic Publishers, 1996, vol. 353. [Online]. Available: citeseer.ist.psu.edu/johnson96dynamic.html
- [18] A. Nasipuri, J. Mandava, H. Manchala, and R. Hiromoto, "On-demand routing using directional antennas in mobile ad hoc networks," *Computer Communications and Networks, 2000. Proceedings. Ninth International Conference on*, pp. 535–541, 2000.
- [19] A. Saha and D. Johnson, "Routing improvement using directional antennas in mobile ad hoc networks," *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*, vol. 5, pp. 2902–2908 Vol.5, 29 Nov.–3 Dec. 2004.
- [20] B. Hu and H. Gharavi, "Dsr-based directional routing protocol for ad hoc networks," *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pp. 4936–4940, 26–30 Nov. 2007.
- [21] H. Gossain, T. Joshi, C. Cordeiro, and D. Agrawal, "A cross-layer approach for designing directional routing protocol in manets," *Wireless Communications and Networking Conference, 2005 IEEE*, vol. 4, pp. 1976–1981 Vol. 4, March 2005.

- [22] T. Joshi, C. Cordeiro, S. Vogety, J. Yin, H. Gossain, and D. Agrawal, "Broadcasting over switched single beam antennas," *Networks, 2004. (ICON 2004). Proceedings. 12th IEEE International Conference on*, vol. 2, pp. 671–675 vol.2, Nov. 2004.
- [23] R. Choudhury and N. Vaidya, "On ad hoc routing using directional antennas," 2002. [Online]. Available: citeseer.ist.psu.edu/choudhury02ad.html
- [24] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *Mobile Computing and Networking*, 2000, pp. 243–254. [Online]. Available: citeseer.ist.psu.edu/karp00gpsr.html
- [25] R. Jain, A. Puri, and R. Sengupta, "Geographical routing using partial information for wireless ad hoc networks," *Personal Communications, IEEE [see also IEEE Wireless Communications]*, vol. 8, no. 1, pp. 48–57, Feb 2001.
- [26] J. Li and P. Mohapatra, "Panda: a novel mechanism for flooding based route discovery in ad hoc networks," *Wirel. Netw.*, vol. 12, no. 6, pp. 771–787, 2006.
- [27] C. Hu, Y. Hong, and J. Hou, "On mitigating the broadcast storm problem with directional antennas," *Communications, 2003. ICC '03. IEEE International Conference on*, vol. 1, pp. 104–110 vol.1, May 2003.
- [28] F. Dai and J. Wu, "Efficient broadcasting in ad hoc wireless networks using directional antennas," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 17, no. 4, pp. 335–347, April 2006.
- [29] L. Ding, Y. Shao, and M. Li, "On reducing broadcast transmission cost and redundancy in ad hoc wireless networks using directional antennas," *Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE*, pp. 2295–2300, 31 2008–April 3 2008.

- [30] Qualnet network simulator 3.9, scalable network technologies. [Online].
Available: <http://www.scalable-networks.com/>