MULTIPOINT COMMUNICATIONS FOR WIRELESS AD HOC NETWORKS

by

CHUNHUI ZHU

A dissertation submitted to the Graduate Faculty in Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

2008

This manuscript has been read and accepted for the Graduate Faculty in Engineering in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

Date	Professor **Myung J. Lee** Chair of Examining Committee
Date	Dean **Mumtaz Kassir** Executive Officer

Professor **Tarek N. Saadawi**
The City University of New York

Professor **Yi Sun**
The City University of New York

Professor **Kaliappa Ravindran**
The City University of New York

Professor **Jizhong Xiao**
The City University of New York
Supervision Committee

THE CITY UNIVERSITY OF NEW YORK

MULTIPOINT COMMUNICATIONS FOR WIRELESS AD HOC NETWORKS

by

CHUNHUI ZHU

Advisor: Professor Myung J. Lee

Abstract

Wireless ad hoc network refers to a class of wireless networks in which devices are connected in a fully distributed manner. Devices in such networks tend to have short transmission ranges, and communicate with other devices through wireless links over multiple hops. Because devices are not directly connected, some sort of routing is needed for data packets to be transmitted from one device to another.

Multipoint communications refers to a communication mode in which the participating parties involved are more than just a sender and a receiver. It could mean the communication among a special group of devices in the network (i.e. multicasting) or among all devices in the network (i.e. broadcasting). This research work concentrates on the design and analysis of multipoint routing algorithms and protocols, including multicast and efficient broadcast routing, for wireless ad hoc networks.

We started our work with evaluating and enhancing the famous Multicast Ad hoc On-Demand Distance Vector (MAODV) [1] routing protocol. Our study showed that MAODV has problems on multicast tree maintenance and robustness. We enhanced the

protocol by introducing new mechanisms and algorithms to the protocol to make it workable.

In the second part of our work, we designed a dynamic RTT-based algorithm which can dramatically reduce the route discovery latency of reactive ad hoc routing protocols. We evaluated our algorithm by applying it to MAODV and achieved great improvement on route discovery latency.

In the third part of our work, we developed a very efficient multicast routing protocol for wireless ad hoc networks with resource constrains. This algorithm fully utilizes the "logical tree + local link state" information so that a multicast algorithm with very low overhead can be achieved.

In the last part of this work, we study the "Broadcast Strom" problem in wireless ad hoc networks and developed a "smart" broadcast scheme that is simple, highly efficient, robust and scalable. This fully distributed algorithm introduces no control overheads (in the terms of control traffic), and can adapt to node density and network movement at no extra costs.

To my dad and my mom

学而不思则罔，思而不学则殆
— 孔子

Learning without thought is labor lost; thought without learning is perilous.
— Confucius

Acknowledgements

I'm full of gratitude to Dr. Myung J. Lee, my Ph.D. advisor, for his guidance and patience. He has always been generous in sharing his extensive knowledge, inspirational ideas, and profound insight. Without his direction and support, this work would not have been possible.

I'm also grateful to Dr. Tarek N. Saadawi, Dr. Yi Sun, Dr. Jizhong Xiao and Dr. Kaliappa Ravindran, the members of my supervisory committee, for taking time to review this thesis and make valuable comments on my work.

I feel so fortune that I have been working in the Center for Information Networking and Telecommunication (CINT), and Samsung-CUNY Joint Lab of CCNY. I appreciate the great helps received from every individual of this energetic group. Special thanks go to Peixiang Gong, who gave me many helps on UNIX-related problems during my early years of study. I also want to thank the colleagues I have worked with during my study for their help and friendship. They are Dr. Yong Liu, Dr. Jianliang Zheng, Xuhui Hu, Dr. Taekyoung Kwon, Min Wang, Changshi Zhou, Dr. Zhengliang Yi, Dr. Guanhua Ye, Dr. Osama Hussein, Alberto Aponte and Ahmed Abd El Al.

I would like to show my great appreciation to my parents, who, with their love, expectation, encouragement and financial support, help me and support me in all aspects. Thank you so much, dad and mom.

I would also like to thank Mr. Lyes Guemari and National Institute of Standards and Technology (NIST) for contributing their OPNET model of unicast AODV. Their design of AODV provided a good reference for our design of Multicast AODV.

Last, but not the least, I want to thank the financial supporters of my Ph.D. research; the U. S. Army Research Laboratory, the EE Department of the Engineering School of CCNY, the Graduate School and University Center of CUNY and the Samsung Advanced Institute of Technology of Samsung Electronics Co. Korea. Without your support, this work would be a “mission impossible”.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ABR	Associativity-Based Routing
ACK	Acknowledgement
AMRIS	Ad hoc Multicast Routing protocol utilizing Increasing id-numberS
AMRoute	Ad hoc Multicast Routing protocol
AODV	Ad Hoc On Demand Distance Vector
BCL	Broadcast Control Layer
CAMP	Core-Assisted Mesh Protocol
CD	Collision Detection
CTS	Clear To Send
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
DCF	Distributed Coordination Function
DIFS	DCF Inter-Frame Spacing period
GC	Group Coordinator
GL	Group Leader
GM	Group Member
GPS	the Global Positioning System
GRPH	Group Hello
G-JREP	Group Join Reply
G-JREQ	Group Join Request
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol

ISI	Inquiry Scans Interval
LAM	Lightweight Adaptive Multicast
MAC	Medium Access Control
MACT	Multicast Activation
MANET	Mobile Ad Hoc Network
MAODV	Multicast Ad-hoc On-demand Distance Vector
MC	Mesh Coordinator
MR	Multicast Router
NACK	Negative Acknowledgement
NSMP	Neighbor-Supporting Multicast Protocol
ODMRP	On-Demand Multicast Routing Protocol
PCF	Point Coordination Function
PDA	Person Digital Assistant
PPP	Point-to-Point Protocol
QoS	Quality of Service
RBM	Reservation-Based Multicast
RREP	Route REPly packet
RREQ	Route REQuest packet
RTS	Request To Send
RTT	Round Trip Time
SIFS	Short Inter-Frame Spacing
SNIR	Signal to Noise and Interference Ratio
SSR	Signal Stability Routing

TCP	Transmission Control Protocol
TL	Tree Level
TORA	Temporally-Ordered Routing Algorithm
TTL	Time-To-Live
UDP	User Datagram Protocol
WPAN	Wireless Personal Area Network
WLAN	Wireless Local Area Network

1 GENERAL INTRODUCTION

1.1 Background of the Work

This section gives background information on wireless ad hoc networks. The topics covered are distinct features of wireless ad hoc network, multicast and broadcast in wireless ad hoc networks.

1.1.1 Distinct Features of Wireless Ad Hoc Networks

Wireless ad hoc network is very different from traditional wired networks and brings many new challenges to its design and applications. We will discuss the distinct features of wireless ad hoc network in the aspects of physical layer (the wireless channel), the MAC layer and the Network layer.

1.1.1.1 Wireless Channel

Unlike the communications in the traditional wired network, wireless channels are extremely random and very difficult to analyze [30][37]. The transmitted signal might have experienced reflection, diffraction and scattering when it arrives at the receiver. Therefore, the error rate of wireless channel is much higher that in the wired media.

In order to describe the characteristics of wireless channel, researchers have to rely on modeling techniques [30]. There are two aspects of channel modeling, large-scale modeling and small-scale modeling. Large-scale models try to predict the mean signal strength for different separation distance between the transmitter and the receiver; while small-scale models attempt to characterize the rapid fluctuations of the received signal strength (caused by multipath effect, Doppler shifts and etc.) over very short travel distance or short time durations. Typical large-scale models include Friis free space

model and ground reflection (2-way) model; Rayleigh and Recian fading distributions are commonly used to analyze the small-scale fadings.

1.1.1.2 The MAC Layer

The Media Access Control (MAC) data communication protocol sub-layer is a part of the data link layer specified in the seven-layer OSI model (layer 2). It provides addressing and channel access control mechanisms that make it possible for several terminals or network nodes to communicate within a multipoint network, typically a wireless local area network (WLAN) [29] or wireless person area network (WPAN) [87].

The MAC sub-layer acts as an interface between the Logical Link Control sublayer and the Physical layer. The MAC layer provides an addressing mechanism called MAC address. This is a unique 64-bit number assigned to each network adapter, making it possible to deliver data packets to a destination within a network.

The MAC layers of wireless networks are very different from that of the wired networks. The following are the distinct features of a typical wireless MAC layer protocol of WLAN and WPAN.

A. Carrier Sense Multiple Access with Collision Avoidance

The basic media access mechanism in both Wireless LAN and Wireless PAN is Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) [29][87]. It is different from the MAC layer of Ethernet, which is CSMA/CD (CD standing for Collision Detection). We will first introduce the mechanism of CSMA, then talk about how CA can partially solve the packet collision problem of CSMA.

A CSMA protocol works as follows: A station desiring to transmit senses the medium. If the medium is busy (i.e. some other station is transmitting) then the station will defer its transmission to a later time; if the medium is sensed free then the station is allowed to transmit.

These kinds of protocols are very effective when the medium is not heavily loaded, since it allows stations to transmit with minimum delay. But there are always chances that stations transmit at the same time since they sensed the medium free and decided to transmit at once. The simultaneous transmissions of packets lead to packet collisions – packets overlay in time cannot be decoded correctly.

These collision situations must be identified, so the MAC layer can retransmit the packet by itself and not by upper layers, which would cause significant delay. In the Ethernet case, this collision is recognized by the transmitting stations which go to a retransmission phase based on an exponential random backoff algorithm.

While the collision detection (CD) mechanism is a good idea on a wired LAN, it cannot be used on a WLAN or WPAN environment, because of two main reasons:

- Implementing a CD mechanism would require the implementation of a full duplex radio, capable of transmitting and receiving at once, an approach that would increase the price significantly.
- On a wireless environment we cannot assume that all stations hear each other (which is the basic assumption of the collision detection scheme), and the fact

that a station willing to transmit and senses the medium free, doesn't necessarily mean that the medium is free around the receiver area.

In order to overcome these problems, the WLAN and WPAN standards use a Collision Avoidance (CA) mechanism together with a positive acknowledge scheme, as follows:

A station willing to transmit senses the medium, if the medium is busy then it defers. If the medium is free for a specified time (i.e. DIFS) then the station is allowed to transmit, the receiving station will check the CRC of the received packet and send an acknowledgement packet (ACK). Receipt of the acknowledgement will indicate the transmitter that no collision occurred. If the sender does not receive the acknowledgement then it will retransmit the fragment until it gets acknowledged or thrown away after a given number of retransmissions.

B. Virtual Carrier Sense

In order to reduce the probability of two stations colliding because they cannot hear each other, the WLAN defines a Virtual Carrier Sense mechanism.

A station willing to transmit a packet will first transmit a short control packet called RTS (Request to Send), which will include the source address, destination address, and the duration of the following transaction (including the packet and the respective ACK). The destination station will respond (if the medium is free) with a response control packet call CTS (Clear to Send), which will include the same duration information.

All stations receiving either the RTS and/or the CTS, will set their Virtual Carrier Sense indicator, the Network Allocation Vector (NAV), for the given duration, and will use this information together with the physical carrier sense with sensing the medium.

This mechanism reduces the probability of a collision on the receiver area by a station that is "hidden" from the transmitter, to the short duration of the RTS transmission, because the station will hear the CTS and "reserve" the medium as busy until the end of the transaction. The duration information on the RTS also protects the transmitter area from collisions during the ACK (by stations that are out of range from the acknowledging station).

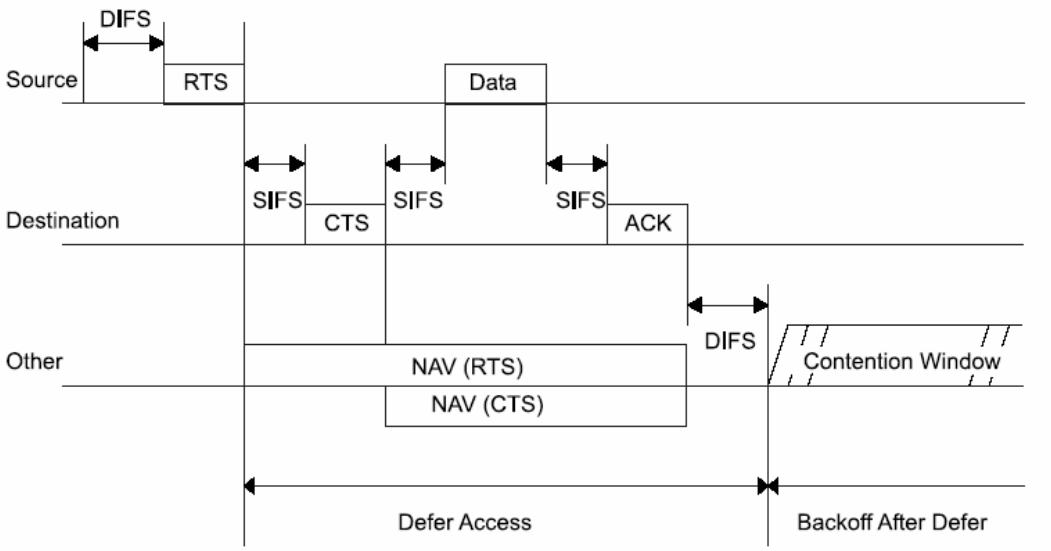

Figure 1 - RTS/CTS Illustration

It should also be noted that because of the fact the RTS and CTS are short frames, it also reduces the overhead of collisions, since these are recognized faster than it would be recognized if the whole packet was to be transmitted. This is true if the packet is

significantly bigger that the RTS, so the WLAN standard allows for short packets to be transmitted without the RTS/CTS transaction, and this is controlled per station by a parameter (called RTSThreshold). The use of RTS/CTS and NAV is illustrated in **Figure 1**.

C. Hidden Terminal Problem

In wireless ad hoc networks, hidden terminal problem occurs when a sender cannot detect the carrier of another sender on the other side of the receiver [73][76]. As illustrated in **Figure 2**, assume Node S1 is currently sending packets to node R. When Node S2 wants to send packets to Node R, it senses the carriers from its neighboring nodes. Because S1 is outside node S2's communication range, S2 cannot detect the existence of S1. Therefore, if S2 sends some packet when S1 is transmitting to node R, collisions will happen at Node R.

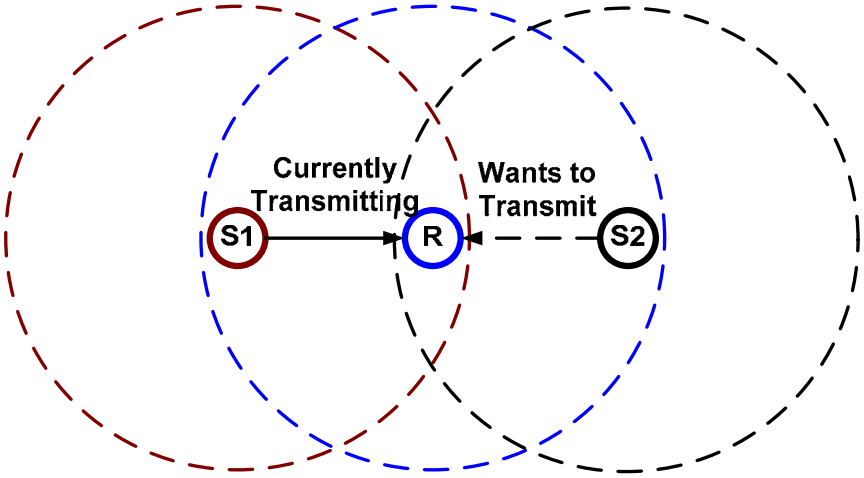

Figure 2 - Illustration of Hidden Terminal Problem

WLAN standard uses RTS/CTS mechanism and handshake packets to partly overcome the hidden terminal problem. However, RTS/CTS is not a complete solution and may decrease throughput even further.

D. Exposed Terminal Problem

In wireless networks, the exposed terminal problem occurs when a node is prevented from sending packets to other nodes due to a neighboring transmitter. Consider an example of 4 nodes in **Figure 3**, where the two receivers, R1 and R2, are out of range of each other, yet the two transmitters, S1 and S2, in the middle are in range of each other. Here, if a transmission between S1 and R1 is taking place, node S2 is prevented from transmitting to R2 as it concludes after carrier sense that it will interfere with the transmission by its neighbor S1. However note that R2 could still receive the transmission of S2 without interference because it is out of range from S1.

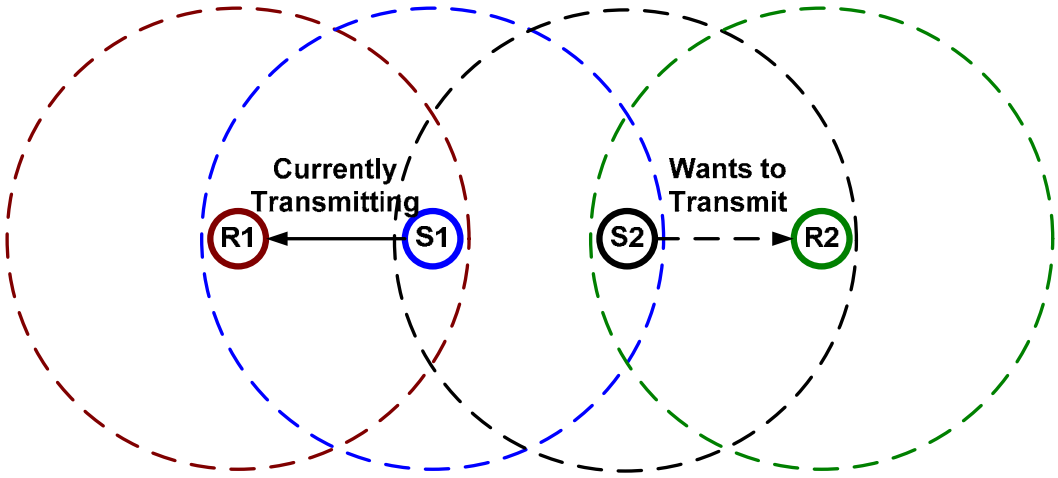

Figure 3 - Illustration of Exposed Terminal Problem

The RTS/CTS mechanism helps to solve this problem only if the nodes are synchronized. When a node hears an RTS from a neighboring node, but not the

corresponding CTS, that node can deduce that it is an exposed terminal and is permitted to transmit to other neighboring stations. If the nodes are not synchronized the problem may occur that the sender will not hear the CTS or the ACK during the transmission of data of the second sender.

1.1.1.3 The Network Layer

A. Networks without infrastructure

The wireless ad hoc networks are infrastructureless networks. Their operations do not rely on pre-deployed backbone nodes. In many cases, networks are formed on demand and are torn down when a specific task is completed. For example, students in a classroom can form a dynamic wireless ad hoc network when they are having a class. The network can then be torn down when the class is ended.

Because of this infrastructureless feature, each router node has to agree to forward packets for other nodes in wireless ad hoc networks. In other words, nodes in mobile ad hoc networks are data end-points as well as routers.

B. Multihop Communications

Ad hoc networks typically consist of groups of wireless devices which communicate with each other in a peer-to-peer mode. Because of the limit of the RF transmission range of these mobile devices, data packets usually need to traverse multiple hops before reaching the destinations. The purpose of routing in wireless ad hoc networks is to find the best route according to the preset routing metrics, e.g. the smallest hop count, the best link quality or the lowest energy consumption.

C. Dynamic Network Topology

In wireless ad hoc networks, many nodes are mobile nodes, such as mobile PC or handheld devices. The mobility of these devices often leads to frequent changes of network topology. This chrematistics of the wireless ad hoc network generates the following impacts at network layer.

- The wireless links and routes are instable. Nodes have to frequently check its links to neighboring nodes. Once a link breakage is detected, a node has to repair or re-discover routes to some destinations. This will generate more control packets in the network and reduce the effective throughput of the network.
- Some traditional routing algorithms used in the wired network can not be used directly in wireless ad hoc network. For example, the link state based routing algorithms in general are not suitable because the network may suffer from excessive link state update packets due to the instable links.

These features bring new challenges for protocol and algorithm designs in wireless ad hoc networks.

1.1.2 Multicasting in Wireless Ad Hoc Networks

Multicasting refers to the ability of a communication network to accept a message from an application and to deliver copies of the message to multiple recipients in the network. When unicast routing is used, the source has to establish multiple connections to deliver the packets to all recipients. However, if multicast routing is used, the source needs to establish only one connection to the multicast tree, no matter the number of the recipients. As illustrated in **Figure 4**, (a) shows in order to deliver a data packet to 1,000

recipients, the source node needs to establish 1,000 unicast connections. This imposes heavy burden to the source and generates heavy traffic to the network. On the other hand, (b) shows the same task can be done by using a single multicast connection.

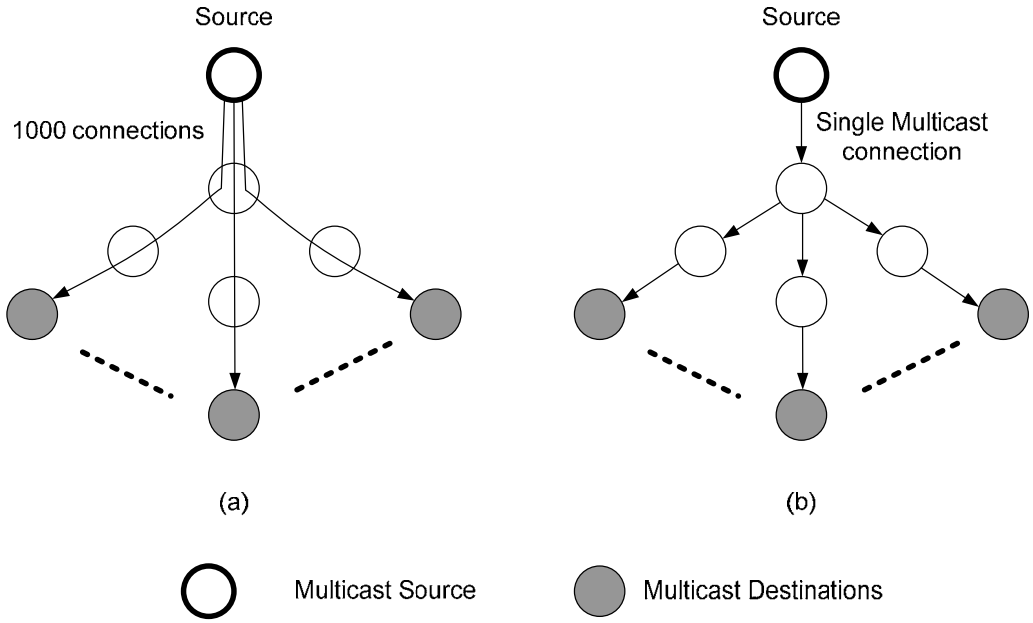

Figure 4 - An Illustration of Multicasting

The challenges of doing multicast in wireless ad hoc networks include

- routes often break due to node movement;
- control overhead must be emphasized;
- protocols for fixed network are not going to work well (trees are fragile, link states are too expensive).

Multicast routing protocols are thus responsible for maintaining and reconstructing the routes in a timely manner as well as establishing the durable routes. In addition, routing protocols are required to perform all the above tasks without generating excessive

control message overhead. Control packets must be utilized efficiently to deliver data packets, and be generated only when necessary.

1.1.3 Broadcasting in Wireless Ad Hoc Networks

Under many circumstances, broadcasting is the only way of service discovery, device paging, and even data transfer in wireless ad hoc networks. For example, most of reactive routing protocols discover routes by broadcasting route requests. However, broadcasting may lead to "Broadcast Storm" problems if it is done by simple flooding in a single common channel. The problems include redundant rebroadcast (overlaying), contention and collision. When the network density is high or the network is getting congested, the problems become severe and may directly lead to degraded quality of service. For example, the broadcast storm led by one node may prevent other nodes from accessing the channel or make other sessions in the network suffer from longer delay. The following two features differs broadcasting from unicasting in a wireless ad hoc network.

A. No Acknowledgement for Broadcast Packets

Unlike unicast packets whose transmission can be confirmed by acknowledgement packets, broadcast packets are not acknowledged at the MAC layer. Therefore, the sender has no knowledge on whether the packet has been delivered after sending it.

B. No RTS/CTS dialogue for Broadcast Packets

The RTS/CTS dialogue is able to deal with hidden terminal problem for unicast packets. However, in the broadcast case, exchanging RTS/CTS messages between the

sender and all receivers is impractical. With the absence of RTS/CTS dialogue, the chance that packets suffer from “hidden terminal problem” at receiver side increases.

1.2 Outline of the Thesis

This thesis consists of five chapters.

Chapter 1 is the general introduction of the thesis. It sets forth the background, motivation and contributions of this research work.

Chapter 1 presents the evaluation and enhancement of Multicast Ad-hoc On-demand Distance Vector routing protocol [1]. An analysis on its problems and protocol ambiguity is described in details. A modification and enhancement of the protocol is provided to solve the problems. Simulation model is developed to evaluate the protocol and the results are analyzed.

Chapter 1 presents our findings on route discovery latency of reactive routing protocols designed for wireless ad hoc networks. An RTT-based algorithm for determining the waiting time to collect route replies is developed to provide a solution that is simple, widely applicable and highly efficient.

Chapter 4 presents our work on hybrid multicast routing protocol for wireless ad hoc network. It is designed under the assumption that a logical tree structure and local link state information are available. The description of the protocol and its operation are presented in details.

Chapter 5 presents a smart broadcast scheme for wireless ad hoc networks. The broadcast storm problem in wireless ad hoc networks and existing solutions to the problem are first introduced. Our Smart approach is then described. The efficiency of the algorithm is investigated through simulation analysis.

Chapter 6 concludes the thesis.

2 EVALUATION AND ENHANCEMENT OF MULTICAST AD-HOC ON-DEMAND DISTANCE VECTOR ROUTING PROTOCOL

2.1 Introduction

With the fast prevalence of mobile computing devices, such as laptop computers and personal digital assistants (PDAs), the demand of connecting these devices become greater and greater. In many cases, people not only want to connect to the fixed infrastructure network (e.g. Internet), but also want to be connected with a small group (e.g. students in a classroom sharing files). In the later case, an infrastructure network is not needed or does not exist. Nodes are usually equipped with wireless communication devices and movable. The topology of the network changes dynamically. This is a typical Ad hoc network.

Multicast is an important application in ad hoc networks because usually an ad hoc network is formed when network hosts want to work together to carry out a specific task. Routing data packets to members within a group is much more challenging than to a single destination node. One reason is that traditional tree structures used in the fixed network is no longer suitable for ad hoc network simply because of the dynamic characteristic of the network. The other reason is that control messages used to exchange routing information among participating nodes often lead to excessive overhead which in turn may cause data collisions.

Despite the difficulty there are some multicast routing protocol have been proposed designated for ad hoc networks. These protocols are usually classified into two categories according to their delivery structures, tree and mesh. Among these protocols, MAODV[1], AMRIS[4], RBM[5], LAM[6] and AMRoute[7] are of tree structures while ODMRP[8], CAMP[9] and NSMP[10] are of mesh structures. Paper [10] compares the performance

of AMRoute, ODMRP, AMRIS, CAMP and flooding. The general result is that mesh protocols usually can achieve higher packet delivery ratio (especially when the speed is high) at the cost of higher control overhead.

However, as one of the most widely discussed multicast routing protocols for the ad hoc network, there is not a detailed, public performance study of MAODV. In this study, we give our evaluation on MAODV.

The rest of this chapter is organized as follows. Section 2.2 and 2.3 present the overview and operation of the protocol. In Section 2.4 we point out the ambiguities of the protocol descriptions and our modification. Our enhancements to the protocol are given in Section 2.5. Section 2.6 describes the simulation design and the environment of simulation. In Section 2.7 and 2.8, the simulation results are analyzed and discussed in details. Section 2.9 summarizes this chapter.

2.2 Multicast AODV Overview

Multicast AODV is an extension of unicast AODV protocol [2]. It provides dynamic, self-starting, multihop routing between mobile devices wishing to join and participate in a multicast group in an ad hoc network. MAODV classifies network devices in an ad hoc network into three different categories – group members, tree routers and pure unicast devices. Tree routers are themselves unicast devices but they are also on the multicast tree providing “bridges” for group members. A special group member call “group leader” is used to broadcast group hello messages periodically. The group hellos are introduced to ensure the freshness of the routing information for all the members. Non-members can send data packets to the group but will not receive.

MAODV can also handle network partitions and reconnections. When network partition happens, MAODV elects a new group leader for the partition without a group leader. When two partitions join, the group leader with larger IP address will become the only group leader for the joined group.

AODV and MAODV are tightly connected. MAODV uses the same Route Request (RREQ) and Route Reply (RREP) packets as AODV does but utilizes some fields reserved for multicast. Besides RREQ and RREP, MAODV also uses Group Hello (GRPH) and Multicast Activation (MACT) packet types. All these message types are handled by UDP, and normal IP header processing applies.

In addition to a unicast routing table, MAODV also maintains a multicast routing table and a group leader table. Multicast routing table contains information such as Group IP Address, Group Leader's IP Address, Group Sequence Number, Next Hops, Hop Count to Next Multicast Group Member and Hop Count to Group Leader. The Next Hops field is a linked list of structures, each of which contains Next Hop IP Address, Next Hop Interface, Link Direction and Activated Flag. Link Directions can be either UPSTREAM (towards the group leader) or DOWNSTREAM (away from the group leader). A node can only have one UPSTREAM.

2.3 Protocol Operation

A device wishes to join the multicast group to participate in the multicast session or has data to send to the group but found no active routes to the group initialize the route discovery procedure by broadcasting a RREQ packet. A timer call RREP_WAIT_TIME is started when the RREQ is sent. The RREQ packet contains the group IP address and

this device's last known group sequence number. A node which receives the RREQ checks its multicast routing table to see whether it has fresh enough route to the group or not. If it has, it replies to the RREQ by sending back a RREP packet to the RREQ source; if it does not have fresh enough route, it rebroadcasts this RREQ and keeps a record of the reverse route to the RREQ source. Note that if the Join flag of the RREQ is set, only a group member can reply; otherwise, any nodes with fresh enough route can reply. The previous hop of this RREQ is also added to the next hop list of the multicast route entry and marked DOWNSTREAM. The Activated Flag of this previous hop is unset.

When the RREP reaches an intermediate device on its way back to the RREQ source, the intermediate device must create a multicast next hop entry for the device from which it received this RREP and set the direction of this next hop to UPSTREAM. At this time the Activated Flag is still left to unset. After increasing the hop count by 1 and updating the lifetime field in the unicast entry of the previous hop, the intermediate node forwards the RREP along the path to the RREQ source.

It is possible that multiple RREPs are received by an intermediate device. In this case, the first RREP will be forwarded to the RREQ source and the rests will be forwarded only if they are better than the previous RREPs – their group sequence numbers are greater or their hop counts are smaller than the previous ones. The rest RREPs will be discarded if they are worse than the current one.

When the first RREP travels back to the RREQ source, the RREQ source will not take any actions. Instead, it will wait for more RREPs and hope there are better ones among them. A RREP better than the currently recorded one will replace the current one

so that when the timer RREP_WAIT_TIME expires, the RREQ source has the best route among all the possible routes it obtains. At this time, the source of RREQ activates the route by sending a MACT to the RREP sender along the route the RREP packet is received. When devices on the route receive this MACT, they activate this route by setting the Activated Flag. When the MACT reaches the sender of the RREP, the route is completely activated and a branch of the multicast tree is added. All other neighbors not receiving this MACT time out and delete the device as a next hop for the multicast group.

Other operations, such as a group member leaving the group, repairing a broken link, handling tree partitions and reconnections, will be omitted here. Readers with interest can refer to [1] for details.

2.4 The Ambiguity of the Protocol

To start our study and evaluation on MAODV, we implemented the protocol using the simulation tool OPNET. However, while we are implementing the protocol, we found the following ambiguities of the MAODV specification.

2.4.1 Multicast Routing Table

MAODV is an on-demand routing protocol. This means a device maintains its route to the multicast group only when the route is needed and active. The route should be deleted if it hasn't been used for a pre-defined time period, e.g. ACTIVE_ROUTE_TIMEOUT. However, the authors didn't mention in the draft how and when the route entries should be deleted. How and where to record the expiration time of the route is not mentioned either. In addition, each multicast address entry in the routing table may have multiple next hops in its next hop list. Should each next hop have

its own expiration time (because they are created at different time) or the entry maintain an expiration time for all of its next hops?

The answers to the above questions are so important to the protocol that we have to contact the authors for them. The authors explained that the next hops in the multicast route table do not have lifetimes. Instead, the lifetimes from the unicast route table are used. As long as the link to the next hop remains valid, that next hop is maintained in the multicast route table. The only time a next hop is removed from the multicast route table is if the unicast route table indicates that the link is gone, or if a Prune-MACT is received. Our implementation follows this description.

2.4.2 Unicasting RREQ to the group leader

In Section 9.2 of [1], the draft says if the node knows the group leader and has a route to it, the node MAY place the group leader's address in the Multicast Group Leader extension, and unicast the RREQ to the corresponding next hop for that destination.

However, the behavior of the device which receives this unicast RREQ is not defined in the draft. When a device receives a unicast RREQ, it should look at its unicast routing table for the next hop to the group leader and route this RREQ via that next hop. This process continues until the RREQ reaches the group leader. But how about an intermediate node can not find the route to the group leader to route the RREQ? Many reasons can make this happen in a wireless ad hoc network. The possible solutions may be, although not described in the draft,

1) to queue the RREQ and start a route discovery process to the group leader from the intermediate node. Once the route to the group leader is obtained, route the RREQ to it;

2) to simply discard the RREQ if the route to the group leader is not available since the source device will broadcasts subsequent RREQs for that multicast group cross the network if a RREP is not received within RREP_WAIT_TIME milliseconds.

We think neither way is good; the first one will finally start a broadcast for a unicast address just like broadcasting a RREQ for the multicast group and has less chance to find a better route to the group than the later (due to the fact the link to the group leader from the intermediate node may not exist any more); the second one increases the route discovery time.

In addition, one more question exists; since this unicast RREQ is destined to the group leader, can a group member or a tree router on its path to the group leader response to it, or only the group leader is allowed to response to it? Allowing the intermediate nodes to response to the unicast RREQ will obviously expedite the process of route discovery but may violate the authors' original purpose of this method.

Because there is not satisfactory solutions can be found to solve the said problems, although we implement the function using the second method, we don't really use it. Every time a route to the multicast group is needed, a broadcast RREQ is sent.

2.4.3 Creating or updating unicast routing table when receiving GRPH

In section 9.10 of [1], the specification says when the group leader in one partition (GL1) receives a GRPH from the group leader in the other partition (GL2), GL1 unicasts a RREQ with both the 'J' and 'R' flags set to the group leader of the other network partition (GL2), using the node from which it received the GRPH as the next hop.

This means that when a node receives a GRPH, it should create or update its unicast routing table for the reverse route to the group leader. But there is no place in the draft which address this. The problem is when creating or updating the unicast routing table, how to handle the source sequence number and destination sequence number. They are not contained in the GRPH packet and could mix up with the sequence numbers associated with RREQs and RREPs.

We add a new field call "*Source Sequence Number*" to the packet format of GRPH so that whenever a GRPH is received, the receiving node knows whether it should update its unicast routing table back to the group leader.

2.5 Enhancement of the Protocol

The MAODV protocol has several problems that affects its performance or even makes it fail to work. We enhance the protocol in the following aspects.

2.5.1 The algorithm of determining *route_discovery_timeout*

One of the key criteria of evaluating on-demand routing protocols is the route acquisition latency. The route acquisition latency is defined as the duration between the time a node discovers its need for a route to some destination, and the time that it

acquires that route and can begin using it. According to the MAODV draft, no matter how long it takes the source of RREQ to get the routing information, it has to always wait for *route_discovery_timeout* seconds before selecting the next hop and unicasting a MACT. Therefore, the route acquisition latency for a multicast route will always be *route_discovery_timeout* seconds. However, the real time for route discovery could vary largely depending on the topology of the network. In order for every node to get a route to the multicast tree (if the route does exist), the source of RREQ must wait the longest possible *route_discovery_timeout*. This is a drawback of the protocol because most of the nodes really do not need to wait so long. If the data rate is high and the buffer size is limited, drops of data packet from the buffer will be very possible. For real-time traffic, even there is not packet drop, the fixed *route_discovery_ timeout* increase the delay of the transmission.

To overcome this shortage, we developed a dynamic mechanism for determining the *route_discovery_timeout.* The mechanism works as follow.

Whenever a RREQ is sent, node records the time T_1 of sending it. The initial *route_discovery_timeout* is set using the algorithm provided by the protocol,

$$2 * TTL * NODE\_TRAVERSAL\_TIME \quad (1)$$

When the node receives the first RREP, it records the receiving time T_2. The time difference

$$T = T_2 - T_1 \quad (2)$$

is the RTT (round trip time) of the fastest RREQ/RREP pair. Now, instead of wait *route_discovery_timeout*, the node either waits only another T to send the MACT if

$$2T < route\_discovery\_timeout \quad (3)$$

or uses the original *route_discovery_timeout.* if

$$2T >= route\_discovery\_timeout \quad (4)$$

Our algorithm is based on the assumption that the earlier the route is discovered, the higher the possibility that the route is better (with fewer hop counts). Also, we give another T for the rests of RREPs to arrive. This ensures us that we will not miss the real good routes (if the RTT of for a RREQ/RREP pair for a route is larger than 2T, it could not be counted as a good route). The development of this algorithm is also port of this work and it is described in details in Chapter 1. Our study shows that in the most cases, the fastest RTT of a RREQ/RREP pair is only a very small portion of the fixed *route_discovery_timeout.*

2.5.2 The Acknowledgement of MACT

To solve the MACT loss problem, we introduce a new packet type MACT_ACK to the protocol. It has only two fields, Group IP Address and the Source of MACT. Whenever a MACT reaches a group member (the destination of a MACT packet), the group member replies with a MACT_ACK. The Source of MACT field will be set to the IP address of the very first sender of the MACT. This MACT_ACK will be unicast back to the sender of the MACT. Nodes receive the MACT_ACK know how to route this packet by looking up their unicast routing table.

In addition, when a node sends a MACT, it starts a timer for the MACT_ACK to arrive. Again, the RTT of the RREQ/RREP message is used here to determine the waiting time of MACT_ACK. The initial waiting time will be set to twice of the RTT. If the MACT_ACK is not received within this time, the node will send another MACT and the waiting time will be doubled. If, however, the nodes can not receive a MACT_ACK after sending MAX_MACT_RETRIES, the route will be considered invalid and a new route discovery procedure will be started. The reasons and benefits of adding a new MACT_ACK packet will be discussed later in the simulation result section.

2.5.3 The Processing of GRPH

According to the protocol, the actions a member node will take after receiving a GRPH are updating the Group Leader Table and Multicast Routing Table and determining whether two partitions are joining by checking the group leader's IP address. In addition to the above functions, we added the action of refreshing all the active next hops (reset their route expiration timer) when a member node receives a GRPH message. We believe this is necessary in order to maintain the multicast tree. Detailed analysis will be provided in the later sections.

2.6 The OPNET Simulation Model

We studied and evaluated MAODV using the simulation tool OPNET. We implemented the MAODV protocol using OPNET Version 8.0. We have modeled all the functions defined by the draft except the following

- group leader leaving the group (usually won't happen);
- actions after reboot (won't affect the testing of a routing mechanism).

The main purpose of our implementation is to test the routing algorithm and we believe that the above-mentioned features will not affect the result of our tests.

2.6.1 Node Model

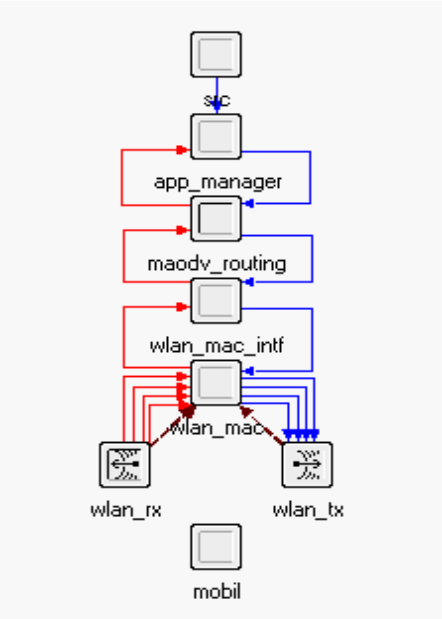

Figure 5 - Model of a Wireless Mobile Node

As shown in **Figure 5**, the node model has the following modules:

src – a data source module which is designed to be able to send data packets at certain rate and at random time according to the selected distributions

app_manager – sending different data packets, unicast or multicast, according to the simulation attributes; destroying data packets received from other nodes and doing some statistics, e.g. end-to-end delay

maodv_routing – handling all the routing tasks

wlan_mac_intf and **wlan_mac** – modules provided by OPNET which model the IEEE 802.11b Wireless LAN standard and its interface to the routing layer.

wlan_rx and **wlan_tx –** wireless transmitter and receiver provided by OPNET

mobil – modeling a waypoint moving pattern

2.6.2 Process Model

The only process model will be described here is the maodv_routing process.

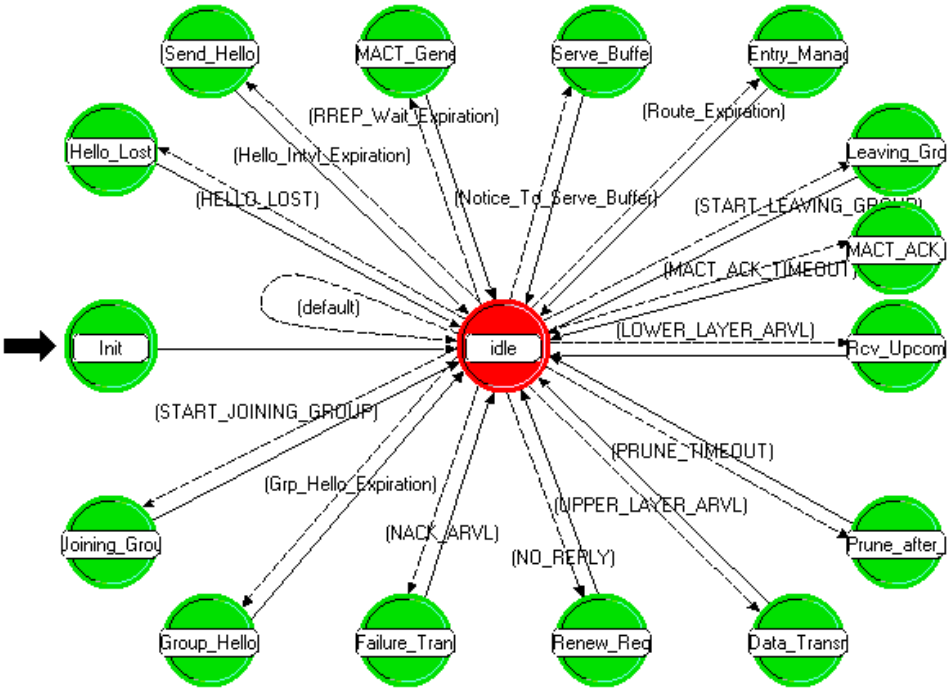

Figure 6 - Model of maodv_routing Process

Init state: In this state, state valuables are initialized; necessary model and simulation attributes are obtained; the template of a GRPH message is created.

Send_Hello state: nodes go to this state every hello interval to send a hello message when the hello mode is enabled.

Hello_Lost state: In this state, since the maximal number of allowed hello message losses has been reached, the node considers the link to the destination is broken. The node attempts to repair the link. In the case of multicast, should try to repair the multicast tree.

Joining_Group state: At the scheduled time (specified by node attribute "Time to join the multicast group" or a random distribution), the node generates a RREQ packet with the J flag set in order to join the multicast group.

Group_Hello state: the group leader broadcasts a Group Hello Message every group hello interval.

Rcv_Upcoming_Strm state: This state receives the incoming packet stream from the lower layer. It first checks the type of received packet then calls the appropriate function to proceed.

MACT_Generating state: When the *route_discovery_timeout* event happens, the node selects the route it wishes to use by sending a MACT message.

MACT_ACK_Timeout state: when the timer of waiting MACT_ACK timeouts, nodes resend the MACT. If the MAX_MACT_RETRIES is exceeded, nodes should re-discover the route.

Data_Transmission state: This state receives incoming packet stream from upper layer. It extracts the packet, and calls appropriate function to route the packet depending on the types of the packets (unicast/multicast).

Serve_Buffer state: This state dequeues a data packet from the buffer and sends it to the MAC layer. Interrupts that lead to this state are creation of a new entry following a discovery step and the successful reparation of a broken link.

Entry_Management state: This state is in charge of routing table management, i.e. invalidation of active entries following an expiration and deletion of invalid entries.

Leaving_Group state: Nodes can leave the group but some restrictions apply. A leaf node can leave the group without any restriction; a tree router can not leave but can revoke its membership; the group leader can also leave, BUT not implemented currently.

Prune_after_link_break state: If the loss of a link causes a node to become a leaf node, it sets a prune timer to wait for the link to be repaired. If, when this timer expires, the node has not received a MACT message selecting it to be a part of the repaired tree branch; it prunes itself from the tree by sending a MACT with set 'P' flag to its next hop.

Renew_Request state: This state is entered when the RREP_WAIT_TIME expires. If the TTL of RREQ has not reached the NET_DIAMETER, the new RREQ is sent with increased TTL so that the expanding ring search is achieved. If the NET_DIAMETER has been reached, the node then use the NET_DIAMETER as the TTL and retries the RREQ the maximal retry times. If the node still can not get a route reply, it then becomes the group leader of this group or the group leader of its own partition and starts sending GRPHs.

Failure_Transmit state: This state is called when a failure to occurs at the MAC Layer. In this case, the MAC Layer notifies the upper layer by sending a NACK message

containing both final and next hop destinations of the lost data packet. At this point, the following operations are executed:

- Node invalidates the next hop entry;
- Node perform local repair on the broken link.

2.7 Simulation Design and Environment

In order to make our simulation results comparable to the authors' results, we make our simulation environment as close to the author's as possible. This environment is described as follows.

The simulated network contains 50 nodes. These nodes are initially placed randomly within a fixed-size L x L area. When the simulation starts, each node randomly picks a destination within the area and moves toward it at the predefined speed. When it reaches the destination, it then chooses a rest period from a uniform distribution between 60 and 300 seconds. After the rest period, the node travels toward another randomly picked destination. This process repeats throughout the simulation.

The communication radius of the nodes is set to 10 meters. Any packet received from a neighbor that is not further than 10 meters from the receiving node is considered acceptable; otherwise, even though the packet is physically received, it will be discarded without any processing.

The MAC layer model is the OPNET implementation of IEEE 802.11b WLAN model with some minor modifications. The speed of the Wireless LAN is set to 1Mbits/sec.

In this study, we only examine the multicast performance of the protocol. Therefore, there is only multicast traffic in the network. All simulations were run for 300 seconds.

The following simulation design differs slightly from that of the authors.

Nodes join the group within a short time period (0~60 second, we call it *joining period* hereafter) after the simulation starts. This simulates the scenario that after a multicast session is announced, nodes join the group to participate in the session. Nodes can choose to generate data packets to the session after the joining period or they can just join to listen. At the time nodes join the group, they also schedule a time randomly to leave the group.

Table 1 - Simulated Parameter Values

Parameter Name	Value
allowed_hello_lost	2
group_hello_interval	5000 msec
Hello_interval	3000 msec
max_retrans	2
rev_route_life	9 sec
route_expiration	10 sec
rreq_retries	2
rte_discovery_timeouts	dynamic
max_mact_retries	3

Data packets are sent by both group members and non-members to the group at randomly selected time throughout the simulation. Data packets are 64 bytes in length

with constant bit rate and the number of data packets transmitted per session is a geometric distribution with average 3,000.

Essential parameters of the simulation are given in **Table 1**.

2.8 Simulation Results and Analysis

In order to examine the performance of MAODV, [3] use the goodput ratio and the amount of control overhead as criteria. The goodput ratio is defined as the number of data packets received compared to the number of data packets sent. However, we found out that collecting the right number of received data packets is a very difficult task. The reasons are simple. First, we don't know at the time a data packet is sent, how many group members are supposed to receive the packet because of the dynamic membership; second, even though we know the number of group members in the network, we don't know how many partitions exist at that moment. Therefore, we still have no idea on the number of group members that should receive the packet. Members in the partitions other than that of the sender will not receive the packet and should not be considered as failure of receiving the packet. In short, to calculate the goodput of the protocol, we have to have both membership and topology information at every moment a data packet is sent.

As an alternative of using the goodput ratio, we use an easier way to roughly estimate the packet delivery ratio of the protocol. This method is based on the assumption that if a node is expected to receive a data packet, it should receive at least a portion of the data packet. Therefore, the lost data will be detected. We assign every data packet a sequence number when it is sent. The receiving members keep a record of the IP address and the latest data sequence number of the received packet. Whenever a gap is found

between the sequence numbers, the difference will be counted as the number of data lost. Again, this method is not accurate. If the lost data packets are at the beginning or the end of the session, they won't be detected. In addition, a node's failure to detect the loss of data packets does not mean it should not receive it. Due to the reasons mentioned above, the goodput ratio will not be presented and discussed in this work. In the rest of this section, will discuss the control overheads and the problems we found about the protocol.

We test the protocol in two different scenarios. In the first scenario, 50 nodes are placed in a room of size 50m x 50m. As we stated before, the transmission radius is 10m. Therefore a room of this size provides high enough node density so that almost every node can reach all other nodes in one or multiple hops. Simulations show that there are few, if any, partitions of the multicast tree throughout the whole simulation process. The main purpose of this scenario is to examine whether MAODV is able to construct a multicast tree and maintain the tree till the end of the multicast session.

In the second scenario, the room size is increased to 85m x 85m. All other environment parameters remain the same. In a room of this size, node density is much low than the previous one (the area is about 3 times of the previous one). This room size allows the nodes to be partitioned into several small groups so that the tree maintaining functions of the protocol, such as tree repairing, network partition and reconnection can be tested when the simulation progresses.

In order for our results to be comparable to the authors', we ran our simulations at six different speeds ranged from 0 m/s to 1 m/s. For each speed, 10 simulations with different seeds were run and the results were taken as the average of 10 different runs.

Considering that the highest speed the authors of [3] have tested is too low (1 m/s equals to 3.6 km/hr, a speed of ramble), we tested the speeds ranged from 0 m/s to 20 m/s (20 m/s equals to 72 km/hr, a speed of a moving vehicle).

Same as in [3], there is only one multicast group the nodes may choose to join. There is not unicast traffic in the multicast simulations. In [3], nodes generate new data sessions for the multicast group once they join the group. However, we reckon that in most applications, the majority of the nodes are receivers rather than senders. Therefore, in our simulations nodes may play the roles of pure "listeners" – joining the group without generating data packets. The limit of the number of group members is set to 10, which means at any given time, there are as many as 10 nodes are group members. However, a node can send data to the multicast group regardless of whether it is a group member. Non-member senders use the same route discovery procedures as the member senders do except they do not join the group and become members.

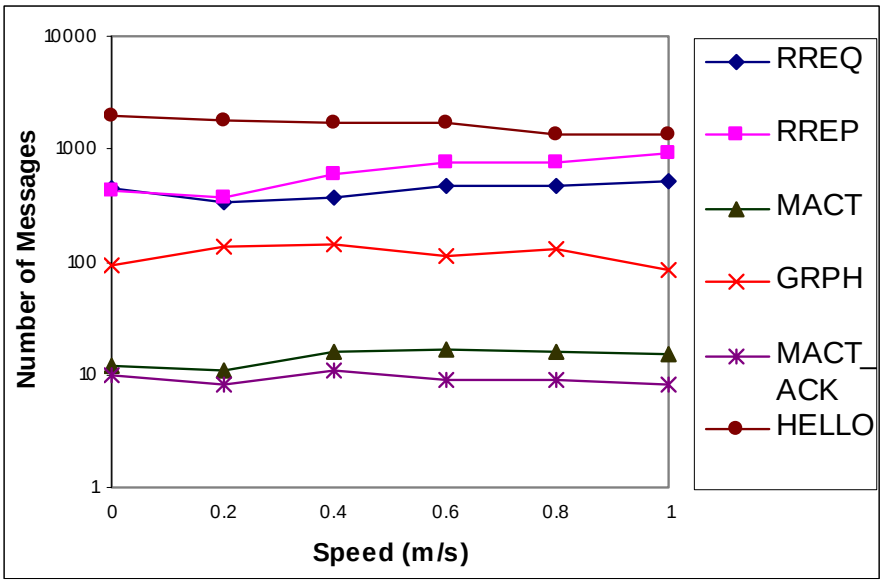

Figure 7 - Control Overhead for 50m x 50m Multicast Simulations (Slow Speed)

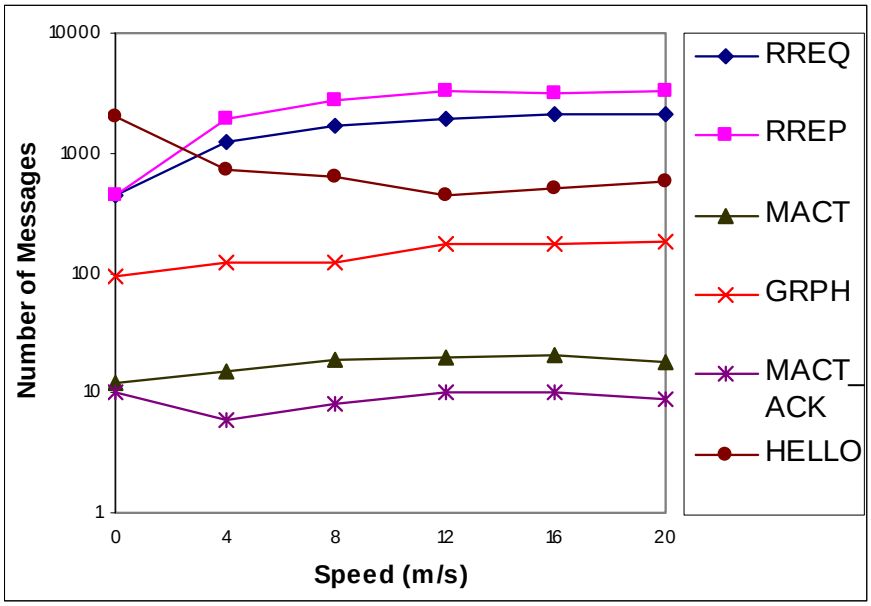

Figure 8 - Control Overhead for 50m x 50m Multicast Simulations (Fast Speed)

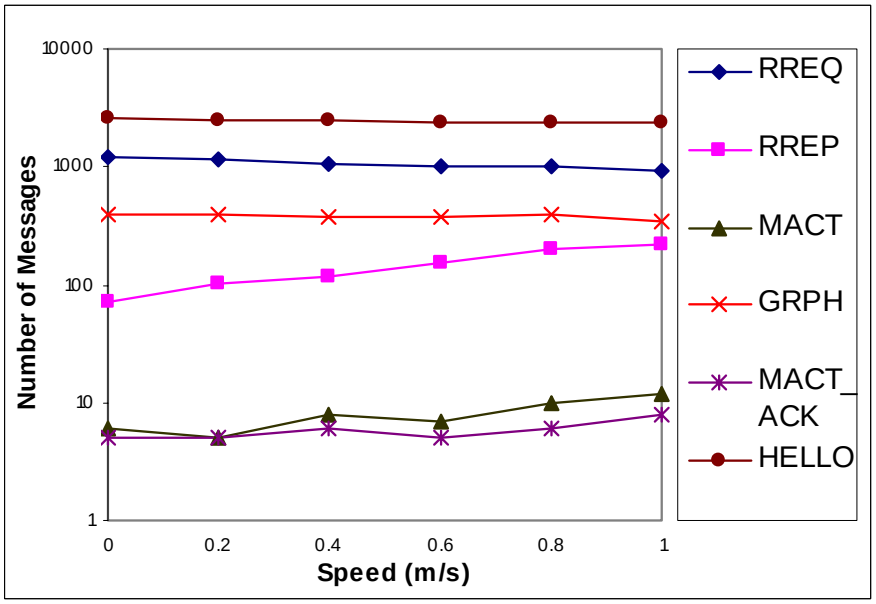

Figure 9 - Control Overhead for 85m x 85m Multicast Simulations (Slow Speed)

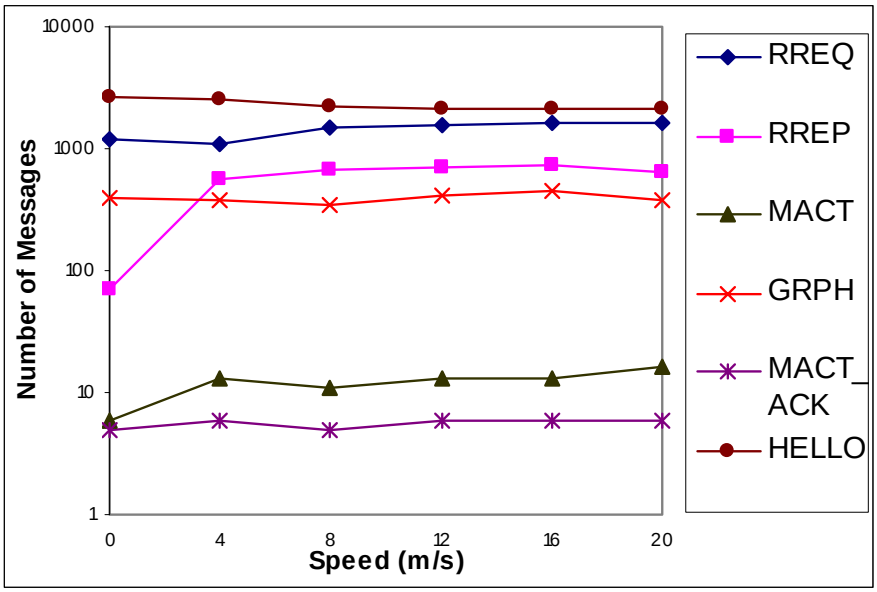

Figure 10 - Control Overhead for 85m x 85m Multicast Simulations (Fast Speed)

Figure 7 and **Figure 8** show the control overheads for the first scenario, 50m x 50m. The results of the second scenario, 85m x 85m, are shown in **Figure 9** and **Figure 10**. The obvious differences the readers might notice between our results and the results in [3] are the overheads of MACT_ACK and Hello packets.

As we stated at Section 2.5.2 of this chapter, in order to guarantee the route activation process to be successful, we add packet type MACT_ACK. Since MACT_ACK is a unicast packet and is sent only when a MACT is received, the number of MACT_ACK packets is only a very small portion of other control messages. The figures also show that the number of MACT_ACKs is less than the number of MACTs. There are two reasons for this result. One is that not all MACTs can get MACT_ACKs. Some MACTs were lost either due to the collisions or due to the expected receiver moved outside the communication range. This is the exact reason why we introduce this new packet type. The detail analysis why MACT_ACK packets are necessary is given later in this paper. The other reason is that not all MACTs require acknowledgements.

For example, when a MACT is broadcasted, no MACT_ACK is expected and no MACT_ACK_TIMEOUT is scheduled. Please refer to section 9.8 of [1] for the information of broadcasting a MACT.

In paper [3], hello messages are not counted as control overheads. However, we believe it should be considered as one of the overheads. In the scenario like the 50m x 50m room, the node density is so high that each node can have around 10 neighbors. In this case, even though the hello messages propagate only one hop, they could lead to a high possibility of collisions in its neighborhood.

In the unicast AODV protocol, the authors recommend several ways for maintaining a node's local connectivity. One of them is to utilize the link layer notifications; the others are methods of using different kinds of passive acknowledgements. Hello message is considered one of the passive acknowledgements. We use the hello messages as our way of detecting the lost of local connectivity because we do not like to modify the link layer model provided by OPNET. The readers might have already noticed that our number of RREQs and RREPs is dramatically larger than the corresponding number in [3]. The one who leads to this difference is Hello message. In our simulation, a node' hello function can either be turned on or off. Once it is turned on, it will select a random time within the first second of the simulation and start sending Hello messages till the end of the simulation. Since we depend on the Hello messages to detect the link break, this function was turned on for all 50 nodes. This means each node keeps track of Hello messages from all of its direct neighbors. Whenever a Hello lost is found, the node immediately sends out a link repair RREQ if the link is repairable. If a node detects the

broken link is to a DOWNSTREAM node of the multicast tree, a tree repair RREQ is also sent. Considering there are 50 nodes in the network, each of them have several direct neighbors, and each link break will lead to 2 link repair processes and maybe 1 multicast tree repair process, it is not difficult to understand why the number of RREQs and RREPs are so big in our simulations.

In fact, the approach specified in [3] is used to reduce the number of Hello messages. At the sender's end, if a node transmits any packet between the time the last hello was sent and the next scheduled hello sending time, the node will reset its timer for sending next Hello message to *hello_interval*. At the receiver end, if any packet is received, the timer of generating a hello lost event will be reset to ALLOWED_HELLO_LOSS * HELLO_INTERVAL.

There is an interesting phenomenon we found about the hello messages in our simulation. There are many nodes that are neither group members, tree routers, nor non-member senders. However, they keep on exchanging Hello messages and repairing the lost links once the link breaks are detected no matter these links are in use or not. This is a real waste of network bandwidth and the processing power of nodes. It is also the potential factor of collisions. Therefore, we suggest a node do not send Hello message unless it is "used" – it is a sender, receiver or a router on a unicast route; it is a group member or a tree router of a multicast group or a non-member sender. We believe this will further reduce the number of Hello messages although this idea has not been tested by simulations.

The rest of our results are similar to the results published in [3]. In the first scenario, the vast majority of the nodes are connected in one partition so the number of Group Hello messages is low. Simulations show there are only one or two group leaders in this case. When the area increases in the second scenario, the number of Group Hello messages also increases to about 3 or 4 times of the number of the first scenario.

Comparing **Figure 7** and **Figure 8**, we found that the number of RREQs and RREPs increase greatly when the nodes move faster. This is due to the fact that faster movement leads to more link breaks. Nodes were busy on repairing the links and trees no matter the links are used or not. Because of the frequent transmissions of other control messages, the number of Hello messages goes down when the speed goes up. This is well illustrated by **Figure 8**. The figures also show that the speed has less impact on control overheads to a sparse network than to a dense network.

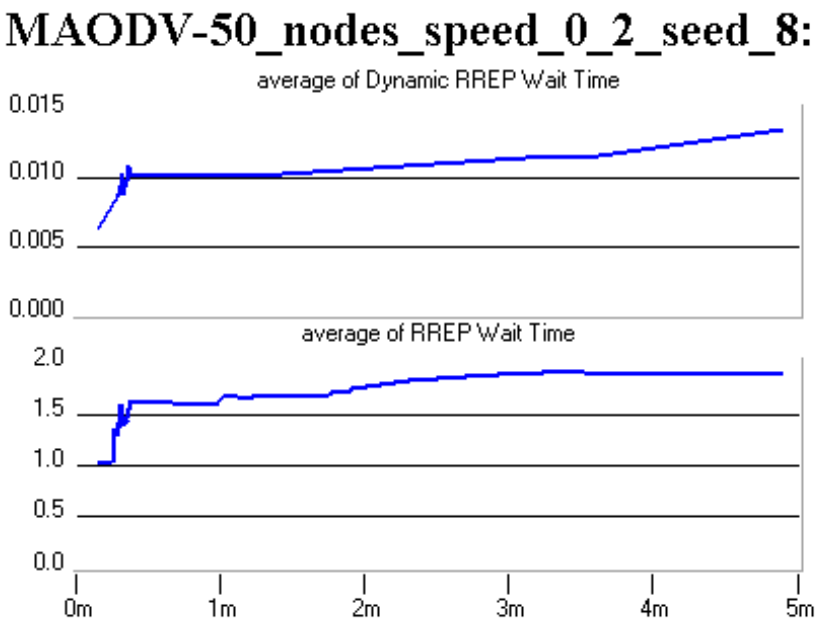

Figure 11 - *route_discovery_timeout* - Dynamic VS Fixed

It is clearly stated in [3] that MAODV use a fixed *route_discovery_timeout* value, 1000 msec, to wait for the RREPs and finally activate the route to the multicast tree. As we described in section 2.5.1, we have developed a dynamic algorithm for calculating the accurate value of *route_discovery_timeout.* **Figure 11** shows the comparison of our algorithm vs. the original method. The averages of two different waiting times show that our algorithm shortens the route discovery latency 138 times in this simulation scenario. As a matter of fact, our algorithm can always reduce the waiting time tens to hundreds of times of the original one. Of more importance, our waiting time is accurate because it is calculated based on the measurement of the RTT of RREQ and RREP packets while the original one is based on some sort of estimation.

2.9 Problems of MAODV and Our Solutions

Our simulation results have revealed problems of MAODV. The major ones are discussed as follows.

First, MAODV is not able to maintain a shared multicast tree.

MAODV is an on-demand protocol. There is not question that MAODV can construct a multicast tree. However, after the construction of the tree, can it keep the tree members connected until the session ends or the nodes explicitly leave the group?

In order to answer this question, we first look at how the multicast routing table entries are refreshed in MAODV because the tree is maintained via maintaining the routing table. After a node (say Node B) joins a multicast group, its upstream next hop (say Node A) to the tree in the multicast routing table will be assigned a certain amount of time (said ACTIVE_ROUTE_TIMEOUT) during which the route will be considered active. Node A also has Node B in its next hop list indicating that Node B is one of its downstream next hops and will be considered active for certain amount of time. This route expiration time is actually maintained in the unicast routing table. When the session starts, every time a packet is transmitted to or received from a next hop, the expiration time of the next hop will be refreshed to ACTIVE_ROUTE_TIMEOUT.

The multicast tree will remain connected as long as group members have something to communicate with each other. However, if there is a time period in the session during which no group members are sending anything and this idle time is longer than ACTIVE_ROUTE_TIMEOUT, problem happens; the next hops to the tree in every

node's multicast routing table will all expire and the next hop entries will all be deleted from the routing table. That means the multicast tree does not exist any more. Although the value of ACTIVE_ROUTE_TIMEOUT is adjustable, it should not be very large because the route can be outdated due to the movement of the nodes. Therefore, this situation is very likely to happen. When a node has something to send, the situation is not too bad because it can start a new route discovery process. The sender can find the route again. But its receivers will be all gone, suppose most of them don't have to send anything (which is a very common case for the applications like teleconferencing). These receivers will not start a new route discovery process until they become senders.

We found this problem in our simulations and we believe this is a severe problem of the protocol. Since no one has the knowledge about the group membership in the network, the sender might think that it can reach all of its members even when some of the members are no longer connected to the tree. In version 9 of the unicast AODV draft, the authors claimed that "*AODV allows mobile nodes to obtain routes quickly for new destinations, and does not require nodes to maintain routes to destinations that are not in active communication*." In other words, it also says AODV is not capable of maintaining routes that are not in active communication. This is not a problem in the case of unicast, but a lethal problem for multicast application.

Although the authors of [3] claimed that "*AODV is able to find a route to the multicast group each time it is needed, and it is able to successfully maintain the links of the multicast tree for the lifetime of the group*", our simulations show the opposite side of it. We guess the traffic pattern used in [3] leads to above statement. In [3], *"once a node*

is a member of the multicast group, it generates new sessions for that multicast group…" In this case, there seems to be endless data transmissions among group members hence continuous refreshing of each next hops in the routing table. However, this is not the only application.

A possible solution of this problem is to let every member know that the session has not finished yet and it should remain connected to the tree. This is the reason we also refresh all the active next hops when a member node receives a GRPH message. By doing this, every next hop can be refreshed at least once every GROUP_HELLO_INTERVAL. This interval is usually smaller than the ACTIVE_ROUTE_TIMEOUT value so that the multicast tree is maintained.

Second, MAODV is not robust.

Scenario 1: Loss of MACT

In MAODV, a node wishes to join a group or find a route to the group first broadcasts RREQs and then waits for RREPs. After a RREP_WAIT_TIME period, the node picks the best route from a set of RREPs and sends a route activation message MACT to the node from which it received the RREP. This will finally activate the route to the multicast tree.

However, this MACT is likely to be lost simply due to the lossy characteristic of the wireless network and the frequent topology changes. If the MACT is lost in the middle, the sender of MACT has no knowledge about it and will assume the route is now available. If the sender of MACT is going to transmit anything to the group, the situation

is not too bad. The nodes which receive packet from this node will start their own route discovery process to route this packets to all other members; but if the sender of MACT is just going to join the group to "listen to" other members, then it will never join the group.

If nodes send data packets right after they send out MACT (one of the two reasons why nodes send RREQs and then MACT after receiving RREPs), the possibility of losing the MACT will become higher. This is because the MACTs must compete with the data packets for the channel if hidden terminals exist. This is illustrated in **Figure 12**. Collision happens when Node 2 tries to forward MACT to Node 3 while Node 1 is trying to send data to Node 2.

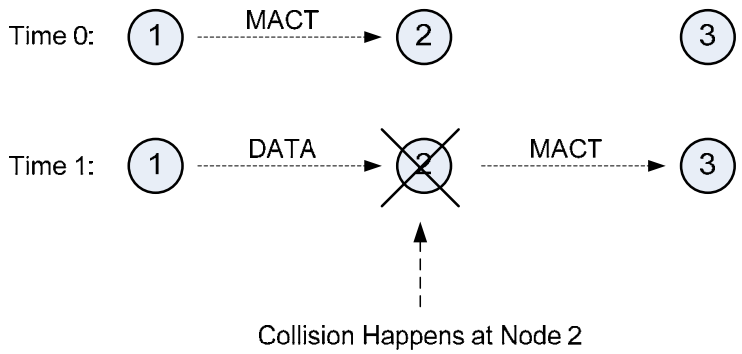

Figure 12 – Illustration of Collision between Data Packet and MACT

The possibility of losing MACT will be very high if a node which is sending data has many packets waiting in its buffer. This is very likely to be true since the sending node buffers all the data packets during the route discovery period.

Our solution to this problem is to send an acknowledgement of the MACT, MACT_ACK, back to the sender. This will solve the problem and does not cost much.

Also, after sending MACT, wait for a random time before sending buffered data packets if MACT_ACK is not used. This will reduce the chance of collision between the MACT and data packets.

Scenario 2: Loss of hello messages

The protocol allows only the downstream nodes repair the broken tree. This will cause problems under some circumstances.

Some times after a link break is detected, the corresponding next hop must be deleted from the next hop list. If the detecting node of the break is an upstream node, it is not responsible for repairing the tree. Therefore, the tree will be repaired after the downstream node also detects the break. The unicast protocol is responsible for the link repair. If the link is repairable and finally repaired by the unicast protocol, how can the unicast protocol tell the multicast protocol that the next hop is repaired and now can be used to route packets? Current protocol doesn't address the issue. According to protocol, the only way for the multicast tree to be repaired is to wait until the downstream node starts the tree repair process.

Let's consider this scenario – an upstream node A didn't receive the expected hello from one of its downstream node B. But this is not the result of link break or network partition, but simply the result of collision. When the hello expiration time arrives, node A has to delete node B from its next hop list. The unicast part of the protocol will try to repair the link but the multicast part will do nothing because node A is an upstream node. Node A will wait until Node B detects the link break and starts the tree repair process.

But remember, the link is not really break! The hello lost was caused by the collision! Therefore, node B will never detect the link break – it can receive hello from node A. The final result is the tree will never be repaired if node B have nothing to send (it has already join the group and will never join again).

One of the possible solutions to this problem is to allow more hello message loss or use a longer hello interval. However, this will cost more time to detect the link breakage. This is not a good way especially for the multicast application. A link break happened in the middle of the tree would cause packet loss and delay among all child nodes of the break point.

Another possible solution is to keep the next hop in the list but mark its status to UNDER_REPAIR. After the unicast protocol repairs the link, restore its status to ACTIVE. If the link is not repairable, then delete the next hop.

Scenario 3: quick topology changes

Our simulation result show that fast network topology changes may lead to failures of some functionalities of MAODV. As an example, let us look at the following scenario.

When a group leader GL_1 of one partition in the network receives a GRPH from a group leader GL_2 of another partition, it realizes these two partitions are merging. Since its IP address is smaller then GL_2's IP address, GL_1 sends a RREQ to try to reconnect the two partitions. When the RREP from the GL_2 travels back to the GL_1, it hits an intermediate node in GL_1's partition. This intermediate node is supposed to forward the RREP through its upstream next hop to its previous group leader GL_1. However, it just

cannot find the upstream next hop in its next hop list – the next hop has been deleted due to the break of link. At this moment, the protocol does not know what to do but wait for the next reconnection attempts.

2.10 Summary

In this chapter, we have examined the performance of multicast AODV. We pointed out some ambiguity points of the protocol and explained how we implemented these parts in our OPNET model. We also enhance some parts of the protocol in order to make it work well. We found that MAODV is not going to work if there is an idle time during the multicast session and we fixed the problem without making big modification to the protocol. A dynamic method of calculating *route_discovery_timeout* is introduced and proved by the simulations to be about 100 times better than the original one. A new packet type MACT_ACK and the mechanism of MACT_ACK_TIMEOUT is used to solve the problem of MACT lost. Our simulations also show that MAODV has some unsolved problems when the control messages are lost or links are broken.

3 ROUTE DISCOVERY LATENCY OF WIRELESS AD HOC NETWORKS

3.1 Introduction

As explained in Chapter 1, in order for a data source to find a route to the destination in a wireless ad hoc network, some sort of routing mechanisms are needed. So far, there are bunches of routing protocols have been proposed to fulfill this necessity. These routing protocols generally fall into two categories, proactive protocols and reactive protocols. Typically, proactive routing protocols discover and maintain the routes to all destinations by periodically exchanging neighbor or link state information among participating nodes. Reactive protocols, on the other hand, discover and maintain the routes on-demand, which means the protocol discovers the route to a specific destination only when this route is requested. The source node maintains this route until the current communication session is done. Once the route becomes idle for a predefined route expiration time, it will be deleted from the routing table and the route no longer exists. Since routes are not maintained when they are not in use, if later, after been deleted, these routes are requested again, nodes have to restart the route discovery process to find the routes.

One of the advantages (probably the biggest) of proactive protocols is that whenever a route is requested at a source node, it is immediately available. However, as a result of periodical information exchange, proactive protocols consume lots of network resources (e.g. bandwidth and power), generate heavy control overheads and may lead to packet collisions (if operate over a shared channel like IEEE 802.11). In contrast to proactive protocols, reactive protocols avoid these drawbacks by finding the routes only when they are needed. By doing this, periodical message exchange is not necessary, hence saves the

network resources and reduces the control overhead. However, reactive protocols have their own problems. The biggest problem is their long route discovery latency. When a data packet arrives at the routing layer of a source node running reactive protocol and the route is not currently available, the source node has to first buffer the data packet and, at the same time, start finding the route to the destination. Depending on the distance between the source node and the destination node, this route discovery process may take different amount of times. If the two communication parties are far away from each other or there are too many collisions along the route (nodes then have to backoff before retransmitting at the MAC layer), this process will take quite some time. This route discovery period may be large enough to cause the continuously coming data packet to be dropped from the source node's limited sending buffer. This is very likely when the data rate is high and the nodes are kind of compact devices like PDAs which have limited amount of memory.

In this chapter, we investigate the reasons of route discovery latency of reactive routing protocols and propose our RTT-based dynamic algorithm which can dramatically reduce this latency. The rest of this chapter is organized as follow. Section 3.2 gives the detailed analysis of the problem. Section 3.3 describes our dynamic algorithm. The application of the algorithm is illustrated in Section 3.4. Simulation design and result analysis are provided in Section 3.5 and 3.6. Finally, Section 3.7 concludes this chapter.

3.2 The route discovery latency problem

Although belonging to the same reactive category, different reactive protocols use different mechanisms to select the best route from multiple available routes. Generally,

there are two kinds of route selection mechanisms. We refer the first kind as "source selection". In this case, the initiator of the route discovery selects the best route from multiple route replies. Protocols belong to this category include AODV [2], TORA [13] etc. The second kind, we call it "destination selection". In this case, the destination of the route request selects the best route based on some predefined criteria and then sends the route reply back to the source along the route just found. Here the destination of the route request could be either the final destination or some times the intermediate nodes with fresh enough route to the final destinations. We refer both of them as "destination" hereafter. ABR [14][15] and SSR [16] fall into this category.

Let us now go through a typical route discovery process and find out the cause of route discovery latency. A source running a reactive protocol broadcasts a route request when a route to the destination is needed. After sending out this route request, the source starts a timer, T_{wait}, to wait for the route replies. Intermediate nodes relay the route requests until either the Time-To-Live limit is met or the request reaches a destination.

When the request reaches the destination, there are two situations. If the destination selection mechanism is used, the destination (usually the final destination, not an intermediate node which has the route to the final destination in this case) has the following two options. It can simply take the first arrived route request, send a route reply back to the source, and ignore all route requests arrived later than the first one; or it can wait for another period of time to collect more requests (usually travel via different routes) and then select the best among them based on its selecting criteria. If the source selection

mechanism is used, the destination shall simply response with a route reply for each route request it receives.

When the route reply (or replies) traverses back to the source, the source also has the following two options. If the destination selection scheme is used, or if this is the first route reply and the source believes the first arrived route reply is always the best one, the source will then start sending data packets waiting in its buffer using this route to the destination without any further delay. However, if the source selection mechanism is used, the source has to wait for other possible route replies to arrive until its T_{wait} timer expires. When the timer expires, the source selects the best route from all received replies, puts only best route into its routing table and starts routing the data packets.

In short, the route discovery latency of a reactive routing protocol consists of two parts. The first part is the physical packet traversal time it takes for a route request to reach the destination plus the time it takes for a route reply to travel back to the source. This part of the latency is beyond the control of any routing protocols and hence we call it “hard latency”. The second part of the latency is the time it takes for the route selecting party to wait for more route replies (when it is source selection) or more route requests (when it is destination selection) to arrive so that it can pick the best route among all available routes. Remember that this latency starts when the first request or the first reply has been received. At this moment, the source already has at least one route reply (source selection) at hand to use or the destination already has at least one route to reply. Therefore, we call it “soft latency” because this part of the latency is controllable and variable. The main purpose of this latency is two fold; first, to make T_{wait} big enough to

guarantee there will be at lease one route reply to arrive; second, to select the best route among available routes. Surprisingly, no existing reactive routing protocols even try to reduce this latency.

So far, we have gone through the general process of a route discovery process of reactive routing protocols. We now give the formula of describing the total route discovery latency $T_{wait.}$

$$T_{wait} = (T_{request} + T_{reply}) + T_{soft} \quad (5)$$

where $T_{request}$ is the time it takes for the first request to traverse from the source to the destination; T_{reply} is the time it takes for the first reply (or the only route reply, when the protocol is destination selection based) to traverse from the destination back to the source; T_{soft} is the extra waiting time after receiving the first request or reply, namely, the soft latency. For source selection protocols, T_{soft} happens at the source side and after T_{reply}; for the destination selection protocols, T_{soft} happens at the destination side and before T_{reply}. In addition, note the hard latency, T_{hard}, is the sum of $T_{request}$ and T_{reply}.

Before introducing our algorithm, we like to review how current reactive routing protocols deal with this problem. More specifically, we would like to see how they determine the value of T_{wait}. The simplest way is using a fixed T_{wait} just like what DSR [11] does. Obviously, a fixed timer is too simple to suit various network conditions. If the source and the destination are very close to each other, T_{wait} may be much larger than the necessary response time. If the source and the destination are far away from each other, then T_{wait} may not be large enough to collect all possible replies safely. Therefore, using a

fixed T_{wait} is either a waste or unsafe and the protocols using fixed T_{wait} are not scalable. Protocols like AODV [2] use a more advanced mechanism than fixed scheme. Since an expanding ring search approach is used to limit the flooding effect in AODV [2], T_{wait} in this protocol is associated with the TTL of each route request attempt. That is, when the TTL of the route request increases, the T_{wait} also increases proportionally. However, this mechanism, although makes the T_{wait} more adaptive, still cannot reduce the T_{soft} for each route request-and-reply cycle. In addition, our simulation study shows that the packet traversal time in a contention-based network does not always increase as hop count increases. Increasing waiting time for larger TTL does not make sense in this situation.

The purpose of our proposal is to reduce T_{soft} to the greatest extent yet can still guarantee the protocol will not miss the best route. We believe the hard latency, T_{hard}, provides very important information that should not be neglected. The T_{soft} must be a function of T_{hard} to avoid unnecessary latency because T_{hard} measures the real distance (in time) between the source and the destination.

3.3 The RTT-Based Algorithm

The following describes our RTT-based algorithm of determining the necessary actual route reply waiting time. In this study, only the distance in hop count is considered as the route selection criterion. Therefore, when we say one route is better than another, we are saying one route has less hop count than another to the destination.

3.3.1 The Algorithm

When a source node initiates a route request, it starts a timer called $T_{wait\_scheduled}$ to wait for the route replies. The value of this initial timer is estimated based on some sort of

adaptive algorithm like the one used in AODV [2] (which will be described later). Please note that this $T_{wait\_scheduled}$ is the T_{soft} we described in (5). Another thing to note is that $T_{wait\_scheduled}$ must have a "safe" value initially because we do not want to miss the route replies if they are under the coverage of the TTL of the route request. At the same time, the source records the sending time of this route request, T_1.

Upon receiving the first route reply, the source node again records the receiving time T_2. The time difference between T_2 and T_1 is then the *RTT* (Round Trip Time) of the fastest route request/reply pair. Please note that this *RTT* is what we called T_{hard} which is not under our control.

$$RTT = T_2 - T_1 \tag{6}$$

Now that we have the first route reply at hand, we would like to compare its *RTT* with the previously scheduled timer $T_{wait\_scheduled}$. A big difference between them would reveal that our initial estimation is not accurate; hence, the actual waiting time should be adjusted to adapt to the network situation. We propose the following algorithm to recalculate the actual waiting time.

$$T_{wait\_actual} = (1 + \alpha) RTT \tag{7}$$

where α is a constant between 1 and 3. Our recommended value for α is 2. The selection of α is discussed later in this section.

Now we compare T_{wait_actual} with $T_{wait\_scheduled}$:

If T_{wait_actual} is smaller than $T_{wait\_scheduled}$, then instead of waiting until pre-scheduled timer $T_{wait\_scheduled}$ to expire, the source node waits only T_{wait_rest} from this moment, which

$$T_{wait\_rest} = T_{wait\_actual} - RTT = \alpha * RTT \quad (8)$$

In addition, if an RREP with smaller hop count is received during this extra waiting period, the source node will take the route reported by the later RREP even without waiting T_{wait_rest} to expire. The reason will be discussed later in this section.

If, however, T_{wait_actual} is greater than $T_{wait\_scheduled}$, then the original $T_{wait\_scheduled}$ will be used and T_{wait_actual} will be disregarded.

As a summary, after receiving the first route reply, the source node calculates the actual time it is going to wait for more replies using formula (7), and compare the result with the pre-scheduled waiting time. It will then chooses to use the smaller one of these two and wait another T_{wait_rest} period given by (9) from this moment.

$$T_{wait\_rest} = \alpha * RTT \quad (when\ T_{wait\_actual} < T_{wait\_scheduled})$$

$$= T_{wait\_scheduled} - RTT \ (when\ T_{wait\_actual} >= T_{wait\_scheduled}) , \text{ where } 1 < \alpha <= 3 \quad (9)$$

3.3.2 Determining the value of α

Our algorithm is based on the assumption that the earlier the route is discovered, the higher the possibility that the route is better (with fewer hop counts). In addition, to make sure we will not miss any real better routes, we give another $\alpha * RTT$ period for the rest of route replies to arrive. Our simulation study shows that about 90% of the time, the first route reply will be the best reply. If a route better than the first received route does exist,

88% of the time it will arrive during the first RTT period and 7% of the time it will arrive during the second RTT period after the source receives the first route reply. In other words, only 5% of the time a route better than the first route reply will arrive after two times of RTT after receiving the first route reply. That is why we suggest setting α to 2.

Please note that the smaller the value, the more aggressive the algorithm (which means the source may lose some of the better routes); the greater the value, the safer the algorithm (more better routes may be received). The recommended range of α is from 1 to 3.

3.3.3 Why stop waiting after receiving one better route reply?

Our simulation study shows that after a source receives the first better-than-the-first RREP, it is safe to stop waiting for more possible "even better" ones because there simply will not be any "even better" routes. Detail simulation results and discussion is presented in the Section 3.6.

One advantage of our algorithm is its simplicity. Only local clock information is needed to calculate the dynamic waiting time. Very little processing power is needed from the source node and the destination node does not involve. Please note that in general our algorithm works for source selection protocols because the algorithm needs to measure the round trip time of the first route reply. If, however, a global synchronization system like GPS is used, it can also be applied to destination selection protocols.

3.4 Application of the algorithm

Although our algorithm can be applied to any reactive source-selection ad hoc routing protocols, we select Multicast Ad hoc On-demand Distance Vector routing protocol (MAODV) as our benchmark protocol. MAODV and its unicast brother AODV share the same mechanism of determining the route reply waiting time. Therefore, we use the word MAODV and AODV alternatively in this work.

In AODV, after sending out a route request (RREQ) packet, the source waits for RING_TRAVERSAL_TIME to collect all possible route replies (RREP). Specifically,

RING_TRAVERSAL_TIME = 2 * NODE_TRAVERSAL_TIME

*(TTL_VALUE + TIMEOUT_BUFFER) (10)

where NODE_TRAVERSAL_TIME is the time it takes for a packet be transmitted from one node to its direct neighbor. Its recommended value is 40 second. The TIMEOUT_BUFFER is configurable and is designed to provide a buffer for the timeout so that if the RREP is delayed due to congestion, a timeout is less likely to occur while the RREP is still en route back to the source. This parameter can be omitted by setting its value to zero while the recommended value is 2 second. Note here our T_{wait} equals to the RING_TRAVERSAL_TIME.

For a specific network, both NODE_TRAVERSAL_TIME and TIMEOUT_BUFFER are constant. Therefore, RING_TRAVERSAL_TIME will only adapt to TTL_VALUE. However, for each TTL value (that is, each request-reply cycle), the RING_TRAVERSAL_TIME is also a fixed value.

Upon receiving the broadcast RREQ, an intermediate node checks its routing table for the route to the destination. If it has the route, it then sends a route reply back to the source; if it does not have the route, it broadcasts the RREQ to its own neighbors. This process continues until either the RREQ reaches a node (an intermediate node or the destination) that has the route or the TTL limit is met.

When the timer of T_{wait} expires, the source checks whether it has found the route. If it has, it will select the best route based on the arrival time of the RREPs and hop count to the destination, and then route the data packets to the destination. If it has not, it will increase the TTL_VALUE by TTL_INCREMENT and starts another route-searching attempt using the new TTL value. This expanding ring search process continues until the NET_DIAMETER is reached.

As a conclusion, AODV uses the source selection mechanism and an adaptive approach to calculate the reply waiting time for different TTL values.

3.5 Simulation design and environment

We implement our algorithm using OPNET version 8.0. The simulated network contains 50 nodes. These nodes are initially placed randomly within a fixed-size L x L area. The communication radius of the nodes is set to 100 meters. Any packet received from a neighbor which is not further than 100 meters from the receiving node is considered acceptable; otherwise, even though the packet is physically received, it will be discarded without any processing.

A random way point mobility model is used. When the simulation starts, each node randomly picks a destination within the area and moves toward it at the predefined speed. When it reaches the destination, it then chooses a rest period from a uniform distribution between 60 and 300 seconds. After the rest period, the node travels toward another randomly picked destination. This process repeats throughout the simulation. Six moving speeds ranging from 0 m/s to 1m/s are simulated.

The MAC layer model is the OPNET implementation of IEEE 802.11b WLAN model with some minor modifications. The speed of the Wireless LAN is set to 1Mbits/sec.

In this study, we only examine the multicast performance of the protocol. Therefore, there is only multicast traffic in the network. All simulations were run for 300 seconds.

Nodes join the group within a short time period (0~60 second, we refer this as *joining period* hereafter) after the simulation starts. This simulates the scenario that after a multicast session is announced, nodes join the group to participate in the session. Nodes can choose to generate data packets to the session after the joining period or they can just join to listen. At the time nodes join the group, they also schedule a time randomly to leave the group.

Data packets are sent by both group members and non-members to the group at randomly selected time throughout the simulation. Data packets are 64 bytes in length with constant bit rate and the number of data packets transmitted per session is a geometric distribution with average value 3,000.

Table 2 - Simulated Parameter Values

Parameter Name	Value
allowed_hello_lost	2
group_hello_interval	5000 msec
Hello_interval	3000 msec
max_retrans	2
rev_route_life	9 sec
route_expiration	10 sec
rreq_retries	2
max_mact_retries	3
RING_TRAVERSAL_TIME	dynamic
NODE_TRAVERSAL_TIME	0.4 sec
TTL_INCREMENT	2 sec
TIMEOUT_BUFFER	0 sec

Essential parameters of the simulation are listed in **Table 2**. Here we would like to mention the value of parameter NODE_TRAVERSAL_TIME. The recommended value for this parameter in the latest version (12th) of AODV draft is 40 second. However, we found out that was too big and hence changed it to 0.4 second. Also, note we omit the parameter of TIMEOUT_BUFFER by setting it to zero.

We test our algorithm in two different scenarios. In the first scenario, 50 nodes are placed in an area of size 500m x 500m. As we stated before, the transmission radius is 100m. Therefore, an area of this size provides high enough node density so that almost

every node can reach all other nodes in one or multiple hops. Simulations show that there are few, if any, partitions of the multicast tree throughout the whole simulation process.

In the second scenario, the area is increased to 850m x 850m. All other environment parameters remain the same. In an area of this size, node density is much low than the previous one (the area is about 3 times of the previous one). This area size allows the nodes to be partitioned into several small groups.

For each scenario, 10 simulations were run with different seeds. The results shown below are the average values of 10 simulation runs for each moving speed and scenario.

3.6 Simulation result and analysis

In our study, simulations are run for two general purposes; first, to explore the delay characteristics of route replies and determine the key parameters like α; second, to examine the performance of our algorithm.

3.6.1 Delay characteristics and α

In order to determine the value of α, we initially set its value to 10. This means that after receiving the first RREP, the source waits 10 times of the round trip time of the first RREQ/RREP pair to see whether there will be better routes to arrive. By setting this extra waiting time to 10 times of RTT, we are safe to say that we will not lose any better routes if they exist. To describe the problem better, we chop the extra waiting time after the first RREP into ten RTT slots. In **Figure 13**, we plot the percentage distribution that better routes fall into the RTT slots. As it shows, 87.69% of the time the best routes arrive during the first RTT period after the source receives the first RREP. If we sum the first

two histograms, we will notice that 95.38% of the time the best routes will arrive within the first two RTTs after the first RREP is received. This is the reason we recommend the value of α as 2. Note the result of **Figure 13** is a summary of two node densities at six different moving speeds.

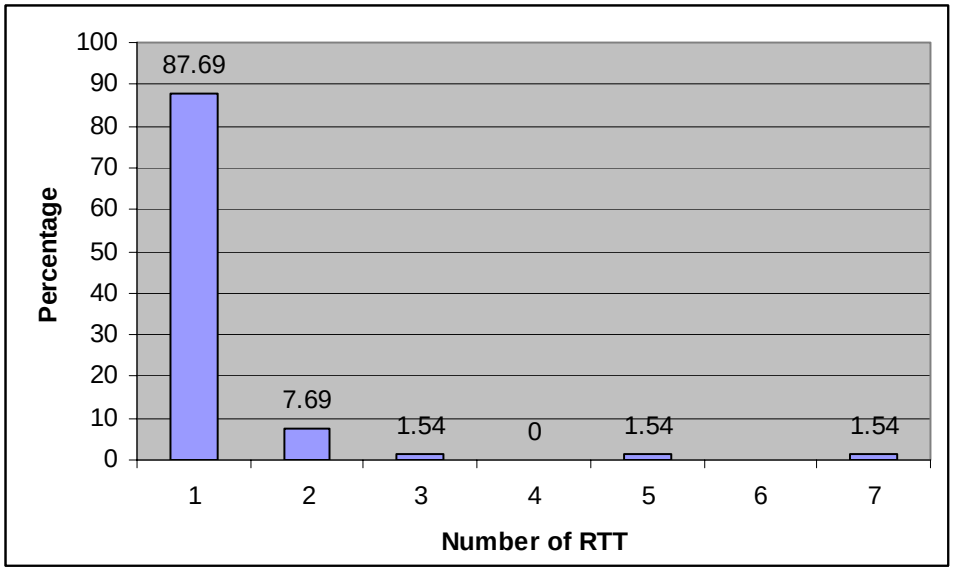

Figure 13 - The Percentage the Better RREPs Fall into Different RTT Slots

Because, normally, a source will receive multiple route replies after sending out a route request, it is also interesting to see the percentage of the RREP that is the best one – the first arrived RREP, the second arrived, and so on. To explore this, we count the number of the first RREPs. If no RREPs better than the first RREPs are received during the extra 10 RTT periods, the first RREPs will be considered as the best ones. If a better route is received right after the first RREP ("first better RREP" hereafter), the first RREP is then not the best one. At this time, the first better RREP just arrived will be considered the best one unless an even better route is received. Surprisingly, we observed that after

receiving the first better RREPs, there are not any more routes received are even better than the first better ones.

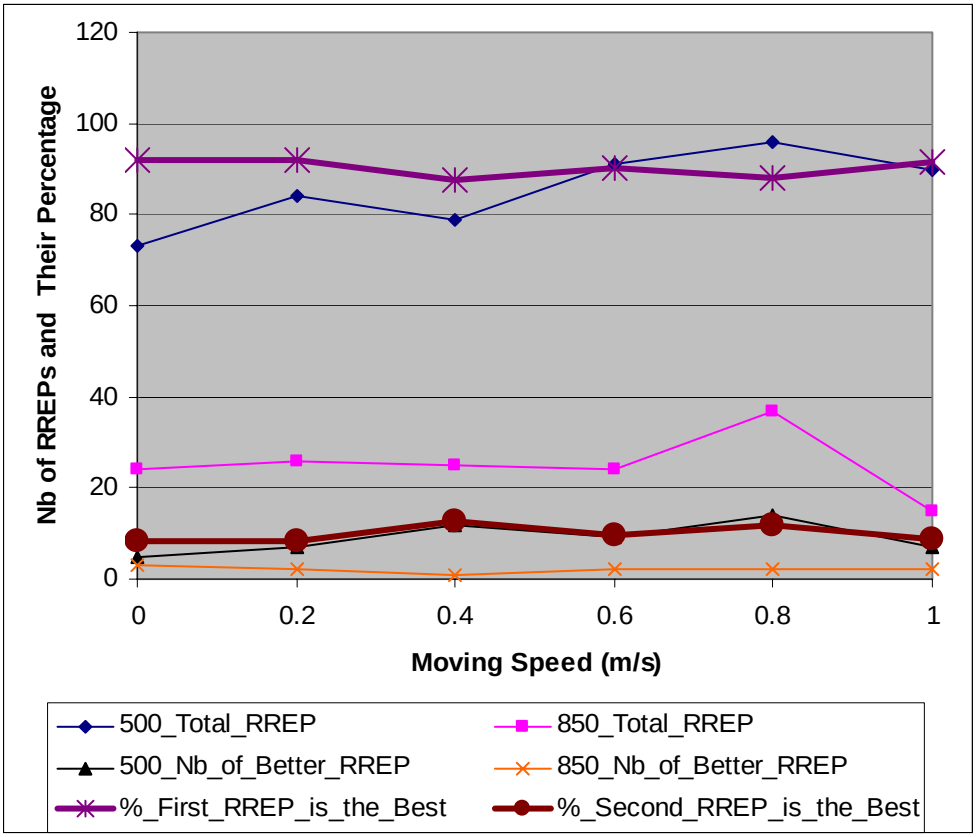

Figure 14 - Number of the First RREPs and the First Better RREPs and Their Percentages to be the Best Routes

Figure 14 plots the number of first RREPs and the first better RREPs for both dense and sparse network situations at six different speeds. The two thick lines represent the percentages that the first RREPs are the best ones (90.15% on average) and the first better RREPs are the best ones (9.85%) separately. As we mentioned above, these two percentages add up to 100% of the best routes. In other words, either the first RREP is the best or the first "better than the first" RREP is the best one. No other routes will be the best routes. Therefore, after a source receives the first "better than the first" RREP, it is

safe to stop waiting for more possible even better route replies because there will not be any of this kind based on our simulation result.

In **Figure 14** we can also see that even though all simulation parameters are kept the same except the network density, more RREPs are received in the dense scenario case than in the sparse scenario case. This is because in the sparse scenario, the network connectivity is not high enough for each request to get a route reply. In fact, the network nodes are typically separated into several partitions in this scenario. Most of the time during the simulation, there are many destination nodes are physically unreachable by a source node.

Although some protocols simply pick the first arrived RREP and believe it is the best route, **Figure 15** shows even though only about 10% later arrived RREPs are better the first ones, they do shorten the route (in hop counts) by fair high percentages. This indicates it is worthwhile to wait for the better routes in most of the cases.

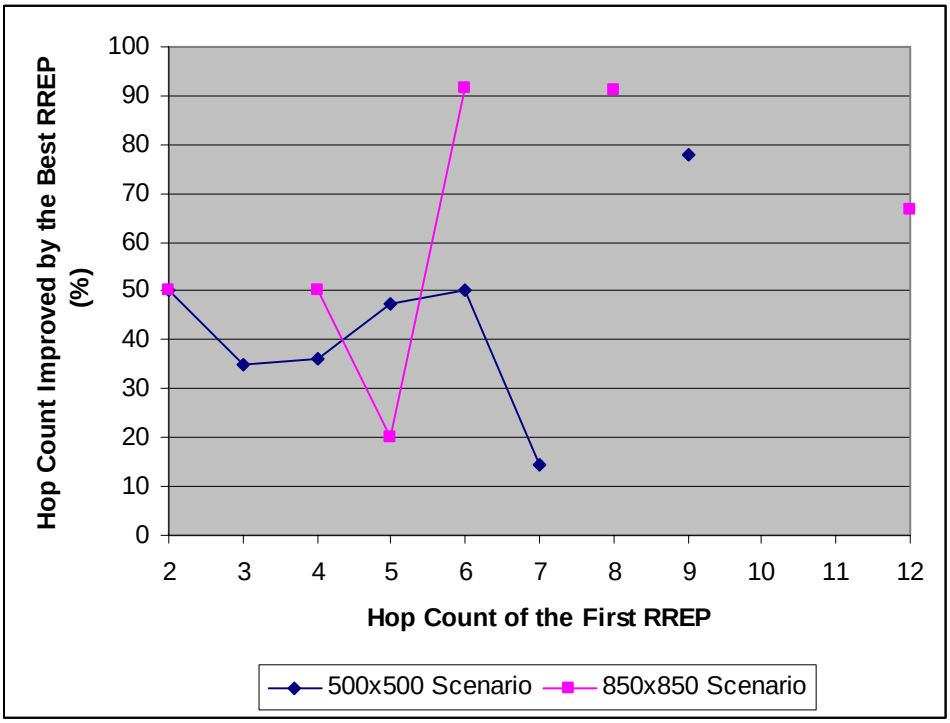

Figure 15 - Routes Improved by the Later Received Best RREPs over the First RREPs

3.6.2 Performance examination

In this sub-section, we are going to compare the performance of our RTT-based algorithm with that of AODV's algorithm. When comparing the results, bear in mind that we have already make the NODE_TRAVERSAL_TIME in AODV's algorithm 100 times smaller than its recommended value. This means that if the recommended value is used, AODV's waiting time in the graphs will be 100 times longer than what we will see.

We first compare the overall waiting time of AODV's algorithm and that of our RTT-based algorithm to see how much we can improve. The waiting times of different moving speeds for both scenarios are compared in **Figure 16**.

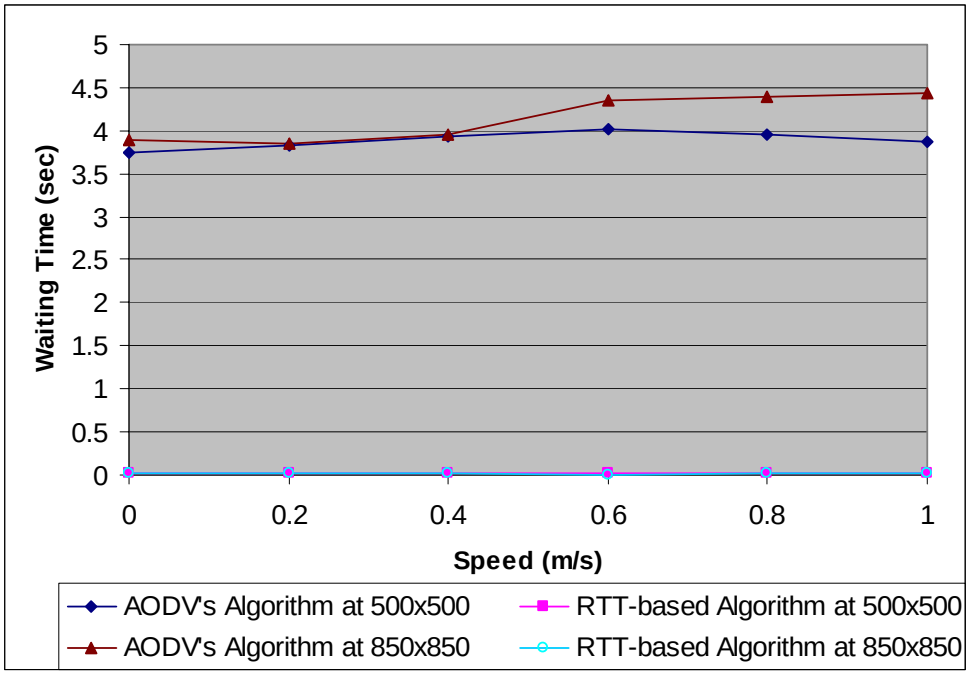

Figure 16 - Comparison of Waiting Times

In this graph, the waiting time of our RTT-based algorithm is too small to tell comparing with that of AODV's algorithm. Therefore, we plot the percentage our result is of AODV's in **Figure 17**.

From **Figure 17**, we can see our algorithm's waiting times are less than one percent of that of AODV's. In addition to the great difference in route discovery delay between these two algorithms, one other thing we can see from **Figure 16** is that, in general, moving speed has not or little impact on waiting time. For AODV's algorithm, it should be because no moving parameter is considered in (6). The reason the waiting times of AODV's algorithm at speed 0.6, 0.8 and 1.0 m/s of 850m x 850m scenario are larger than others are due to the fact that in a sparse network the hop counts between a source and a destination are usually larger than that in a dense network. This stems from the fact that a node with the same transmission range covers more nodes in a dense network than in a

sparse network. Therefore, to cover the same number of nodes, more hops are needed in a sparse network than in a dense network.

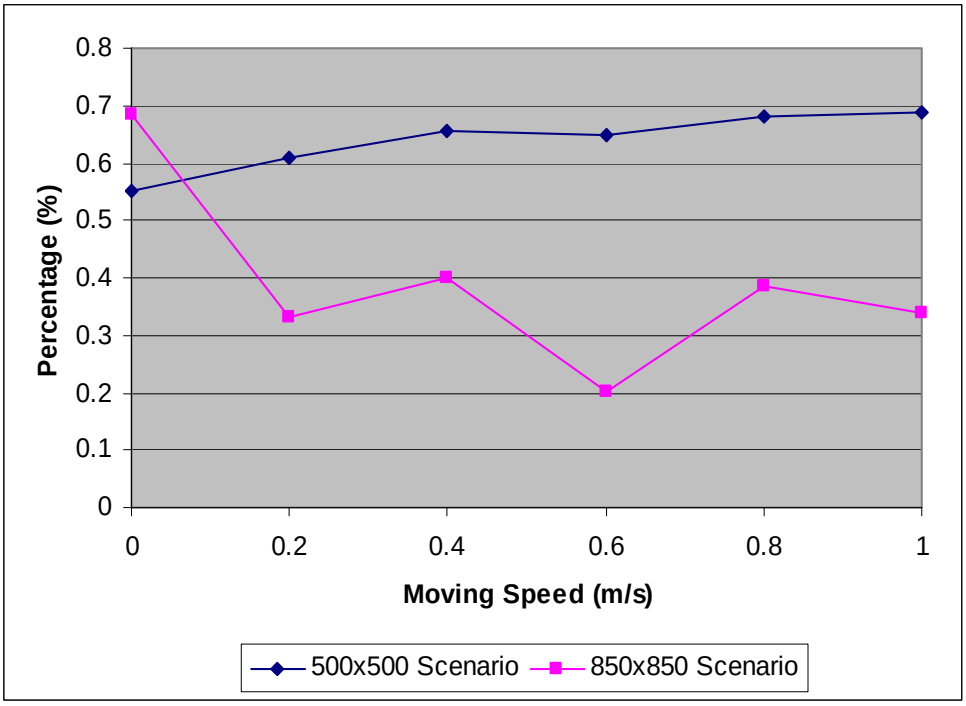

Figure 17 - Percentage Our Waiting Time is over That of AODV

Another result can be observed from **Figure 17** is that our algorithm improves more with the sparse network scenario. This is simply because AODV's algorithm is based on estimation depending on the distance (in hops) between the source and the destination while our algorithm is based on measurement. Therefore, our algorithm is able to adapt to different network densities.

For a detailed comparison, we would like to see how much our algorithm could improve for different distance (hops). This result is plotted in **Figure 18** using a logarithmic scale for the waiting time.

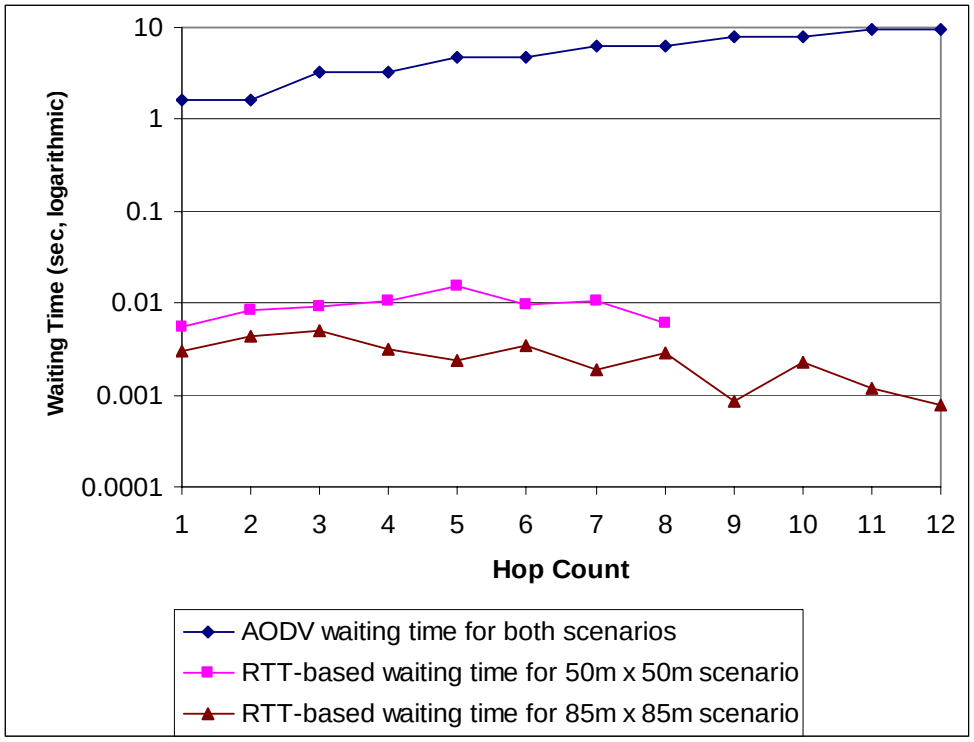

Figure 18 - Waiting Time Comparison for Different Node Distance

The first observation from **Figure 18** is that while the waiting time of AODV's algorithm increases with the distance between the source and destination nodes, our RTT-based waiting time does not. This is especially true in the case of a sparse network. In the dense scenario, we see that our waiting time fluctuates around 0.01 second with some variations after two hops. In the sparse scenario, the waiting time goes down with the increase of hop count beyond three hops. Two conclusions can be drawn from this observation. First, in general, the larger the hop counts are, the more waiting time our algorithm can save; second, the assumption that packet traverse time increases with distance is not true in a contention-based channel like IEEE 802.11, especially in a sparse network. Therefore, the algorithm AODV uses may not be meaningful. This result can be further explained in the second observation of this figure.

The second observation we have is that our RTT-based waiting time in the dense scenario is larger than that in the sparse scenario for the same hop count. This stems from the fact that the more dense the network is, the more contentions the flooding mechanism generates. When contentions are detected at the MAC layer, nodes have to backoff exponentially before re-transmitting the packets, either route requests or route replies. This leads to the longer delay in the dense network than that in the sparse network. These contentions happen when different nodes try to rebroadcast the route request they just received at a highly correlative time or when some nodes try to reply while others try to rebroadcast at the same time. In a sparse network, the chance of this kind of contention is less than that in a dense network, and hence packets may experience less traverse delay between the source and the destination. This observation also explains the first observation; when the destination is far away from the source (with large hop count), the effect of flooding is less severe so that the packets can travel faster. This phenomenon is referred as the "broadcast storm". Our work in [18] provides a highly efficient solution to this problem.

3.7 Summary

In this chapter, we have discussed the problem of route discovery latency in reactive ad hoc routing protocols. To avoid the "blind" estimation of the route discovery time, our algorithm measures the round trip time of the first route request/reply pair and use it as a rule to measure the quality of later received better routes. We showed by simulation that 95% of the time the better route replies would arrive during the first two RTT periods after the source node receives the first route reply. In addition, either the first route or the first "better than the first" route is the best route; it is not possible for a route other than

the first route and the first “better than the first” route to be the best route. Our algorithm waits up to three RTTs (if a better route is received) or less (if a better route is not received) to determine the best route to the destination. The simulation results also show that the primary reason for route discovery delay is the backoff time of the contention led by the flooding of route requests. This delay does not always grow as the distance between the source and the destination grows. By comparing with one of the reactive routing protocols, we show our algorithm is much better than the “blind estimation” algorithms and can adapt to different network densities.

4 A HYBRID MULTICAST ROUTING PROTOCOL FOR WIRELESS AD HOC NETWORKS

4.1 Introduction

In this chapter, we will introduce a new hybrid multicast routing protocol specifically designed for wireless sensor network, the most popular kind of wireless ad hoc network so far. The outcome of this work has been adopted by the IEEE standard of low rate WPAN mesh networks, numbered as IEEE 802.15.5.

The protocol is called “hybrid” because it is built on top of a hybrid unicast routing structure that combines a routing tree and local link state information. That says, it is neither a pure tree-based nor a pure link state based unicast routing protocol, but a combination of the two. To better describe our hybrid multicast routing protocol, we first introduce the hybrid unicast routing protocol of IEEE 802.15.5 LR-WPAN mesh network standard, which is the foundation of our protocol.

4.1.1 The Tree-based Routing

Because a wireless sensor network is usually a low-duty circle network, data traffic is expected to be very light in the network. Therefore the route optimality is less important than that in a traffic-heavy mesh network. On the other hand, because sensor devices are usually low cost devices, resources such as processing power and memory space are usually very limited. Therefore, big routing tables are not affordable in many cases.

Due to these two reasons, table-less addressing and routing scheme is often a good choice for wireless sensor networks. A table-less addressing and routing scheme usually needs to utilize some sort of logical tree to combine unicast routing with addressing. For

example, a binary tree can route data packets from one node on the tree to another without having a routing table; the destination address itself provides enough information for routing the packets. In a logical tree, there is a root device of the entire tree; a device is a child of its parent device (except the root device which has no parent device) and also the parent device of its own child devices (except the leaf devices that have no child devices). Devices get their short addresses assigned when associating with their parent devices. These short addresses can then be used for routing data packets. **Figure 19** shows an example of the logical tree. In the figure, device A is the root device. Device B has two children, device C and H, and has device A as its parent device.

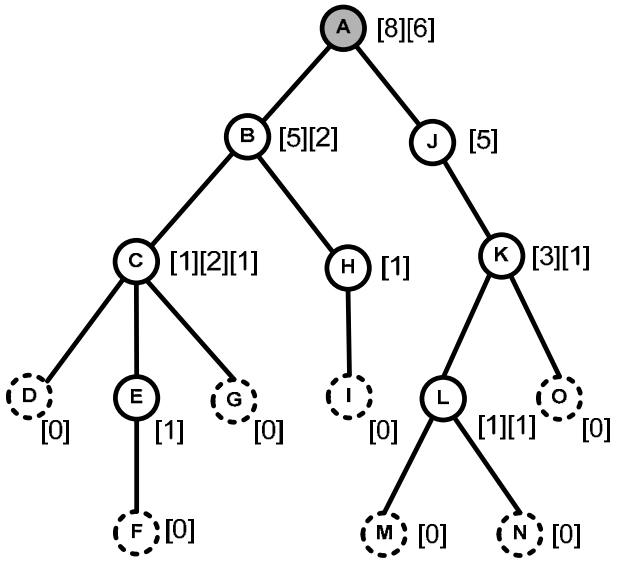

Figure 19 - Illustration of a Logical Tree and Address Assignment

Once the addresses are assigned to all devices, the basic tree routing algorithm becomes functional; no route discovery is needed. Given the destination address, a device searches its child branches to see whether the destination address falls into one of its child branches. If it does, the packet will be sent to the specific child branch; if it doesn't,

the packet will be sent to the parent device. This process repeats at each intermediate device until the packet reaches it destination.

4.1.2 The Local Link State Routing

Although this simple tree-based routing algorithm works, it suffers from the following problems. First, the routes are sub-optimal because packets can only be routed up and down the tree. Second, the routes are not robust because once an up- or down-link breaks, there is no other routes can be used, even these routes may exist. To solve these two problems, the LR-WPAN mesh standard combines tree routing with local link state routing.

To populate the local link state table, devices shall broadcast hello messages up to preset TTLOfHello hops (2 as default value) under the following circumstances.

- When the network is just formed (multiple hello transmissions required);
- When link status changes, such as new devices join the network or link breakage is detected.

In the hello messages, the entire address block of the sender's branch, instead of the sender's address, is carried. Also included in the hello messages are the addresses of the sender's one-hop neighbors. Therefore, link states of neighbor devices up to TTLOfHello+1 hops can be collected by exchanging local hello messages. With this link state information, devices can create connectivity matrixes of their neighbors up to TTLOfHello+1 hops. All these neighbors will be inserted into the neighbor list for routing purpose.

Now that the devices have the neighbor information up to TTLOfHello+1 hops, the routing algorithm is slightly changed; a device searches its neighbor table for the destination before using the tree routing. Depending on the distance between the two communication parties and the network topology, in many cases devices may find the destination is one of its neighbors. Within the coverage of the link state routing, optimal routes can be found. Routes are also more robust comparing to using the tree only; for instance, when the link to its parent is broken, a device can forward the packet to a neighbor with smaller tree level.

4.1.3 The Advantages of the Hybrid Multicast Routing Protocol

Comparing to the MAODV protocol we evaluated in Chapter 1 and Chapter 1, it has the following advantages.

- No need of network-wide broadcast control packets
- Much shorter joining delay
- No need for periodic network-wide hello messages
- No need for routing tables
- Lower power consumption
- More robust due to the local link state routing
- Much simpler algorithm, particularly good for wireless sensor networks
- No complex sequence numbers need to be maintained

The rest of this chapter is organized as follow. Section 4.2 gives the details of the protocol. Section 4.3 describes the operation of the protocol and Section 4.4 concludes the chapter.

4.2 The Protocol Design

In this section, we are going to design the hybrid multicast protocol for the wireless ad hoc network.

4.2.1 The Assumption and the Design Goal

As we discussed in Section 4.1, this hybrid multicast routing protocol relies on a pre-constructed logical tree that spans all nodes in the network and the availability of local link state information up to TTLOfHello hops (the distance, in the number of hops, the hello message will travel) for each node.

Our major goal of the protocol design is to utilize mesh links whenever they are available in order to construct an efficient and robust multicast tree with the least extra overhead (over the hybrid unicast routing protocol) for routing multicast packets.

4.2.2 Definitions and Terminologies

Group Member (GM)

A GM is a node that participates in a multicast group. A GM shall process all frames destinated to its group address and may send frames to it group. A node can be a GM for multiple groups.

Group Coordinator (GC)

The GC is the controller of a multicast group. Each group shall have its own GC and they can be different nodes. A GC is also a GM in most cases.

For many reasons, there is a need for a central controller (the GC) for each multicast group. Its functions may include membership management, group key management/distribution and the like.

For multicast routing, it is also very helpful if we can have GC's address. Optionally, when GC's address is known, join requests to a multicast group can be sent to the GC using unicast routing algorithm.

Mesh Coordinator (MC)

The MC is the root of the unicast routing tree of a wireless ad hoc network. It keeps information of all multicast groups in the network. For each multicast group, the group address, the unicast address of the Group Coordinator, and the number of members are recorded.

Tree Level (TL)

Tree level indicates the hop distance a node is from the root of the logical routing tree (i.e. the MC). The MC, has a TL equals to zero.

Multicast Router (MR)

A MR is a node on the multicast tree but not a GM. A multicast router relays multicast frames for a multicast group. A node can be routers for multiple multicast groups.

4.2.3 Neighbor List

To support multicast application, we need to include necessary multicast related information in the neighbor list created by the hello messages. For each neighbor in the neighbor list, we add a field called "Group Membership". This field indicates the multicast groups the node participates in. Note if a node is a member of many multicast groups, the Group Membership field needs to include all the multicast group addresses of which this node is a member. An example of the neighbor list is illustrated in **Table 3**.

Table 3 - An Example of Neighbor List

Beginning Address	Ending Address	Tree Level	Relationship	Number of Hops	Group Membership
$begAddr_1$	$endAddr_1$	$tree\_level_1$	*parent/child/sibling*	$hops_1$	$grpAddr_1$
$begAddr_2$	$endAddr_2$	$tree\_level_2$	*parent/child/sibling*	$hops_2$	$grpAddr_2$
$begAddr_n$	$endAddr_n$	$tree\_level_n$	*parent/child/sibling*	$hops_n$	$grpAddr_n$

It is worth to note that although (*TTLOfHello+1*)-hop neighbor information can be collected into the neighbor list, the multicast group membership information can be gathered only up to *TTLOfHello* hops because when nodes broadcast their one-hop neighbor list in their Hello messages, the group membership is not included for those one-hop neighbors. Only the broadcast node's membership information is included. This

implies that sometimes we can find a node in our neighbor list, but we don't know its multicast membership information.

4.2.4 Interface to the Next Higher Layer – the Primitives

In order to know when to join and leave a multicast group, the mesh/network layer shall provide an interface to its next higher layer (NHL). This interface is usually defined as primitives between the network layer and its NHL. Examples of the primitives for multicast group membership management are described as below.

4.2.4.1 Joining the Multicast Group

The primitive used to join a multicast group can be defined as below.

```
MULTICAST-JOIN.request  (
                        GroupAddress,
                        GCAddress,
                        JoinAsGC
                        )
```

Table 4 - MULTICAST-JOIN.request Parameters

Name	Type	Description
GroupAddress	Integer	The multicast group address
GCAddress	Integer	The unicast address of the Group Coordinator
JoinAsGC	Boolean	This flag indicates whether this node is joining as the Group Coordinator.

```
MULTICAST-JOIN.confirm  (
                        Status
                        )
```

The Status indicates the result of the join request.

4.2.4.2 Leaving the multicast group

The primitive used to join a multicast group can be defined as below.

MULTICAST- LEAVE.request (
GroupAddress
)

Table 5 - MULTICAST-JOIN.request Parameters

Name	Type	Description
GroupAddress	Integer	The multicast group the node wants to leave

MULTICAST- LEAVE.confirm (
Status
)

The Status indicates the result of the leave request.

4.2.5 The Command Frames

Command frames are used for nodes to update group membership information, to form multicast trees and to leave multicast groups. These commands are modified Hello message and two new command frames, Group Join Request (G-JREQ) and Group Join Reply (G-JREP). Unlike that is in MAODV, the new command frames' purpose is to determine each node's status in a multicast group but not to create routing entries in multicast routing table.

4.2.5.1 Hello Messages

The major function of Hello messages is to exchange connectivity information (i.e. link state information) among neighboring nodes. It provides the foundation for link state routing. A node's group membership information can also be carried by Hello messages. In this work, Hello messages are exchanged under the following circumstances. Note they are not exchanged periodically.

First, when a network is just started, each node exchanges link state information up to TTLOfHello hops in order to build the local link state table. In this stage, the Hello messages are exchanged for unicast routing purpose only and not for building multicast trees.

Second, when a network is in operation, nodes broadcast Hello messages locally when they become members or routers of a multicast group, as well as when they leave a multicast group.

A node shall include all its one-hop neighbors in its Hello messages. Whenever a Hello message is received, the receiver records the sender's information, as well as the information of the sender's one-hop neighbors, in its neighbor list.

An example of the Hello message is illustrated in **Table 6**. In this example, field Multicast Control and Multicast Address List in the Mesh Sublayer Payload are for multicast routing and other fields are used by the unicast routing protocol.

Table 6 - Hello Frame Format

Octets: 2	**2**	**2**	**1**	**…**	**1**	**Variable**	**1**	**Variable**
FC	DA	SA	Command Frame Sub-type	…	Number of one-hop neighbors	Addresses of one-hop neighbors	Multicast Control	Multicast Address List
Mesh Sublayer Header			Mesh Sublayer Payload					

- **Multicast Control Field**

Table 7 - Multicast Control Field

b_7b_6	**b_5~b_0**
Multicast Update Type	Counter indicating the number of multicast addresses included in the multicast address list

Table 8 - Multicast Control Field

b_7b_6	**Multicast Update Type**	**Description**
00	Multicast Address List not presented	When the value is “00”, the “Multicast Address List” field is empty and does not need to be process.
01	Newly joined group	This value indicates the “Multicast Address List” field only includes those multicast groups of which this node just became a member since last hello message.
10	Newly left group	This value indicates the “Multicast Address List” field only includes those multicast groups from which this node just left since last hello message.
11	Full membership update	This indicates the “Multicast Address List” field includes a full list of multicast groups of which this node is a member.

- **Multicast Address List Field**

This list includes all the multicast group addresses of which this node is a member. It can hold up to 64 group addresses. If a node is a member for more than 64 multicast groups, it can use multiple Hello messages to update its group membership to its neighbor nodes.

4.2.5.2 Group Join Request

Table 9 - Group Join Request Frame Format

Octets: 2	**2**	**2**	**1**	**2**	**1**	**1**
Frame Control	Destination Address	Source Address	Command Frame Sub-type	Group Address	Join Options	Hop Count
Mesh sublayer Header			Mesh sublayer Payload			

Table 10 - The Join Options Field

Bitmap	**Options**	**Values**	**Description**
0x01	Join as GC	TRUE, FALSE	When this option is set, it tells the receivers that the Source Address is the GC's address. It also tells the MC that this node will be the GC for the multicast group.
0x02	MC as destination	TRUE, FALSE	When this option is set, it tells the receivers that the Destination Address is the MC's address.
0x04	GC as destination	TRUE, FALSE	When this option is set, it tells the receivers that the Destination Address is the GC's address.
0x08	G-JREP Requested	TRUE, FALSE	When this option is set, the nodes receive the G-JREQ should wait for a G-JREP before change their statuses. Otherwise, they can change their status as soon as the G-JREQ is received since G-JREP is not requested.
0x10	ActAsRouter	TRUE, FALSE	When this option is set, the intermediate node which receives this G-JREQ should change its status to MR if it is not a GM or the GC for the group.
0x20-0x80	Reserved	-	-

4.2.5.3 Group Join Reply

Table 11 - Group Join Reply Frame Format

Octets: 1	**2**	**2**	**1**	**2**	**1**	**1**
Frame Control	Destination Address	Source Address	Command Frame Sub-type	Group Address	Join Option	Status
Mesh sublayer Header			Mesh sublayer Payload			

Table 12 - Join Options of G-JREP

Bitmap	**Meanings**	**Values**	**Description**
0x01	ActAsRouter	TRUE, FALSE	When this option is set, the intermediate node which receives this G-JREQ should change its status to MR if it is not a GM or the GC for the group.
0x02-0x80	Reserved	-	-

Table 13 - Status of G-JREP

Bitmap	**Meanings**	**Values**	**Description**
0x01	Success	TRUE, FALSE	When this option is set, the joining node has successfully joined the targeted node (a GM, the MC or the GC).
0x02	Failure	TRUE, FALSE	When this option is set, the joining node has failed to join the targeted node. The reason can be given by other bits of this field.
0x04	GC not joined yet	TRUE, FALSE	When this option is set, the G-JREQ has reached the MC but the GC has not joined yet.
0x08-0x80	Reserved	-	-

4.2.6 Group Communication Table

Each node involves in multicast communications needs to maintain a group communication table as illustrated below. This table is used to record the multicast groups of which this node is a participant (a member, a router or the GC) and its status in each of the groups. The table is used to forward the multicast data frames and shall be updated whenever a Hello message with multicast extension is received.

Table 14 - Group Communication Table

Group Address	Status (Bitmap)	Number of 1-hop Neighbors to the Group
10	GC	2
1,000	GM	3
11,000	MR	2

Table 15 - Status Field

Status (8-bit Bitmap)	**Description**
0x01	GM
0x02	MR
0x04	GC
0x08	MC
0x10-0x80	Reserved

The status field describes the function a node plays in a multicast group. The statuses include MC, GC, GM, and MR. Note different combination of status may exist for a node, such as MC and GM, MC but not GM, GC and GM. A bitmap can be used to record all applicable status a node has for a group. The “Number of 1-hop Neighbors to

the Group" field may be used to help a node to decide the way a multicast data frame is relayed, unicast or broadcast at the MAC layer.

4.3 Protocol Operation

In this section, the processes of joining and leaving a multicast group are described.

4.3.1 Joining the Multicast Group

A node's process of joining a multicast group is initiated by receiving a MULTICAST-JOIN primitive from the next higher layer and it is not already a member of that group, or when it has a message to send to the multicast group but does not have a route to that group. There are two cases of joining, joining as the Group Coordinator (GC) or joining as a regular member (GM).

4.3.1.1 Join as the Group Coordinator

If the node is set to be the GC of a multicast group (as indicated in the primitive), it shall send a G-JREQ command with the Destination Address set to the MC's unicast address. It shall also set the "MC as destination", "Join as GC" and "G-JREP Requested" option to TRUE. The G-JREQ shall be forwarded to the MC following the unicast algorithm (not necessarily following the tree links).

Intermediate nodes which receive the G-JREQs shall treat them as regular unicast command frames and forward them to the MC. Note when relaying the G-JREQs to the MC, intermediate nodes do not need to cache backward routes in order to route G-JREPs back to the joining node.

Once the G-JREQ arrives at the MC, the MC checks its Multicast Group Record for the specified multicast group. If the record for this group does not exist, the MC creates an entry for this multicast group in its Multicast Group Record. The entry should consist of the multicast group address, the logic address of the GC and the number of members in the group; if the record for this group does exist but with a different address of the GC, it means this new GC is to replace the current GC in the MC's record. The MC shall update its record with this new GC information. If the record for this group does exist but with the address of GC empty and the number of group members larger than one, it means other nodes have tried to join the group before the MC.

The MC shall then reply back a G-JREP toward the source of the G-JREQ with the joining node's unicast address as the Destination Address, the "Join as GC" option set to TRUE and the Status field set to indicate the status of joining. Intermediate nodes between the MC and the GC shall forward the G-JREP back to the GC using the unicast routing algorithm (and that is the reason they don't need to record the backward route when the G-JREQ is received). They don't need to change status to routers because the option "ActAsRouter" is set to FALSE in the Join Options field.

When the G-JREP travels back to the joining node (i.e. the GC), the GC updates its Group Communication Table by creating a new entry for this multicast group and set its status to GC. The joining node is now a member of the multicast group. It shall send a Hello message to its neighbors to update them its multicast group membership.

An example of the GC joining case is shown in **Figure 20**. In the figure, Node F is determined to be the GC of the group. It sends a G-JREQ all the way to the MC, Node A.

The MC records the group and GC information, and then sends back a G-JREP all the way back to Node F. Note in this example it happens the tree route F→E→C→B→A is the shortest path from F to A. In fact, node F can take any shortest path between F and A, such as F→G→C→B→A, F→G→H→B→A or F→G→H→J→A.

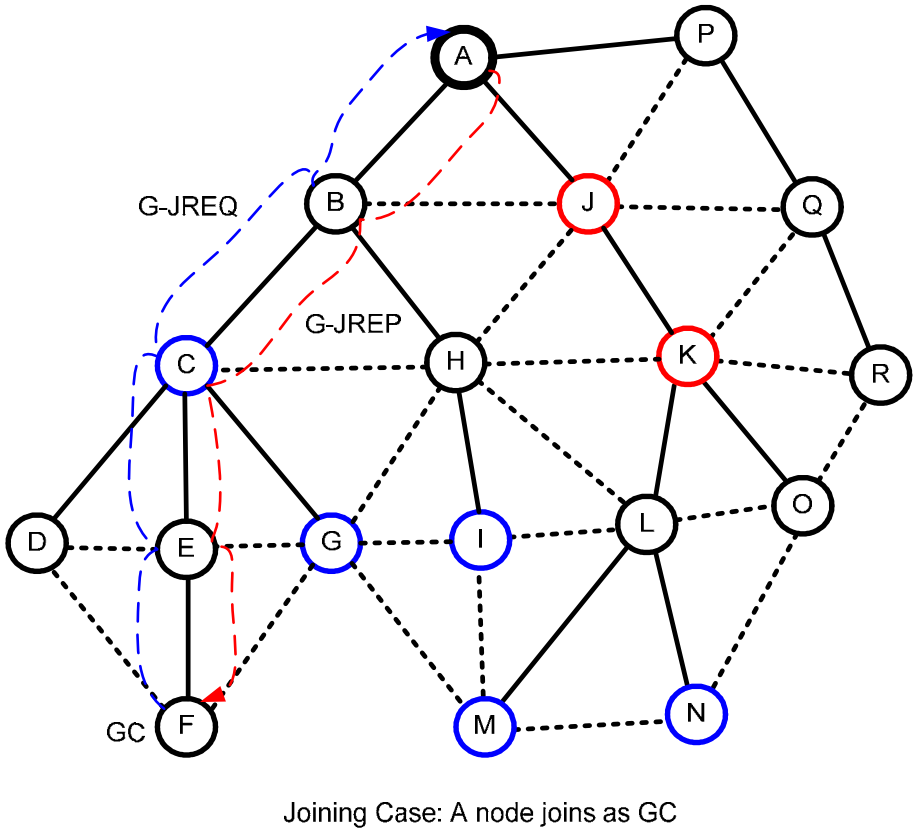

Figure 20 - Joining Case: A Node Joins as the GC

4.3.1.2 Join as a Group Member

Except the GC, all other nodes shall join the multicast following the process described in this sub-section.

The general joining process has the following steps:

(1) Search the neighbor list for existing members in the group.

a. If existing member found, join through one of the members.

b. If existing member not found, go to step (2).

(2) Check whether the GC's address is known

a. If GC's address is known, search the neighbor list for GC to join

i. If GC found in the neighbor list, join through the GC;

ii. If GC not found in the neighbor list, route the G-JREQ to the GC using unicast.

b. If GC's address is not known, check whether the G-JREQ has reached the MC

i. If this node is the MC, find the GC's address in its Multicast Group Records then go to (2)a

ii. If this node is not the MC, forward the G-JREQ to its parent node.

Note nodes don't need to record the backward routes under any circumstances because the G-JREPs can always find their routes back to the G-JREQ originator using the unicast algorithm. This is a save on memory.

Whenever a member or the GC is found in the neighbor list, a node can send a G-JREP back to the source using the unicast algorithm and does not need to wait for the final destination of the G-JREQ to send the G-JREP. Note using the unicast algorithm, instead of the route taken by the G-JREQ, to send the G-JREP could utilize the mesh link so that the route could be shortened.

The following is the detailed explanation of the joining process.

STEP (1) – Search the neighbor list for any member or router to join with

If one or more group member or router is found, the node shall choose one of the members that is the closest to it. When there are multiple member nodes in the neighbor list, it is preferred to choose the tree link because the tree link is established with the consideration of link quality and hence more robust. Mesh links are not created with the consideration of the link quality.

The node shall then send a G-JREQ to the selected existing member using the unicast algorithm (use mesh links if they are better, since the member is in the neighbor list). The Destination Address of the G-JREQ should be set to the member's address. Because the member is in the neighbor list of the joining node, when the joining node sends the G-JREQ, it can optionally set the "G-JREP Requested" flag to FALSE, which means the joining node does not request G-JREP from the member or any other nodes in between them. The ActAsRouter flag of the G-JREQ shall be set to TRUE if the "G-JREP Requested" flag is to FALSE so that intermediate nodes will turn their status to MR for this multicast group. In this case, the joining node should also deem itself has join the multicast group after it has sent the G-JREQ.

When the intermediate nodes in between the joining node and the existing member receive the G-JREQ, they first check the value of the "ActAsRouter" flag in the G-JREQ. If the "ActAsRouter" flag is set to TRUE, intermediate nodes shall change their status to MR for this group as soon as they receive the G-JREQ. If the "ActAsRouter" flag is to FALSE in the G-JREQ, intermediate nodes do not change their status until they receive

the G-JREP. The intermediate nodes shall then forward the G-JREQ to the existing member in the neighbor list of the joining node.

When the G-JREQ finally reaches the existing member, the member shall reply with a G-JREP if the "G-JREP requested" flag is set to TRUE. If necessary, the member node shall also set the "ActAsRouter" flag to TRUE to indicate that intermediate nodes receiving the G-JREP shall change their status to MR. The G-JREP will travel back to the joining node using the unicast routing algorithm. Intermediate nodes shall change their status to MR after receiving and forwarding the G-JREP. The G-JREP will finally travel back to the joining node. If the "G-JREP requested" flag is set to FALSE, the existing member shall not do anything.

After the joining node changes it status to GM and the intermediate nodes change their status to MR, they shall send a Hello message to their neighbors to announce their capability of reaching the multicast group. Note for routing purpose, there are no differences between members and routers. So even routers shall announce they are members of a certain multicast group.

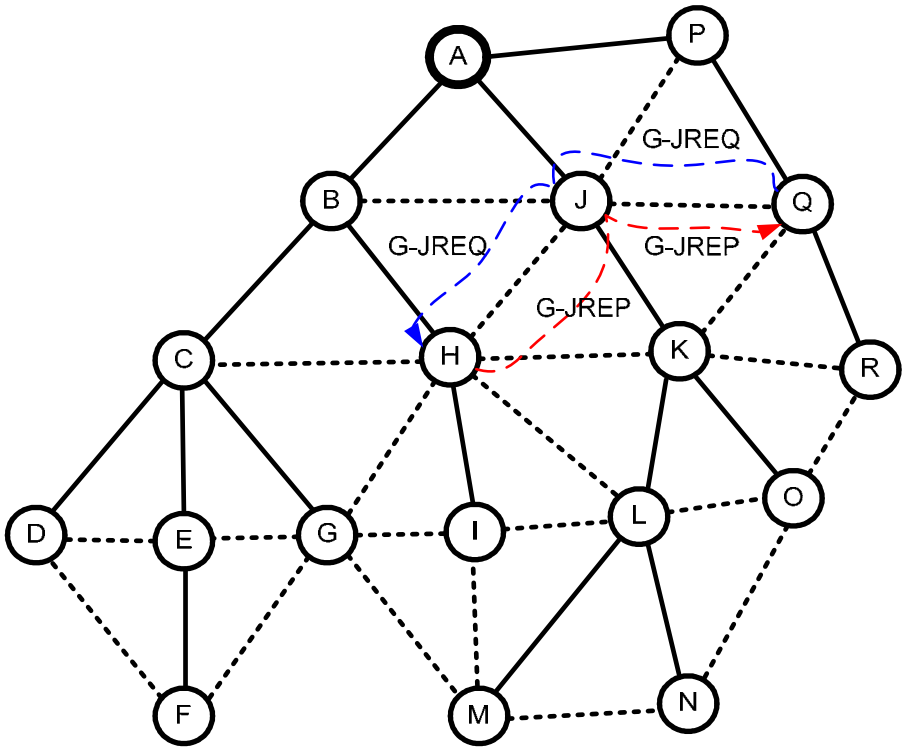

Figure 21 - Joining Case: Members Exist in the Neighbor List

Example:

Assume the TTL of hello message (TTLOfHello) is set to 2. As shown in **Figure 21**, H is a member of a multicast group and Q is going to join this group without given the GC's address. Also assume H is the only member in Q's member list. Q first searches its neighbor list for the members of this group. Q finds H, which is two hops away from it. Q identifies two next hops, J and K, which have the shortest paths to H, using the unicast routing algorithm. Assume Q selects J as its next hop to H and sends a G-JREQ to J (with H's address as the Destination Address and the "G-JREP requested" flag set to TRUE). J uses the similar approach as Q and easily finds H as its one-hop neighbor. Since Q can find H in its neighbor list, which means a route to H can be found without doubt, Q may

optionally change its status to GM without requesting a G-JREP from H or intermediate nodes. It did this by setting the "G-JREP requested" flag to FALSE and the "ActAsRouter" flag to TRUE. When receiving a G-JREQ with "ActAsRouter" flag set to TRUE, J knows it may now become a router of this multicast group even without receiving a G-JREP. Both Q and J shall send a Hello message locally to announce their involvement in the group.

Note although Q's three-hop neighbors, such as C, G, I, M and N, are also in Q's neighbor list, Q doesn't know their multicast group membership. This is due to the fact the TTLOfHello is set to 2, which means only nodes within two hops can send their multicast group membership to Q. Q gets to know the exists of C, G, I, M and N because they were reported by Q's two-hop neighbors.

STEP (2) – Search the neighbor list for the GC if GC's address is known

The GC's address may be given in the MULTICAST- LEAVE.request primitive or in the G-JREQ command frame received from other nodes (when the "GC as Destination" option is set to TRUE, the unicast address in the Destination Address field is the address of the GC).

If the GC is found in the neighbor list, the joining node shall send a G-JREQ to the GC node using the unicast routing algorithm with the GC's address as the Destination Address in the command frame. The "G-JREP requested" flag may be set to FALSE and the "ActAsRouter" flag set to TRUE to join without waiting for the G-JREP.

If the GC is not found in the neighbor list, the node shall continue to route the G-JREQ to the GC node using the unicast routing algorithm. Note even though the GC's address is known and is put in the Destination Address field, it doesn't mean that only the GC can reply with the G-JREP. Any members or routers on the road may reply to the joining node in order for the node to quickly join the multicast group.

Example:

In the example given below, node O is the joining node and node F is the GC. We assume there are neither members nor routers in the neighbor list of node O. We also assume F's unicast address is known to node O when it attempts to join the Group. Since we cannot find any members and GC in the neighbor list, we need to route the G-JREQ using the unicast routing algorithm. We do this by sending the G-JREQ to O's parent node K. K searches it neighbor list for members in this group and fails to find one. K then continues to search the GC in its neighbor list. This time K is able to find the GC, node F, is in its neighbor list. K may send a G-JREP back to node O and forward the G-JREQ toward node F with the "G-JREP requested" flag set to FALSE and the "ActAsRouter" flag set to TRUE. All nodes in between node O and node F (i.e. node K, H, and G) become routers and O becomes a member.

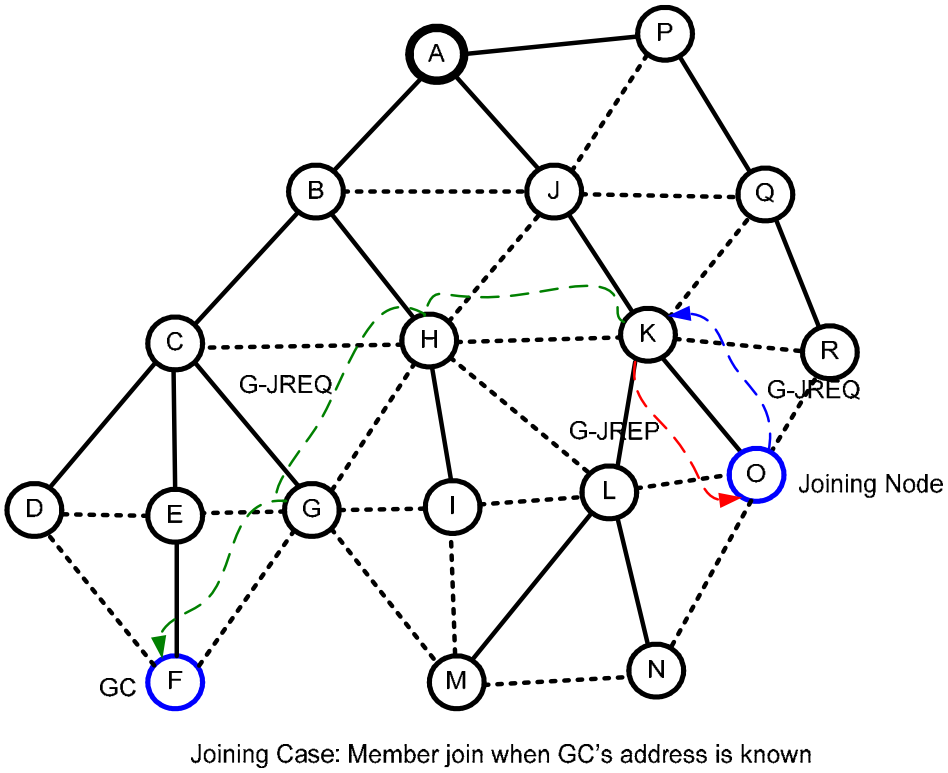

Figure 22 - Joining Case: Member Joining When GC's Address is Known

If the GC's address is not known and the node is not the MC, it shall forward the G-JREQ to its parent node.

If the GC's address is not known but the node is the MC, it shall search its Multicast Group Record for the information of this multicast group. If group and GC information is not available, it means the GC has not presented in the network. MC records the information (group address and increase the counter of number of members presented) and sends a G-JREP with the Status field indicating the reason of join failure.

If the group and GC information is available, the node shall replace the Destination Address with the GC's address in the G-JREQ and forward the G-JREQ to the GC.

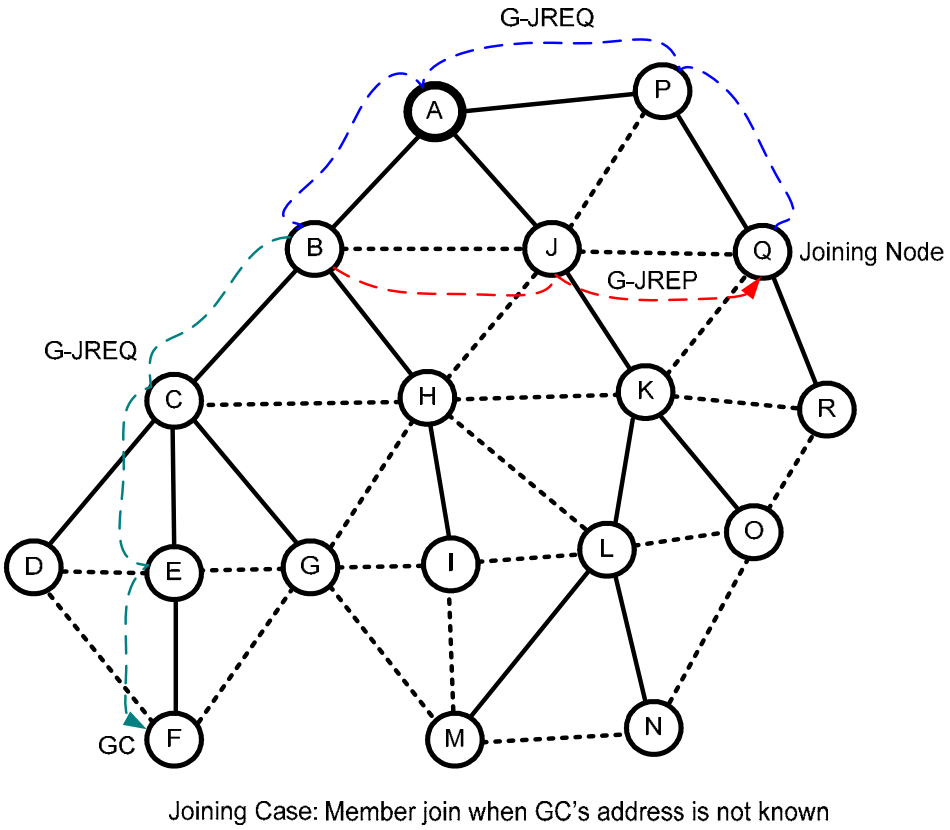

Figure 23 - Joining Case: Member Joining When GC's Address is Not Known

Example:

As illustrated in **Figure 23**, assume Node Q is the joining node and Node F is the GC and the only member in the network for this group. Also assume there are neither members nor routers in the neighbor list of Q. Comparing to the previous example, the difference is the GC's address is not known to the joining node, Q. Therefore, the only thing Q can do is to send a G-JREQ to its parent node, P, with the "G-JREP requested" flag set to TRUE and the "ActAsRouter" flag set to FALSE. P can find neither a member nor the GC in its neighbor list, so it forwards the G-JREQ to its own parent node, which in this case happened to be the MC, node A. Since the GC has already registered with the

MC, we now know where the GC is. So from now on put the GC's address in the Destination Address field and route the packet toward the GC using unicast algorithm.

When the G-JREQ reaches node B, node B finds the GC (node F) is in its neighbor list. Node B can decide whether it should wait for the G-JREP from the GC or reply with its own G-JREP back to the joining node. Assume Node B decide to reply with its own G-JREP back to the joining node. It shall send the G-JREP using the unicast algorithm with the "ActAsRouter" flag set to TRUE. In this example, the G-JREP will take the route B→J→Q using its neighbor list instead of following the route the G-JREQ took (B→A→P→Q). This is the reason we don't record backward routes to the joining node – a better route may be found from the other direction.

At the mean time, node B shall continue to forward the G-JREQ to the GC using the unicast routing algorithm. The "G-JREP requested" flag may be set to FALSE and the "ActAsRouter" flag may be set to TRUE so that nodes C and E will change their status upon receiving the G-JREQ.

4.3.1.3 Summary of the joining process

The joining process can be summarized by the following flow chart.

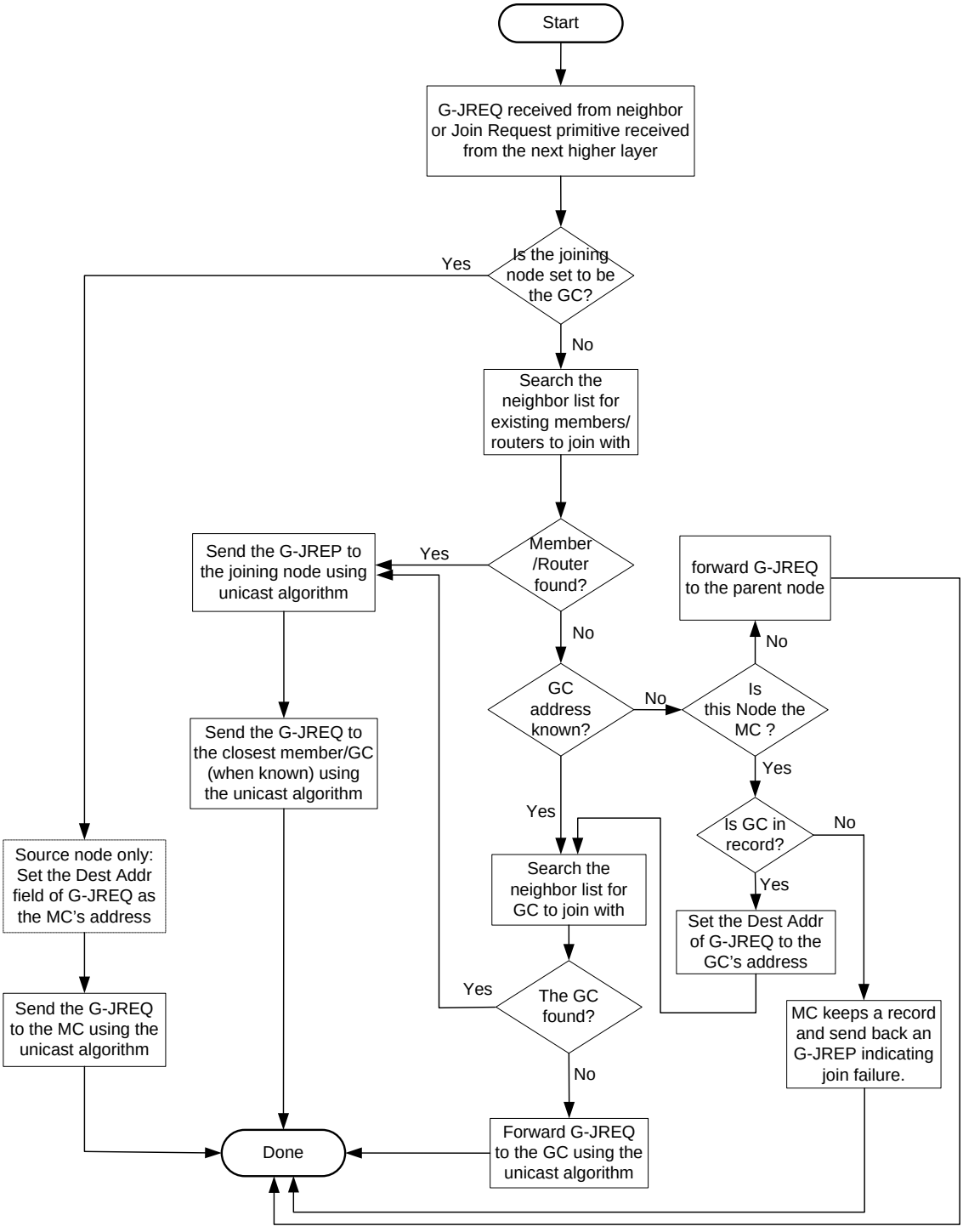

Figure 24 - Flow Chart for G-JREQ/Join Request Primitive Process

4.3.2 Leaving the Multicast Group

A multicast member can leave a group at any time. However, before it leaves a group, it first has to check whether it is a leaf node of the multicast tree. To do this, it checks the number of members among its one-hop neighbors which belong to this group as indicated in the "Number of 1-hop Neighbors to the Group" field of the Group Communication Table. If the number is 1, it can leave the group. Otherwise, it can only change its status to MR and still remain in the multicast tree. In this case, no Hello message needs to be sent for status update. The node only needs to change its status in the Group Communication Table.

To leave a multicast group, a node broadcasts a Hello message to its neighbors with the Multicast Update Type set to "Newly left group" and the multicast group address included in the "Multicast Address List" field. Neighbors receiving this Hello message shall update their neighbor table to reflect the fact that they are no longer able to reach the multicast group through this node.

4.3.3 Repairing Multicast Routes

Whenever a 1-hop neighbor leaves a multicast group or becomes unreachable, a node should reduce the "Number of 1-hop Neighbors to the Group" in the Group Communication Table by one. When this number becomes zero, the node has lost the connection to the multicast group. It shall restart the process of joining the group and try to reconnect to the multicast tree.

4.3.4 Multicast Data Forwarding

Unlike traditionally routing protocol, a routing table is not needed for multicast data forwarding. The decision of forwarding or not fully depends on the status (e.g. member, router and etc.) of the node. When a multicast data frame is received from NHL or from a neighbor node, the node first checks its status in this multicast group by looking up the Group Communication Table.

If the corresponding entry exists and the node is the GC, a GM or a router of this multicast group, it shall relay the frame by broadcast at the MAC layer. The GC and GMs also need to send the data frame to their NHL for data process.

If the corresponding entry does not exist or if the entry exists but the node is not the GC/a GM/a router, the node should check where the data frame comes from. If the frame is from a neighbor node, it shall discard the frame because it is not on the multicast tree and shall not route the data frame for this multicast group. If the frame is from its NHL, it means the node wants to be a member of this group. The node shall initiate the joining process to join the multicast group first and then relay the frame. The data frame shall be stored in the buffer of this node and forwarded after the node joins the multicast group.

To reduce the broadcast traffic at the MAC layer, a node checks the number of 1-hop neighbors which are members or routers of the multicast group before forwarding the data frame (using the multicast communication table).

- If the number is one, then this node is a leaf node and the data frame must be received from its own NHL. It should just unicast the data frame to its next hop member.
- If the number is two (including the neighbor from which it received the frame), it has to determine whether the data frame is from its own NHL or received from other nodes. If the frame is from its NHL, it should broadcast the data frame to its next hops; otherwise, this frame is from one of its two member neighbors. It may choose to unicast the data frame only to the other next hop member.
- If the number is more than two, the node shall just broadcast the data frame at the MAC layer.

Note when some nodes in the network are running in the low-power mode, broadcast takes more effort (longer delay) to be achieved. So it is better to use unicast whenever it is possible.

4.4 Summary

In this chapter, we introduced a new hybrid multicast routing protocol specifically designed for wireless sensor networks. The protocol is built on top of a hybrid unicast routing structure that combines a routing tree and local link state information. It hence utilizes the advantages from both methods; by using the logical tree, control frames for joining and leaving a multicast tree can be unicast, instead of being broadcast; by using the local link state information, the inefficiency of tree routes can be partially compensated and tree repaired issues can be solved. In addition, minimal level of control

overhead is achieved because existing control messages designed for unicast protocol can be used for multicast protocol too.

5 A SMART BROADCAST SCHEME FOR WIRELESS AD HOC NETWORKS

5.1 Introduction

Broadcasting has many applications in wireless ad hoc networks, and is expected to be utilized more frequently than in wired networks. In the applications layer, service and device discovery usually depend on broadcasting. Sometimes, data distribution across all network nodes can take advantage of broadcasting for efficiency. In network layer, broadcasting is a fundamental component for many routing protocols. Both proactive and reactive routing protocols utilize broadcasting to achieve their goals. Proactive routing protocols update their routing tables by periodically exchanging 1 or 2-hop neighbor information via broadcasting so that whenever a route to any destination is needed, it is immediately available; on the other hand, reactive routing protocols do not discover and maintain the routes until they are requested by the applications. When a route is needed, the source node will typically broadcast a route request to the network and then pick the best route from multiple possible route replies responded by either the destination or intermediate nodes that have fresh enough routes to the requested destination.

There are many other applications of broadcasting in wireless ad hoc networks. However, broadcasting may lead to severe “Broadcast Storm” problems [20] if it is done by simple flooding in a contention-based channel, such as CSMA/CA. The major problems include redundant rebroadcast (overlaying), contention and collision. Actually, broadcasting often incurs congestions in dense networks, which will then affect the timely delivery of packets, traffic handling capacity of the network, or the number of end systems the system can handle

In this work, we study the problems of simple flooding and develop a smart scheme to efficiently broadcast packets in a wireless ad hoc network. We note here that reliable broadcasting is not the main subject of this work, which means there is not 100% guarantee that broadcast packets will reach every node in the network. We assume the use of a single common channel and nodes are equipped with Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) MAC layer protocols. It is also assumed that global topology information is not available.

The remainder of this section is organized as follows. Section 5.2 presents the broadcast storm problem and its generation. Section 5.3 lists existing solutions to the problem and Section 5.4 evaluates them. The design of our Smart approach is presented in Section 5.5. Section 5.6 describes the simulation design and the environment of simulation. In Section 5.7, the simulation results are analyzed in details. Section 5.8 gives discussions on related topics, such as the link quality. Finally, Section 5.9 concludes the chapter.

5.2 The Broadcast Storm Problem

The broadcast storm problem refers to the baleful results led by simple flooding [21][22]. The results include overlaying (redundant rebroadcast), contention and collisions. Before we further discuss the problem, we give introduction to some concepts and protocols.

A. Broadcasting

We consider broadcasting a mechanism of delivering data packets from one node to all other nodes in the network by best-effort service, if these nodes are physically reachable.

Please note, again, we are not studying reliable broadcasting. Once broadcast, packets will be transmitted relying on the service provided by underlying MAC/PHY layers. A packet could be dropped due to collisions in the MAC layer or buffer overflow in any layer. Note when CSMA/CA MAC is used, the MAC layer acknowledgement is inapplicable for broadcasted packets. This means when packet loss happens, the sender has no idea of it and hence no retransmission will be done to recover the loss. Reliable broadcasting is assumed a function of transport layer protocol or even application layer protocol.

B. Flooding

We consider flooding as a straightforward and the simplest scheme of achieving broadcasting. By flooding, each node rebroadcast a packet exactly once upon its first reception of the packet. That is say, if simple flooding is used to broadcast packets in a network of N nodes, it will take N-1 re-floodings for a packet to reach all the nodes. It is also referred as simple flooding hereafter.

C. CSMA/CA and IEEE 802.11

We assume the use of a single common channel and nodes are equipped with Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) MAC layer protocols. The typical example of this protocol is the IEEE 802.11 (Wireless LAN) standard [29].

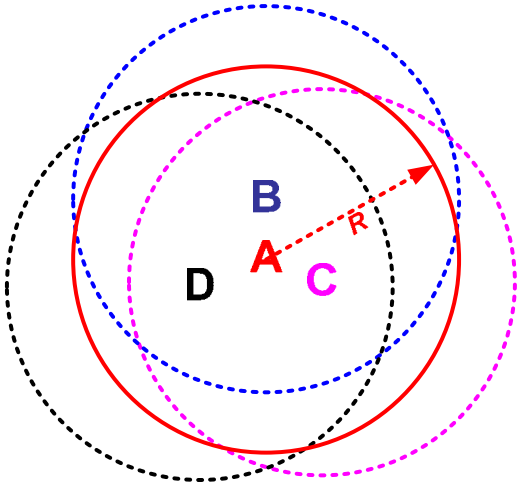

Figure 25 - Overlaying Problem

Now we are ready to discuss the broadcast storm problems. Let's first look at the overlaying problem. As shown in **Figure 25**, Node A can cover a bunch of nodes around it with communication range R when it broadcast a packet. It is obvious that nodes close to the sender A (like Node B, C and D in **Figure 25**), will cover almost the same area that has been covered by A when they rebroadcast the packet. The result is that some areas will be covered multiple times unnecessarily, especially in a dense network. This is a waste of network resources and more importantly, it may lead to the second problem, contention.

When a node floods a packet, all its neighbors will receive the packet at almost the same time. If all the neighbors decide to rebroadcast the packet (as it is done by simple flooding), they may rebroadcast the packet at highly correlated time in the nearby area. This will lead to severe contention for the common channel. Nodes have to backoff when this happens. If the contention cannot be resolved after several attempts, some packets

might be dropped permanently – packets collide. The contention also increases the packet transmission time because of the backoff each time it happens.

Simple flooding can lead to other harmful effects in wireless ad hoc networks, such as extra power consumption and finding non-optimal routes (in terms of number of hops). Unnecessary flooding will cause shortened battery life, which is a fundamental concern for wireless ad hoc and sensor networks.

5.3 Related Works

Some approaches have been proposed to solve the broadcast storm problem. Williams *et al* [19] studied the existing broadcast techniques in wireless ad hoc networks and categorized the protocols into four families: Simple Flooding, Probability Based Methods, Area Based Methods and Neighbor Knowledge Methods. Except Simple Flooding, all other methods can more or less relieve the Broadcast Storm problem by adding different levels of control overheads to the network or/and processing overheads to network nodes.

5.3.1 Probability Based Methods

There are two broadcasting schemes fall into this category, the Probabilistic Scheme and the Counter-Based Scheme.

In Probabilistic Scheme [20], nodes do not rebroadcast every packet they received but rebroadcast packets with a predetermined probability, p, where $0 < p <= 1$. The biggest challenge of this scheme is the selection of probability p. Without knowing the topology of the network, setting a high probability in a dense network will not relief the

broadcast storm much while setting a low probability in a sparse network will dramatically reduce the reachability of the broadcasted packets. In short, this scheme is not adaptive. It may only be used in a fixed scenario with known node density.

In Counter-Based Scheme [20], nodes randomly back off and start a counter for each new broadcasted packet they received. During the backoff period, nodes update their counter every time the same packet is received and compare their counters to a predetermined counter threshold. If the value of the counter is equal to or larger than the predetermined threshold, they drop the packet without rebroadcasting it; otherwise, nodes rebroadcast the packet when their backoff timers expire. Again, this scheme suffers from the same problem of Probabilistic Scheme. Determining the value of the counter threshold is a big challenge.

5.3.2 Area Based Methods

The designers of Area Based Methods focus on how much extra area (than the area covered by the original broadcast) can a node offer if it rebroadcasts the packet. If the extra area a node can cover is large, it rebroadcast; otherwise, it does not. Two schemes were proposed in this category, the Distance-Based Scheme [20] and the Location-Based Scheme [20].

The Distance-Based Scheme estimates the distance between neighbor nodes to make the decision of rebroadcast. It utilizes the relationship between the receiving power of a packet and the distance the receiving node is from the transmitting node. Because the signal power fades when it propagates from the sender to the receiver, the receiver can estimate its distance from the sender by monitoring the receiving power and convert it to

distance using certain propagation model, such as free-space model. Because we will compare our algorithm to Distance-based Scheme later in this paper, we summarize the Distance-based scheme as follow.

> *A node running Distance-Based scheme considers only distance, d, between neighbor nodes and itself. When a broadcast packet is received, it estimates its distance from the sender. If the distance is within the threshold, D, no further process is needed; otherwise, it backoffs randomly. During the backoff period, if it receives one rebroadcast packet from a neighbor that is within the distance threshold, D, the node then suppresses itself from rebroadcasting the packet; otherwise, it rebroadcasts the packet when the backoff period ends.*

In Location-Based Scheme, a more precise estimation of extra coverage area can be made by using some sort of positioning systems, like GPS. When a node broadcasts a packet, it includes its position information in the header of the packet. Upon receiving a broadcast/rebroadcast packet, nodes calculate the extra area they can cover and then determine whether to rebroadcast or not. Although this scheme is proved by simulation in [20] to be the best one among all protocols studied, the assumption that each node is equipped with a GPS-like device is too strong.

5.3.3 Neighbor Knowledge Methods

Protocols in this category usually assume the knowledge of 1-hop [23] or 2-hop neighbors [24][25][26][27][28]. This knowledge is obtained by exchanging “Hello” type messages periodically among neighbor nodes. Nodes usually include their neighbor list in

the broadcasted packets. Upon receiving the packet, receivers determine whether to rebroadcast the packet by comparing their own neighbor list and the neighbor list of the sender. If the receiver can cover some extra nodes which are not in the list of the sender, it rebroadcast; otherwise, it does not.

Schemes in this category can provide accurate information for a node to determine whether to rebroadcast or not. However, the extra overhead led by Hello messages is the most arguable issue. Besides, these schemes usually suffer from changing topologies. When neighbors move in and out of a node's communication range, the neighbor list recorded before this movement provides inaccurate information. Therefore, we believe methods belong to this category are not good candidates for variable topology networks.

5.4 Analysis of the Existing Approaches

According to our understanding, a good broadcast scheme must be, first, simple enough so that it can be used by any wireless device, from notebook computers to tiny wireless sensors. Second, because of the dynamic nature of the wireless mobile network, the approach must be able to adapt to changing topologies. Third, it must be able to adapt to different network densities. Lastly, it must be able to adapt to node movement.

Based on this understanding, we made the following criteria for good broadcasting approaches.

1) The algorithm should not introduce extra network traffic. Therefore, we do not consider Neighbor Knowledge Methods as good ones. Periodically exchanging

Hello message among neighbor nodes is expensive because Hello messages themselves are broadcasted, one or more hops.

2) The algorithm should not make strong assumptions, such as the use of GPS devices. Most of the mobile devices would not have it, especially those tiny wireless sensors.

If we filter the existing approaches reviewed in the last section with our design criteria, only Counter-Based [20] and Distance-Based scheme [20] can be considered roughly satisfactory. However, both of them have their own problems. We address these problems in this section.

5.4.1 Random Backoff

Generally, there are two ways to relieve the broadcast storm effect: reducing the possible redundant rebroadcast and differentiating the timing of rebroadcast. Both Counter-Based and Distance-Based schemes backoff a random number of time slots before rebroadcasting the packet. This backoff time is referred as “Random Assessment Delay” and is a random number uniformly chosen between zero and the maximal allowed delay.

Although random backoff can alleviate the contention to some degree, it is kind of a “blind” backoff scheme, which means the backoff scheme cannot “see” the direction of packet propagation. We illustrate this by giving an example.

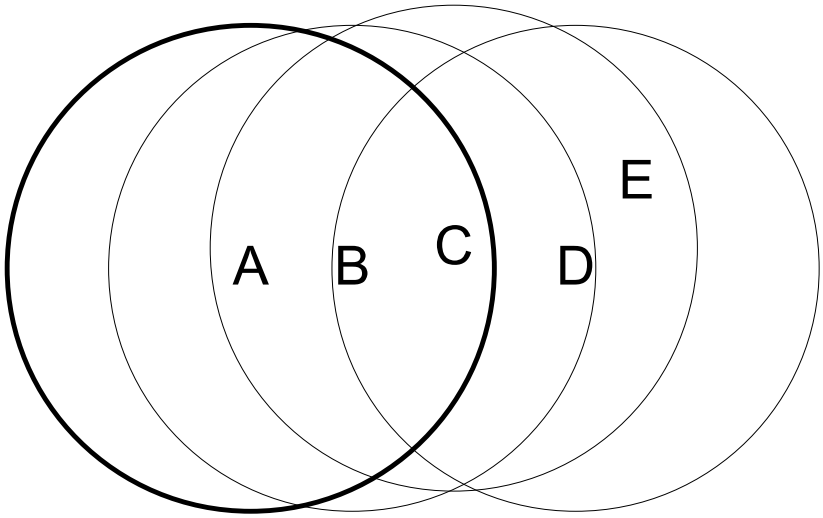

Figure 26 - Random Backoff is "Blind"

Let us assume we are using the Count-Based scheme and the threshold is preset to 2. As shown in **Figure 26**, Node A is the broadcasting node and Node B, C, D are receiving nodes. When Node B, C and D receive Node A's packet, they first backoff randomly and update their counter to 1. Because the backoff is random, Node B may schedule a rebroadcast timer early than the timers of Node C and D (in this case, the chance is 1/3). Therefore, when Node B's backoff timer expires, it rebroadcasts the packet. This packet will then be received by Node C and D again. Upon receiving the rebroadcast packet from Node B, Node C and D now reach their counter threshold, 2. As a result, they drop their packets without rebroadcasting. However, from **Figure 26**, we can easily tell that the extra area Node B can cover is very limited because it is very close to the sender Node A. Assuming there is a node E outside the coverage of A, Node B will miss it also. That is why we call random backoff a "blind" scheme. Because backoff scheme is usually combined with the rebroadcast suppression scheme to achieve the goal of

efficient broadcast, it is important to have an efficient backoff scheme that really works. The random backoff used in this example is certainly not the best one.

Now let's look at the opposite side. If we let Node D or even Node C rebroadcast the packet earlier than Node B, Node E will be covered. This shows that it is desirable the backoff is directional - the backoff time for each node increases from the transmission border to the sending node, no matter which efficient broadcast scheme we use. We refer this as "*Directional Backoff*" and have developed a way of doing it. The details will be discussed later in Section 5.5.

5.4.2 The Adaptivity of Counter-Based Scheme

As we mentioned earlier, the counter threshold in the Counter-Based scheme is very important. The determination of its value greatly affects the efficiency and the reachability of the scheme. In a dense network, each node has many neighbors. Suppressing more nodes from rebroadcasting would not reduce the reachability but can increase the efficiency. In this case, nodes may like to choose a smaller threshold; due to the similar reason, nodes may like to choose a larger threshold in a sparse network. Therefore, the greatest issue is the reachability. Without the knowledge of network density and topology, Counter-Based scheme fails to balance well between efficiency and reachability. The authors of Counter-Based scheme also proposed an adaptive version over the original one in [20]. However, it assumes the knowledge of neighbor and is out of the scope of this study.

5.4.3 Wrong Decisions Made by Distance-Based Scheme

A node running Distance-Based scheme considers only distance information between neighbor nodes and itself. If there is (are) one or more neighbor(s) which is (are) within the distance threshold, *D*, the node then suppresses its own rebroadcast. We believe this scheme is too simple to make right decisions in many situations. Let us look at some of the examples.

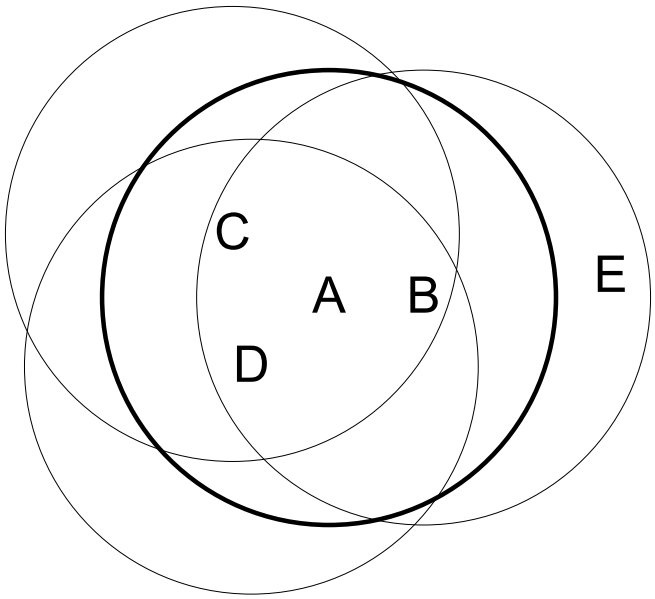

Figure 27 - Wrong Decision Made by Distance-Based Scheme

In **Figure 27**, Node A is the sender and Node B, C, D are the receivers as before. We assume Node B's distance from Node A is smaller than the threshold *D*. Therefore, when Node B receives the broadcasted packet from Node A, it decides not to rebroadcast the packet. However, from the figure we can see that Node B is the only node which can cover Node E. This loss of reachability may lead to the missing of a total branch starting

from Node E. This scenario shows that solely relying on distance information to make the rebroadcast decision in not enough.

5.4.4 The selection of the slot time

Although the authors of [20] did not mentioned how many slots they used in their simulation and the duration of each slot. Our simulations of the Distance-Based scheme showed much different results with different slot durations. Note the random backoff here is not the exponential backoff happens in the MAC layer. Most of the efficient broadcast schemes run at the routing layer. We refer it as a Broadcast Control Layer (BCL). It could be a lower sub-layer of the network layer which directly interfaces with data link layer. At BCL, nodes have no way of knowing the contention and hence the backoff window will be fixed. The choosing of right backoff window size and the slot duration depends very much on the parameters like the number of neighbors (node density), the transmission and processing time of the broadcasted packet, and etc. We will discuss more about this later in this section.

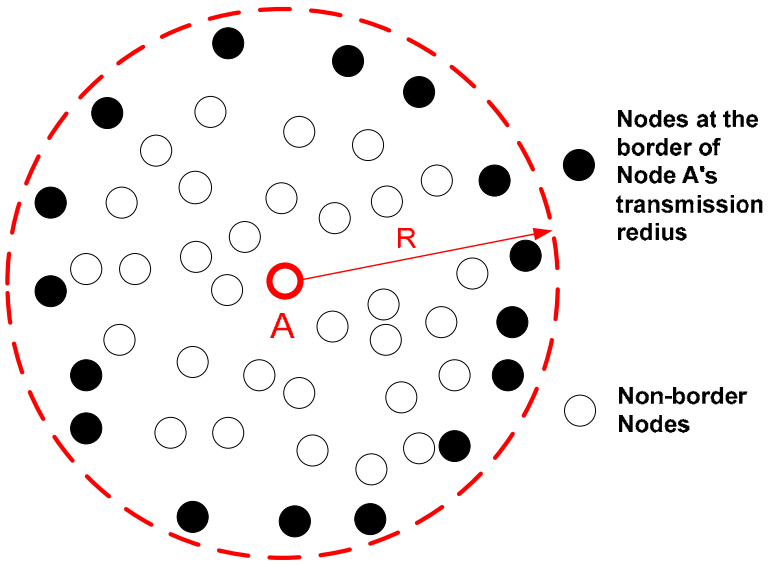

Figure 28 - A Dense Network

5.5 A Smart Approach

Our idea of designing an efficient broadcast approach was triggered when staring at a dense network topology similar to the one shown in **Figure 28**. The immediate idea was why every node has to rebroadcast the packet? The rebroadcasts of the nodes at the border of the sender's transmission range will be enough in a dense network. Nodes close to the sender will not provide more extra coverage than the nodes at the border. Therefore, they do not need to rebroadcast. The question then becomes how to determine which nodes are at the border of the sender's transmission range. The answer came to our mind was the relationship between the receiving power and the distance a receiver is from the sender. One of the reasons of choosing the receiving power is that it is now wildly available. Most of the WLAN and WPAN modules now provide quantized link quality to the utility software. Although the data provided may be signal to noise and interference ratio (SNIR), it can still be used to estimate a node's distance to the sender. An additional assumption is that nodes in the wireless ad hoc network are homogeneous – they use the same fixed transmission power.

The following describes the components of our algorithm.

5.5.1 Distance Estimation

When a node receives a flooded packet, it checks the power of the received signal and estimates its distance from the sender. The simplest way is using the free space propagation model described by the following equation.

$$P_r = P_t \, (C_1/d)^n \, C_2 \qquad (11)$$

where P_r is the receiving power, P_t is the transmitting power, n is a constant between 2 and 4, C_1 is the carrier's wavelength and C_2 is the antenna gain. Since P_r can be measured and P_t is known, the distance d can then be calculated.

In reality, radio channels are extremely random and very difficult to analyze [30]. The transmitted signal might have experienced reflection, diffraction and scattering when it arrives at the receiver. In order to model the signal propagation characteristics of radio channels, researches are traditionally focused on two aspects of channel modeling, large-scale modeling and small-scale modeling. Large-scale models try to predict the mean signal strength for different separation distance between the transmitter and the receiver; while small-scale models attempt to characterize the rapid fluctuations of the received signal strength (caused by multipath effect, Doppler shifts and etc.) over very short travel distance or short time durations. Typical large-scale models include Friis free space model and ground reflection (2-way) model; Rayleigh and Recian fading distributions are commonly used to analyze the small-scale fadings.

The above says that, in the most cases, the calculated distance using free space model will not be accurate because the received power is a result led by the combination of the large-scale fading and small-scale fading. Therefore, one of our major works is to make sure that our algorithm is robust even when the estimation is inaccurate. To achieve this, we calculate the receiving power using free space model and use it as the mean value of the receiving power for one specific distance. We then generate random variances to the mean value and calculate the distance using this varied power. The resultant distances are certainly not the real distances between the senders and the

receivers. We then use these inaccurate distances to test the performance of the Smart scheme. Details about this will be given in the simulation result section (Section 5.7)

5.5.2 Directional Backoff

As discussed earlier, random backoff is "blind" in that it cannot tell the direction of packet propagation. In our scheme, we replace the "blind" backoff with directional backoff by linking the distance information to the backoff scheduling. The algorithm works as below.

Assuming the backoff window size is fixed and equals to N, and the transmission range of the sender equals to R. Then $U = R/N$ is the unit distance if we chop the range R into N units. When distance d is estimated, we would like to see which unit the receiver falls in. Therefore, we calculate the receiver's distance level $u = ceil\ (d/U)$, where function *ceil ( )* returns the smallest integer of d/U.

By doing above calculations, every node falls into a distance level between 1 and N. The distance level increases from the sender to its border of transmission range. Of course, some nodes may fall into the same distance level.

The next step is to determine the backoff slot S for each node. It is done by the following equation.

$$S = (N - u) \tag{12}$$

This equation indicates that receivers with larger distance levels rebroadcast earlier than those with smaller distance levels. The goal of directional backoff is now achieved – the further a node is away from the sender, the earlier it rebroadcasts the packet.

Please note the selection of N affects the performance of the algorithm. If N is too small, then multiple nodes may be classified into the same distance level and assigned the same backoff timer. When their timers expire at the same time, these nodes may collide with each other; if N is too large, then the overall delay will increase due to the backoff slot time calculation algorithm described below.

5.5.3 Slot Time Calculation

Note the interval between any two consecutive back-off slots must be long enough for a node to finish re-flooding a packet. Otherwise, a node with distance one level lower than the rebroadcasting node will try to re-flood the packet while this node is still rebroadcasting and hence lead to channel contentions. To achieve this, we calculate the packet transmission time (packet size and link speed are needed for calculation) when determining the slot time of backoff. This can be done in a per-packet base since every packet has different length. In short, the reaction between the rebroadcasting backoffs and MAC layer backoffs must be considered and the backoff timer needs to be tuned. As we just mentioned, the Counter-Based and Distance-Based scheme do not pay attention to this problem.

5.5.4 Rebroadcast Suppression

To achieve the goal of efficiency, decisions must be made when nodes feel they may not need to rebroadcast the packet. The basic decision rule of our algorithm is very

simple: if a node hears one rebroadcast of the same packet it just received before its backoff timer expires, it simply drops the packet; otherwise, it rebroadcasts the packet when its backoff timer expires. Please note that combining with directional backoff, this simple rule let the nodes close to the border of sender's transmission range rebroadcast earlier than the inner nodes. Every time a node receives a rebroadcasted packet, it knows the packet is from a node closer to the border than it is. Therefore, it gives up its own rebroadcast. By this way, the rebroadcast nodes can cover the biggest possible extra area with the smallest possible number of rebroadcasts.

5.5.5 Reachability Enhancement

The above four algorithms construct the basic schemes of Smart broadcasting. However, like all other efficient broadcast schemes, basic Smart scheme has to face the reachability problem. Because not all the nodes rebroadcast the receiving packet, basic scheme may lead to degraded coverage.

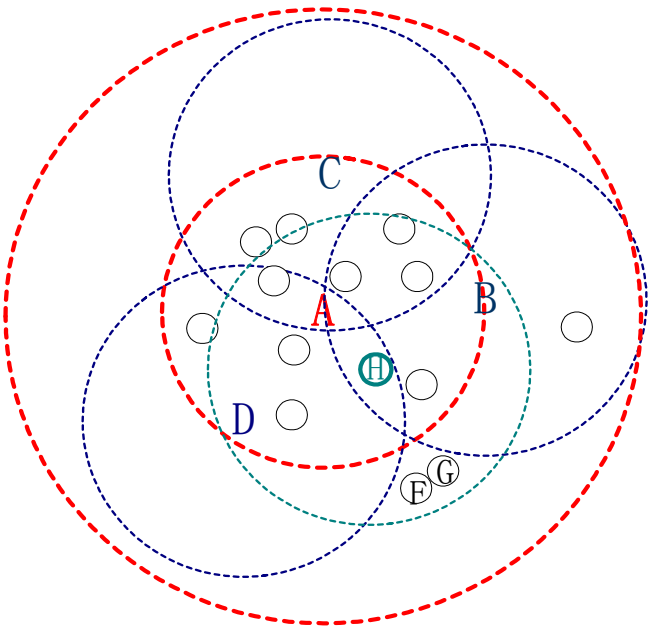

Figure 29 - The Reachability Problem

In an example shown in **Figure 29**, Node H's rebroadcast is suppressed by node B or D, because Node B and D are closer to the border than Node H. However, due to this suppression, node F and G will not be able to receive the broadcast packets they are supposed to receive. Therefore, the simple rebroadcast suppression algorithm needs to be improved if reachability is of concern (usually in sparse networks).

To solve the problem, a Reachability Enhancement algorithm is developed. The following information is utilized to help a node decide whether to rebroadcast the packet in order to enhance the reachability.

1) The number of rebroadcast packets a node has heard before its backoff timer expires. A counter *c* is used to record the number and a threshold *C* is used to make the decision.
2) The estimated distance (*d*) a node is from each sending node of the broadcast packet, including the original packet and each rebroadcast packet. A threshold *D* is used to estimate whether a node is closer to the border or closer to the original sender. The same threshold is also used to determine whether rebroadcast nodes are close to the node that is making the decision or not.

Simulation is used to help determining the right values of thresholds *C* and *D*. The following decision table is the result of our simulation study. The counter threshold *C* is set to 2 for all scenarios.

Table 16 - Decision Table

Number of Rebroadcast Heard (c) with $C = 2$	Distance this Node is from the original sender (d)	Distance this Node is from rebroadcast senders (d')	Rebroadcast the packet?
$c = 1 < C$	Any	Any	**Yes**
$c = 2 = C$	$d>D$	Both $d' > D$	**Yes**
		else	**No**
	$d<D$	Any	**No**
$c > C$	Any	Any	**No**

From the table one can see that a node must rebroadcast the packet if it has heard only one rebroadcast before its timer expires ($c=1$). It does not even care its distance from the original broadcast node and the rebroadcast node; hearing only one rebroadcast means there is only one node in its communication range that is closer to the border of the original sender than itself. In this case, if it rebroadcast, it should have a good chance of covering extra nodes.

When the number of rebroadcast packets heard is 2 ($c=2=C$), the situation becomes a bit more complex. We describe the algorithm in two steps.

First, in order to determine whether to rebroadcast or not, the node must look at its distance from the original sender of the broadcast packet, d. In **Figure 29**, assuming Node H is the node making the decision, then d is the distance from Node H to Node A. If it finds itself far away from the original sender ($d>D$), it then processes to the second step to check the distances between the rebroadcasting nodes (Node B and Node D in **Figure 29**) and itself (d'); otherwise, it simply drops the packet.

In this second step, we use the concept of "*Relative Position*" to make the decisions. The reason a node can hear rebroadcast packets from two different neighbors is that these two neighbors are at least *R* meters away from each other (*R* is the communication range). In other words, these two nodes must be at the different sides of this node; otherwise, one node's rebroadcast would have been suppressed by that of the other. As shown in **Figure 29**, because of the fact that Node B and Node D are away from each other, it is safe to say that if Node H rebroadcasts, it can cover some area which cannot be covered by its two neighbors. In the figure, it is clear that Node H can cover Node F and G if it rebroadcasts. In general, the farther the two neighbors are away from each other, the larger the extra area the middle node can cover. By simulation, we found that a node needs to rebroadcast only when both of its rebroadcast neighbors are far from it ($d'>D$). We suggest the threshold *D* takes the value *R/2*, where *R* is the communication range of the sender.

When the number of rebroadcast packets heard is larger than 2 ($c>C$), simulation results indicate that it is not necessary for a node to rebroadcast the packet. This is also confirmed by the simulation result in [20] which says if a node has heard a packet three times (one original transmission and two rebroadcasts, the same case as $C=2$), the average extra area it can cover is only about 9% of its total coverage (πR^2). Of course this result is obtained using simple flooding. On average, Smart Broadcast covers larger extra area than that of simple flooding because the distance information is used intelligently.

5.6 Simulation Design and Environment

To examine the performance of the Smart scheme, we implemented it using OPNET Version 8.0. We also implemented Distance-Based scheme for comparison. Distance-

Based scheme is picked because it roughly satisfies the requirements we listed in Section 5.4 and also utilizes the estimated distance as the hint for solving the problem.

In order to make our simulation results comparable to that of the Distance-Based scheme, we make our simulation environment close to their settings. The simulated network contains 100 nodes. These nodes are placed randomly within a fixed-size L x L area called *map*. A *map* can be of size 1x1, 3x3, 5x5, 7x7, 9x9 and 11x11 units, where a unit is of the length of the communication radius. The communication radius of the nodes is set to 100 meters. For each size of the *map*, 10 different seeds were used to generate different network topologies. The average number of neighbors of each node in different size of *map* can be found in **Table 17**.

Table 17 - Average Number of Neighbors

Map Size (m)	Average Number of Neighbors
100 x 100	96.804
300 x 300	25.596
500 x 500	10.32
700 x 700	5.556
900 x 900	3.364
1100 x 1100	2.272

When the simulation starts, each node in the network randomly picks a time to broadcast its packet exactly once during the simulation. The backoff window is set to 32 for simulated schemes. The MAC layer model is the OPNET implementation of IEEE 802.11 WLAN model with some minor modifications. RTS/CTS option is disabled. The speed of the Wireless LAN is set to 1Mbits/sec. There is only broadcast traffic in the network. All simulations were run for 100 seconds.

The free space propagation model is used to estimate the distance between sending and receiving nodes. To simulate the inaccurate estimation of the distance, we use the receiving power as the mean value and generate random deviations to the receiving power according to Gaussian distribution. The distance is then estimated using the receiving power with different degree of deviations.

Performance under different node moving speeds is also simulated to test the adaptivity of the algorithm. The random way-point mobility model is used to generate moving topologies.

5.7 Simulation Results and Analysis

The major purposes of the simulation study include

- Determining the appropriate values for key parameters;
- Examining the performance of the Smart algorithm.

We consider the following performance metrics:

- ***Broadcast Efficiency*** – how many redundant rebroadcasts Smart scheme can save (represented by percentage over simple flooding);
- ***Reachability*** – how many nodes Smart scheme can cover (represented by percentage over simple flooding);
- ***Latency*** – will directional backoff scheme suffer from longer delay because of the network layer backoff?

- ***Energy Efficiency*** – how much energy Smart can save by eliminating unnecessary rebroadcast;
- ***Mobility*** – will Smart scheme perform well with node movement?

In this section, we start with the performance analysis of Smart algorithm using pure free space model, assuming the receiving power can provide the exact distance. The performance under the same scenario with distance estimated from Gaussian-distributed receiving power is then examined to see whether the algorithm is robust. In the last part, performance with node mobility is simulated to test the algorithm's adaptivity in a dynamic environment.

5.7.1 Results of Using Pure Free Space Propagation Model

First, we examine our Smart approach without the reachability enhancement algorithms.

Figure 30 shows the efficiency and reachability of basic Smart scheme without reachability enhancement algorithms. The first observation is that our Smart scheme is highly efficient, especially when the node density is high. In the map size of 1x1, Smart scheme saves up to 97% of the rebroadcast traffic over simple flooding.

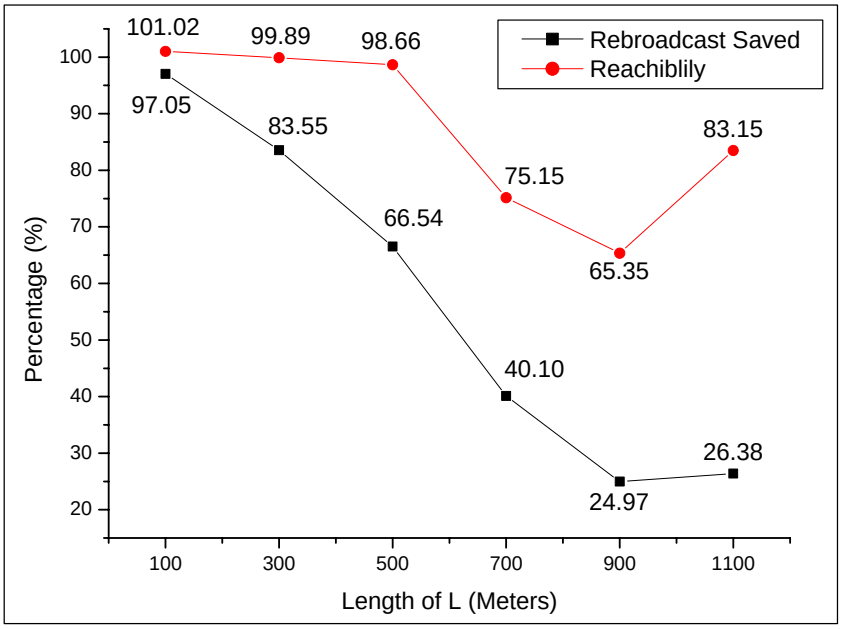

Figure 30 - The Efficiency and Reachability without RE for Different Network Densities

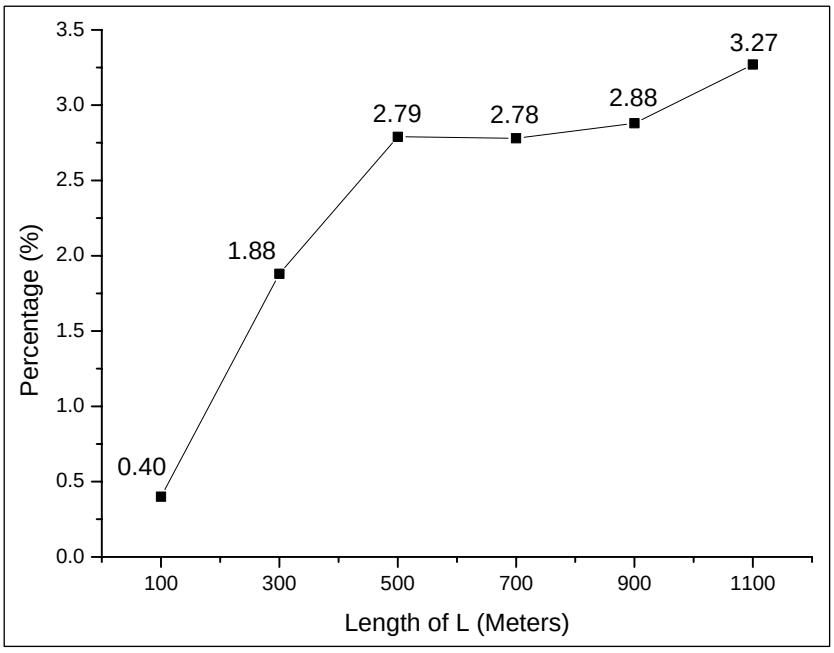

Figure 31 - Number of MAC Layer Backoffs – Smart vs. Simple Flooding

Please note that with the map size of 1x1 (a highly dense network), the reachability of our scheme is even higher than simple flooding. This indicates that when the network is getting congested, simple flooding may degrade the channel accessibility, which is the basic requirement of quality of service. The reason of this "unexpected" result can be explained by **Figure 31**. This figure reveals that in the case of 1x1 map size, the number of backoffs happen in the MAC layer using the Smart scheme is only less than .5% of that in the simple flooding case.

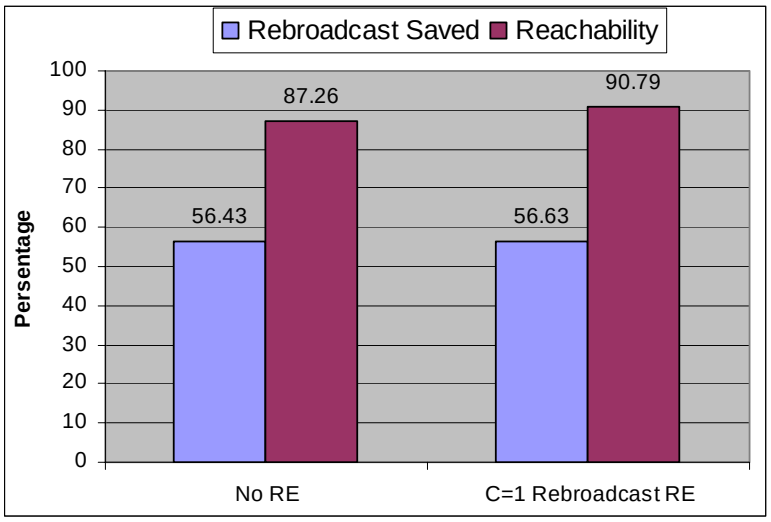

Figure 32 - The Overall Efficiency and Reachability (C=0 and C=1)

The second observation is, as expected, the basic Smart scheme cannot provide satisfactory reachability for sparse networks. The averaged overall reachability among different network density is 87.26% (**Figure 32**, No RE). **Figure 32** also shows the result when a node rebroadcasts the packet with only one rebroadcast heard (*C=1*). The reachability is about 90.79%, 3.5 percent higher than the algorithm without RE. This result indicates setting the counter threshold to 1 is not enough.

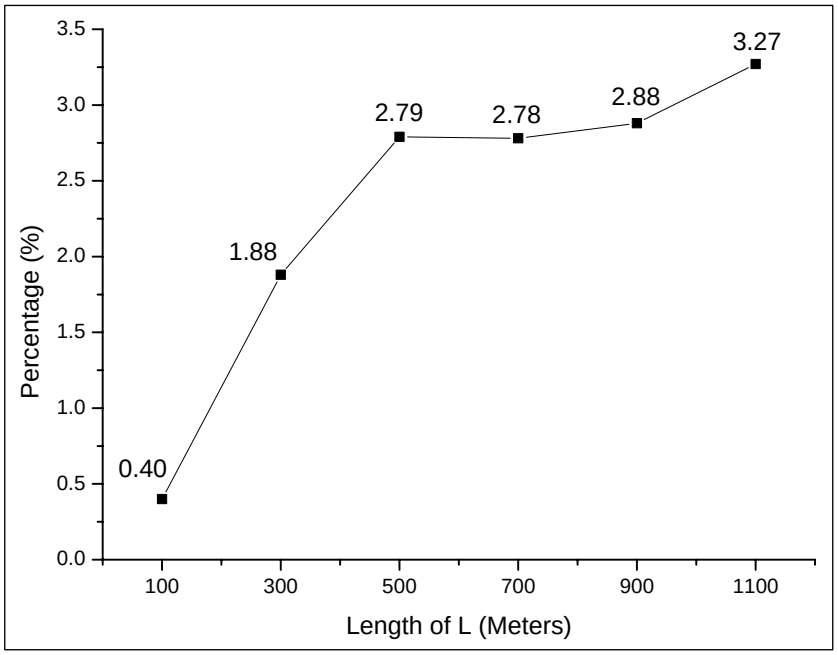

Figure 33 - The Overall Efficiency and Reachability with RE (C=2)

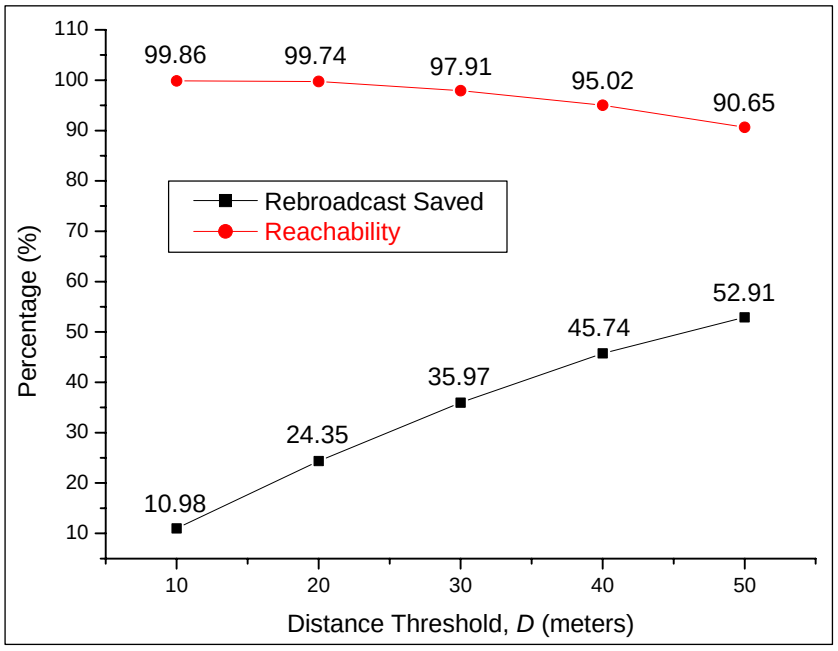

Figure 34 - The Overall Efficiency and Reachability of Distance-Based Scheme

Figure 33 shows the averaged reachability and efficiency for different distance thresholds when *C=2*. It shows that even with the largest threshold, *D=90* meters (which means the node that is making the rebroadcast decision is at least 90 meters away from the original sending node, and its two rebroadcast neighbors are also at least 90 meters away from it), the algorithm is able to provide close to 98% reachability comparing with simple flooding. When *D=50* (50% of *R*), the reachability goes as high as 99.04% with only 1.5% decrease in efficiency (percentage of Rebroadcast Saved). This result illustrates that Smart scheme is not sensitive to the threshold *D*. The difference in efficiency and reachability has little change when *D* varies. **Figure 34** shows the simulation result of Distance-based scheme. What can be easily observed is that this scheme fails to balance between efficiency and reachability. In order to achieve the same efficiency as Smart scheme, it has to sacrifice reachability, and vice versa. Another deficiency of this scheme is that its performance is too sensitive to the setting of the distance threshold.

One concern when designing the directional backoff in Smart scheme is that it may lead to extra delay because of the backoffs we introduced in BCL. However, the simulation result shown in **Figure 35** eliminates this worry. It shows that in average Smart algorithm does not incur longer delay (comparing to simple flooding) even thought there is additional backoff above MAC layer. The reason is that by doing directional backoff above MAC layer, The number of MAC layer backoffs can be saved up to 96.7% (11x11) ~ 99.6% (1x1) for different node densities (as shown in **Figure 31**). Therefore, our algorithm actually shortens the delay in dense networks. In the extremely sparse networks, Smart algorithm may generate longer delay than simple flooding, like the

1100x1100 case shown in **Figure 35**. The reason is that each node has only about 2 neighbors in the network. The possibility that there are nodes close to the transmission border is lower than that in dense networks. The resultant actual backoff time is increased.

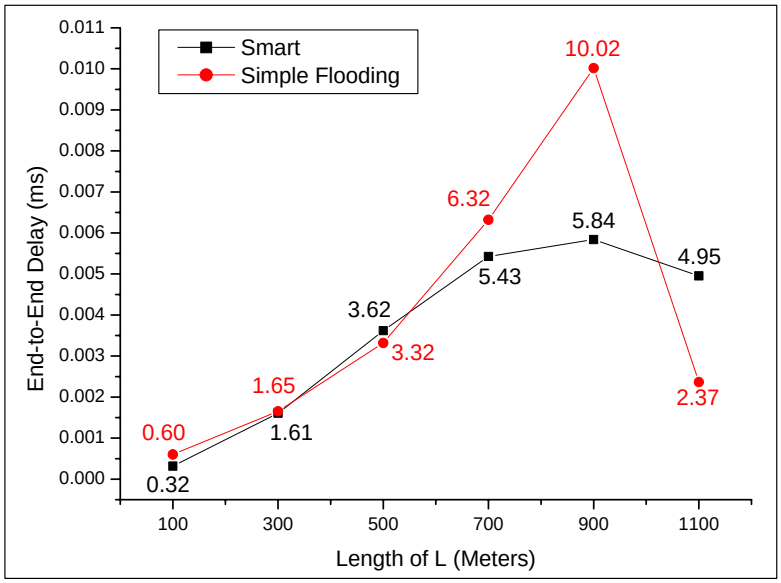

Figure 35 - ETE delay – Our Scheme vs. Simple Flooding

Figure 36 shows that Smart scheme consistently generates less hop counts than pure flooding, especially when the network becomes sparse. This is a result of directional backoff since it always finds the node closer to the border. When hop count is the route selecting metrics of the ad hoc routing protocols, our scheme provides the lower layer support. With random backoff, there is not guarantee the route found by flooding the route request is the shortest (in hops).

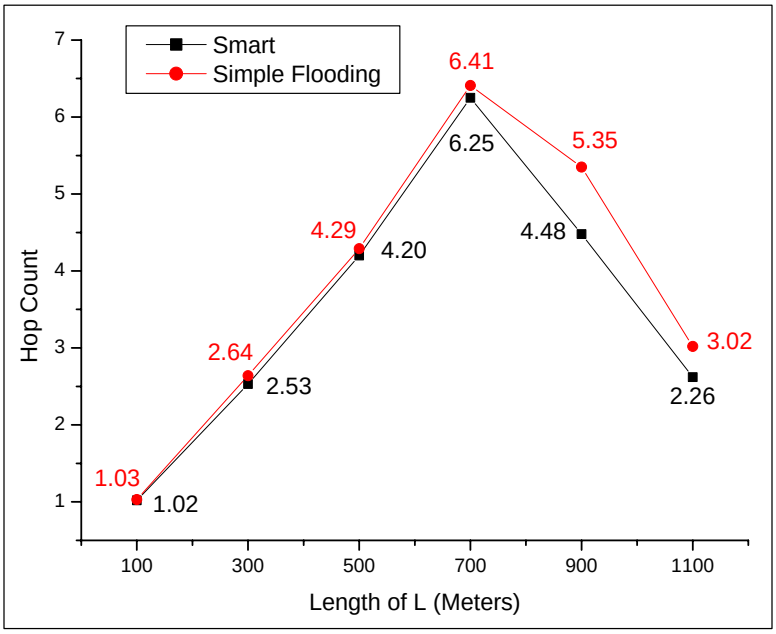

Figure 36 - Average Hop Count- Smart vs. Simple Flooding

In order to examine the power consumption of studied algorithms, we take the power model defined by the popular 2.4GHz DSSS Lucent IEEE 802.11 WaveLAN PC card. This card consumes about 330 mA in transmitting mode, and 250 mA in receiving mode. Given the packet length in bits, we can calculate power consumed by the transmitters and receivers. For example, to transmit a 560-bit MAC layer frame, which is the packet size we used in our simulation, the energy consumption is

$$5V \times 300mA \times 560/1000000 = 0.84 \text{ Joules}$$

Here we ignore the power consumed in idle mode because it does not contribute to the power consumption of packet transmission.

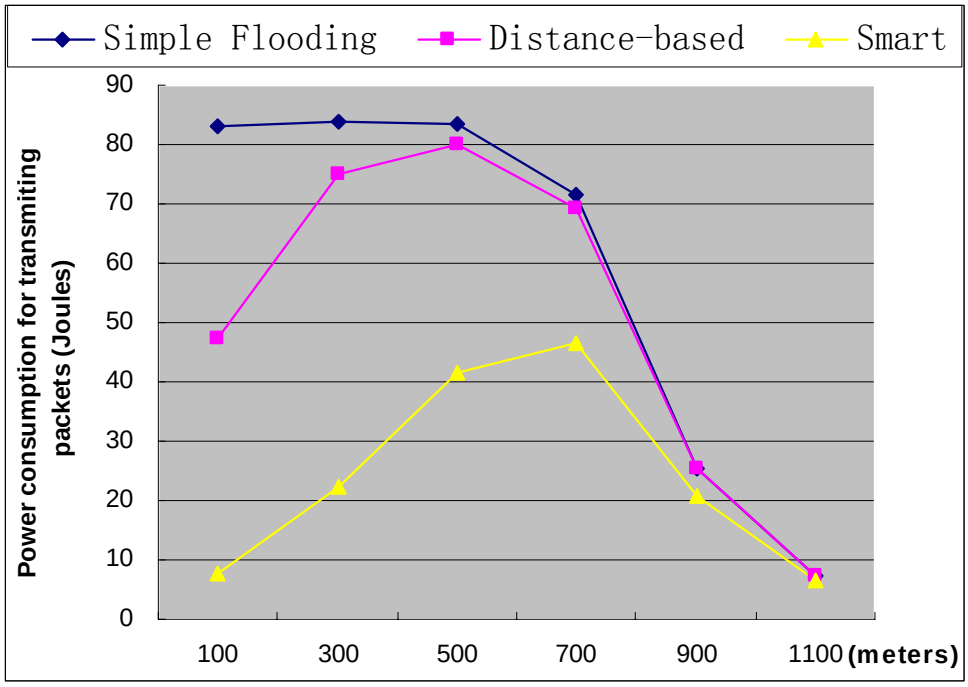

Figure 37 - Transmission power - Simple Flooding vs. Smart (*D=50*) vs. Distance-based scheme (*D=10*)

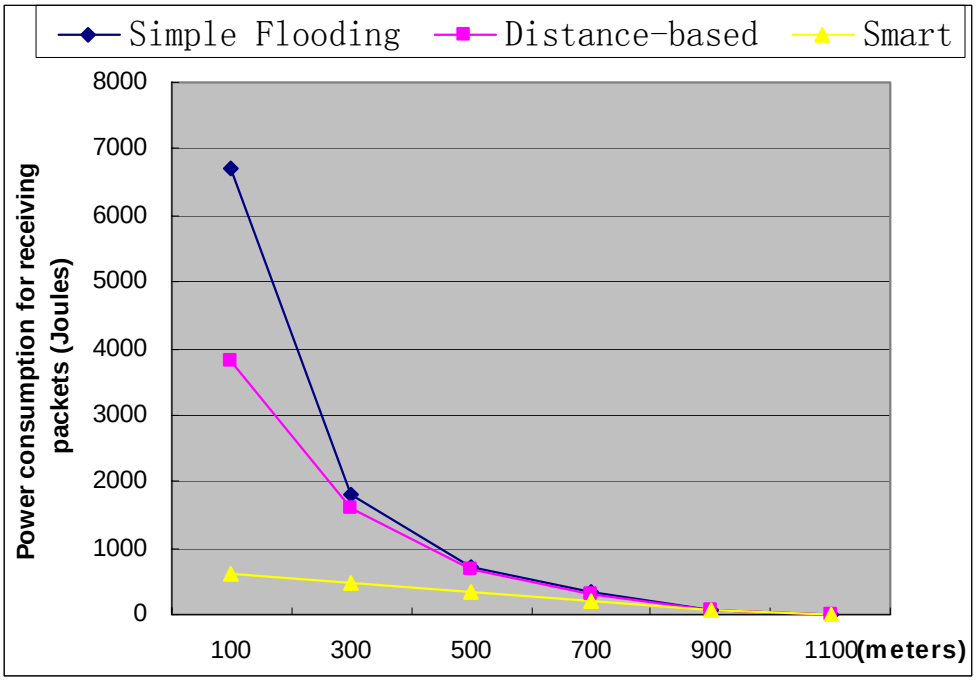

Figure 38 - Receiving power - Simple Flooding vs. Smart (*D=50*) vs. Distance-based scheme (*D=10*)

The power consumed for transmitting and receiving packets of all nodes in the network are shown in **Figure 37** and **Figure 38** respectively. Several observations can be made from these two figures. First, we show again that Smart approach adapts to the network density very well. When the node density is high (100x100 case), the transmitting power of Smart is only 9% of that of simple flooding and 16% of that of Distance-base scheme. When the network becomes sparser, more and more nodes participate in the packet forwarding to ensure the reachability for both Smart and Distance-based schemes. The curves of all there schemes hence get closer. When the area is larger than 700x700, the network becomes partitioned and the number of total transmissions decreases, so does the transmitting power.

Second, the total power consumed by receiving is much larger than the total power consumed by transmitting. This is true for all three algorithms. The result is led by the fact that when a node broadcast, all of its neighbors will receive the packet and hence consume receiving power. The more neighbors a node has, the more energy they consume in total. This is why the total receiving power goes so high when the network is dense (100x100 and 300x300 cases). The observation implies that simple flooding must be avoided in a battery-powered network, which is the typical case of wireless sensor network.

Overall, we observed that when the network becomes denser, the power consumption of Smart increases linearly while that of simple flooding and Distance-based approaches increase exponentially.

5.7.2 Results of Fluctuating Channel

As explained in Section 5.5, the actual radio channel is extremely random and hence free space model is not accurate. To simulate the performance under the real channel condition, we calculate the receiving power using free space model and use it as the mean value of the receiving power for each distance. We then generate some random variances to the mean value according to Gaussian distribution. The distance is then calculated using this variated power. The detail implement is explained below.

The Wireless LAN model comes with OPNET has a parameter call "Wireless LAN Range". Receivers outside this range will not receive the packets from the sender. To create the inaccuracy in distance estimation, we modified this mechanism. The following describes the procedure.

First, when a node broadcasts a packet, all other nodes in the network will receive the packet (there is not range limit). The receivers calculate the receiving power using free space model.

Note this calculation is done in the physical layer of the model. The exact distance is used for this calculation since the simulation kernel knows the X, Y, Z coordinates of each node. We consider this calculated power, *Pr*, the mean value of the receiving power at this distance (certainly there is a different mean value for each different distance).

The next thing is to add deviations to the mean value *Pr* obtained above. To test the robustness of our algorithm, we generate Gaussian-distributed power, *Pr'*, using different

levels of standard deviations. The levels are 10%, 20% and 30% of *Pr* calculated in the first step.

It is well known that standard deviation is a measure of spread of Gaussian distribution. In a Gaussian distribution, about 99% of the scores are within three standards deviations of the mean. To avoid generating some extreme or invalid values, e.g. power less than zero or power several times larger than the transmission power, we set a bound to the generated *Pr'*. Therefore, the final varied receiving power will be bounded by the following equation.

$$Pr * (1-3\sigma) < Pr' < Pr * (1+3\sigma) \tag{13}$$

where σ is the standard deviation of the mean value of the receiving power *Pr*. Because we consider bounds $\pm 3\sigma$ away from the mean value for each different σ, 99% of the distribution will be tested. The pdf of Gaussian distribution and its illustration is given in equation (14) and **Figure 39**.

$$f(X) = \frac{\exp\left[-\frac{1}{2}\left(\frac{X-\mu}{\sigma}\right)^2\right]}{\sigma\sqrt{2\pi}} \tag{14}$$

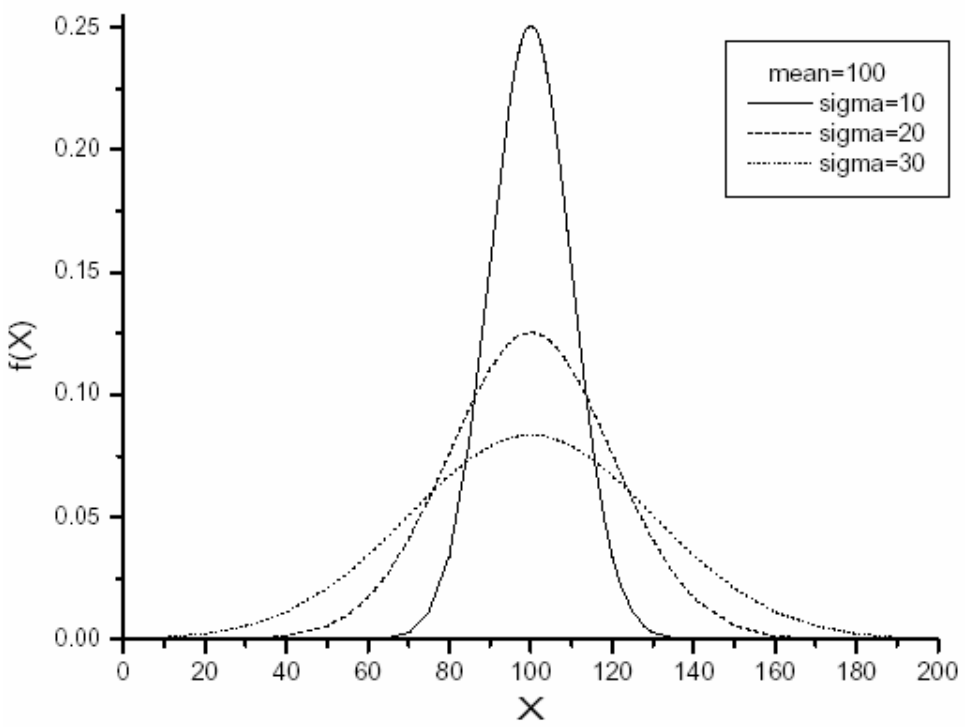

Figure 39 - pdf of Gaussian Distribution with Standard Deviations Equal to 10%, 20% and 30% of the Mean (100)

The third step is to determine how many receivers are still within the communication range after we vary the receiving power. This is done by calculating the estimated distance using the varied receiving power *Pr'*. If the estimated distance is within the communication range (100 meters in our simulation), the received packet will then be eligible for further process; otherwise, it will be discarded.

If a packet passes all the processes in physical and MAC layer, it will finally arrive at the network layer or the Broadcast Control Layer defined earlier, containing the information of the artificially varied power *Pr'*.

In the fourth step, the Smart broadcast algorithm will work with *Pr'* sent from the physical layer to estimate the distance again. This is because we don't assume the

network layer can get the distance information from the lower layers. This estimated distance certainly will be equal to the distance calculated in the physical layer described in Step 3 and is not the real distance between the sender and the receiver. So far, the inaccurate distance estimation is done and we are ready to test the robustness of our approach.

As we mentioned above, we tested the Gaussian-distributed power at three levels of standard deviations, 10%, 20% and 30% of the mean value of the receiving power.

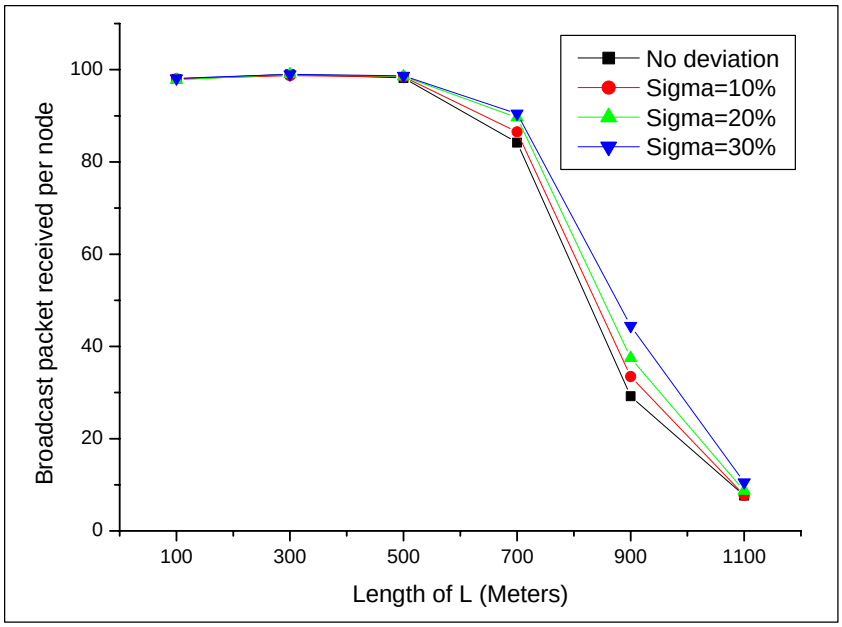

Figure 40 - Simple Flooding – Average Number of Packets Received at Each Node at Different Standard Deviation of Gaussian Distribution of Receiving Power

Figure 40 shows us the result of simple flooding at different levels of standard deviations. The Y-coordinate represents the number of packets a node received in average during the simulation period. Since there are 100 nodes in the network and every node broadcasts exactly once to the network, a node will receive 99 broadcast packets

from other nodes if all the transmissions are successful and all nodes are physically connected by one or more hops. Note the data marked by "No deviation" represents the situation of free space propagation. The observation from this figure is that the packet delivery rate increases with the standard deviations. The simple flooding with highest standard deviation gets the highest coverage. This is because the relationship between the receiving power and the distance is not linear. As shown in (11), $d=sqrt(P_tC_2/P_r)C_1$. Since P_t, C_1 and C_2 are all constants. This equation can be rewritten as $d=C/sqrt(P_r)$, where C is a constant representing the combined effect of P_t, C_1 and C_2. Therefore, when Gaussian distributed power is converted to distance, the deviations are discounted. In addition, the distances are under-estimated, not over-estimated. In other words, more nodes are "pulled" into the communication range than those are "pushed" out. Because the number of nodes participating in the broadcast increases, the overall coverage also increases.

When σ equals to 30% of the mean, the distribution has the widest spread. In other words, more nodes have large deviations than in other cases. Since the distances are under-estimated, more nodes are "pulled" into the communication range and hence higher packet delivery rate.

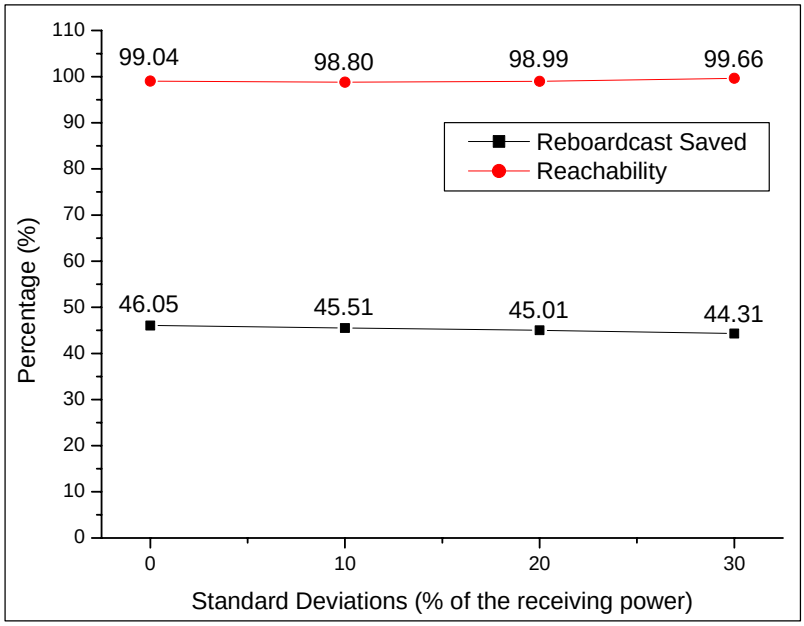

Figure 41 - Smart Broadcast – Efficiency and Reachability at Different Standard Deviation of Gaussian Distribution of Receiving Power

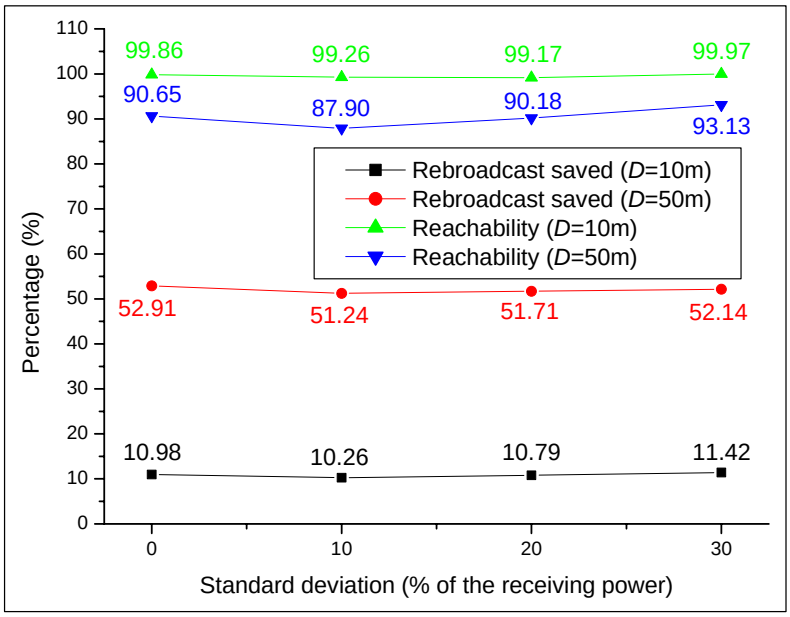

Figure 42 - Distance-Based Scheme – Efficiency and Reachability at Different Standard Deviation of Gaussian Distribution of Receiving Power

Figure 41 and **Figure 42** present the performance of Smart broadcast and Distance-based scheme. In **Figure 41**, X=0 means pure free space model; distance threshold D=50m; both rebroadcast packets received are far. Note all the numbers are compared with that of the corresponding simple flooding scenarios. For example, the performance of Smart broadcast at standard deviation 10% of the receiving power is compared with the simple flooding with the same 10% standard deviation.

The simulation results in these two figures reveal that our Smart approach is really robust when the distance estimation is not accurate. In the worse case, the coverage is 0.74% (σ=10%) less than that of simple flooding. In the case of σ=30%, Smart broadcast is even better than the simple flooding by 0.62%. At the same time, our algorithm still maintains the efficiency.

The Distance-base scheme also survives well in the random channel environment except it still suffers from the balance between the efficiency and the reachability.

5.7.3 Impact of node mobility

To test the performance of the Smart approach in a dynamic environment, we introduced the way-point mobility model to our simulation scenario. The speeds tested are 0, 4, 8, 12, 16 and 20 meters per second. The distance threshold *D* is set to 50 and the counter threshold *C* is set to 2 for all speeds.

As expected, mobility does not have impact to Smart algorithm. This is because Smart algorithm does not use any mobility-related information. The changing network

topology due to node movement has the similar effect as we create new simulation scenarios using changing seeds. The simulation results with D = 50m are illustrated in **Figure 43**.

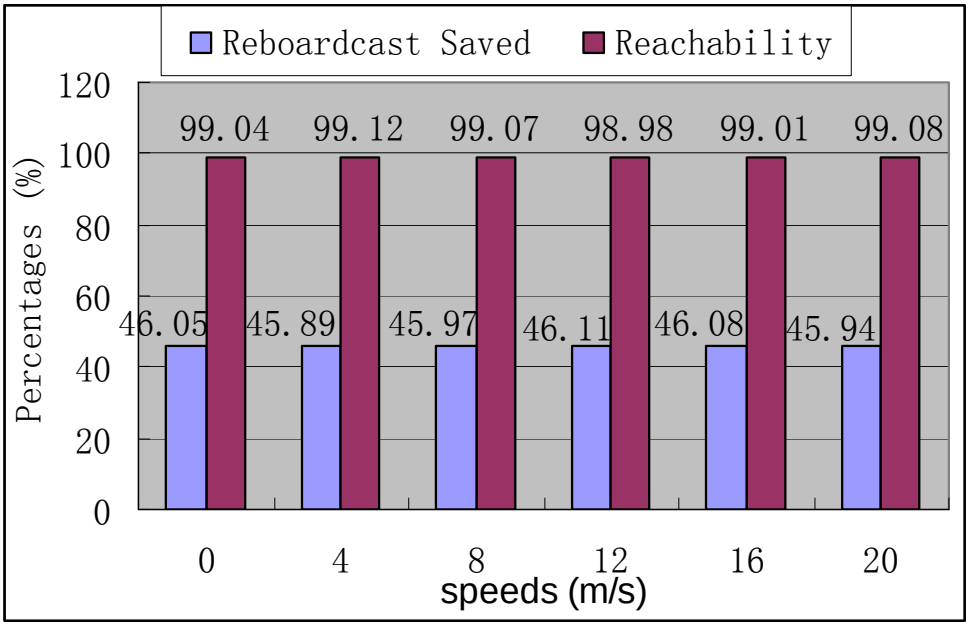

Figure 43 - Performance of Smart Broadcast under Different Moving Speeds

5.8 Discussions on Link Quality

One of the important applications of the Smart broadcast algorithm in a variable topology network is to discover routes for unicast routing protocols, e.g. AODV and etc. It is a fact that Smart algorithm always finds the nodes at the transmission border of the sender as next hop toward the destination. This can be verified in **Figure 36** which shows our algorithm achieves smaller hop counts when broadcasting a packet to the entire network. Because the receiving powers at some nodes may be very close to the threshold of minimal acceptable power, one concern here is whether an end-to-end route established by Smart algorithm is stable.

One simple solution to this problem is raising the threshold of minimal power we use to calculate the backoff level. For instance, let's assume we originally use 100mW as transmission power and 1mW as the threshold of minimal acceptable receiving power for backoff level calculation. In this case, a node receiving a packet with less than 1mW of receiving power will not participate in the rebroadcast business. In order to find a stable route, nodes can raise the minimal acceptable receiving power to, for example, 5mW (even though the real acceptable power at MAC/PHY is 1mW). Actually, it depends on the routing metrics of the unicast routing protocol. Some of them prefer the least number of hops and others prefer the best quality of links. When the metrics is hop count, our algorithm can always find the best route.

Please note if Smart algorithm is used to simply broadcast data packets, it is not necessary to raise the minimal acceptable receiving power. The fact that the packet can be received correctly means the link quality is good enough to receive the packet. Here the goal is to propagate the packet further and the quality of the link to the previous hop is not a concern.

5.9 Summary

In this chapter, we analyze the *Broadcast Storm* problem in variable topology networks and show the *Broadcast Storm* may greatly degrade the quality of service in these networks. We also develop a Smart broadcast algorithm to deal with the problem. Instead of using popular random backoff when differentiating the rebroadcast time among receiving nodes, we propose a "*directional backoff*" scheme which knows the direction of the packet propagation. A concept called "*relative position*" is introduced to

help nodes determine its position among rebroadcast nodes and whether it should rebroadcast the packet. Comparing to other existing methods, Smart algorithm is simple yet efficient. It is a fully distributed algorithm that does not depend on any topology information. It introduces no extra control overhead and is able to adapt to dynamic network situations. Unlike distance-based scheme that has to sacrifice one goal to achieve the other, Smart algorithm accomplishes both efficiency and reachability without compromising between them.

We tested Smart algorithm under the situation where the distance can not be estimated accurately. The simulation results show that our algorithm survives without degrading the performance. Easy to implement is another virtue of our algorithm. For example, as shown by the simulation result, the setting of the distance threshold $D=R/2$ can adapt to different network densities and provide satisfactory efficiency (46% saved) and reachability (99%). In addition, the values of the algorithm parameters are not sensitive to network scenario. We believe these features are desired when it comes to real implementation.

Speaking of the energy efficiency, Smart can save up to 91% of the total energy by reducing the unnecessary rebroadcast in the highest network density case. We also reveal that it is the receiving, not the transmitting of packet, consumes the most power when broadcasting.

6 CONCLUSIONS AND FUTURE WORKS

In this work, we focused on the studies of multipoint communications, including multicasting and efficient broadcasting, in wireless ad hoc networks.

We first examined the performance of multicast AODV. We pointed out some ambiguity points of the protocol and enhanced the protocol in order to make it work well. We found that MAODV is not going to work if there is an idle time during the multicast session and we fixed the problem without making big modification to the protocol. A dynamic method of calculating *route_discovery_timeout* is introduced and proved by the simulations to be about 100 times better than the original one. A new packet type MACT_ACK and the mechanism of MACT_ACK_TIMEOUT is used to solve the problem of MACT lost. Our simulations also show that MAODV has some unsolved problems when the control messages are lost or links are broken.

In the second part of the work, we further explored the problem of route discovery latency we observed when conducting the first part of the research. To avoid the "blind" estimation of the route discovery time, we developed an algorithm which measures the round trip time of the first route request/reply pair and use it as a reference to measure the quality of later received better routes. Our algorithm waits at most three RTTs (if a better route is received) or less (if a better route is not received) to determine the best route to the destination. Our simulation results show that the primary reason for route discovery delay is the backoff time of the contention led by the flooding of route requests. This delay does not always grow as the distance between the source and the destination grows.

By comparing with one of the reactive routing protocols, we show our algorithm is much better than the "blind estimation" algorithms and can adapt to different network densities.

In the third part of the work, we developed a new hybrid multicast routing protocol for wireless sensor networks. The protocol is built on top of a hybrid unicast routing structure that combines a routing tree and local link state information. It hence utilizes the advantages from both methods; by using the logical tree, control frames for joining and leaving a multicast tree can be unicast, instead of being broadcast; by using the local link state information, the inefficiency of tree routes can be partially compensated and tree repaired issues can be solved. In addition, minimal level of control overhead is achieved because existing control messages designed for unicast protocol can be used for multicast protocol too.

In the last part of the work, we analyze the Broadcast Storm problem in variable topology networks and show the Broadcast Storm may greatly degrade the quality of service in these networks. The Broadcast Storm problem is also the direct cause of the route discovery latency we discussed in Part Two of our work. We developed a Smart broadcast algorithm to deal with the problem. Instead of using de facto random backoff when differentiating the rebroadcast time among receiving nodes, we propose a "*directional backoff*" scheme which knows the direction of the packet propagation. A concept called "*relative position*" is introduced to help nodes determine its position among rebroadcast nodes and whether it should rebroadcast the packet. Comparing to other existing methods, Smart algorithm is simple yet efficient. It is a fully distributed algorithm that does not depend on any topology information. It introduces no extra

control overhead and is able to adapt to dynamic network situations. In addition, Smart algorithm accomplishes both efficiency and reachability without compromising between them. We also tested Smart algorithm under the situation where the distance can not be estimated accurately. The simulation results show that our algorithm survives without degrading the performance. Easy to implement is another virtue of our algorithm. Speaking of the energy efficiency, Smart can save up to 91% of the total energy by reducing the unnecessary rebroadcast in the highest network density case. We also reveal that it is the receiving, not the transmitting of packet, consumes the most power when broadcasting.

Further improvements and extensions to the works included in this dissertation can be made in the following aspects.

- For Mulitcast AODV Internet draft, we can suggest to the authors to consider and adopt our improvements and additions to their protocol. In fact although AODV is widely used, MAODV has not been seen adopted in any real applications. We believe our work has revealed at least some of the reasons, if not all. Adopting our improvements and modifications to MAODV could make it usable and more efficient.
- For RTT-based waiting time for route discovery, we can test it with more routing protocols, such as DSR and TORA, to evaluate its performance. We can also extend the algorithm to cover the destination selection approaches.
- For hybrid multicast routing protocol, we shall examine its performance either by theoretical analysis or by simulation. In addition, because up to *N*-hop

neighbor information is available, we can further develop an efficient broadcast scheme based on the neighbor information. Based on the neighbor information, a set of nodes can be elected as members in a dominant set of nodes that can reach all nodes in the network. Efficient broadcast can then be achieved by letting only the nodes of the dominant set rebroadcast the packets.

- For the Smart broadcast algorithm, we can further prove its performance by theoretical analysis in addition to simulation.

7 REFERENCES

[1] Elizabeth M. Royer, Charles E. Perkins “Multicast Ad hoc On-Demand Distance Vector (MAODV) Routing”, Internet Draft, draft-ietf-manet-maodv-00.txt

[2] Elizabeth M. Royer, Charles E. Perkins and Samir R. Das “Ad hoc On-Demand Distance Vector (AODV) Routing”, Internet Draft, draft-ietf-manet-aodv-08.txt

[3] Elizabeth M. Royer, Charles E. Perkins “Multicast Operation of the Ad-hoc On-Demand Distance Vector Routing Protocol”, in Proc. of ACM/IEEE Intl. Conference on Mobile Computing and Networking (MOBICOM), pp. 207-218, Aug. 1999

[4] C.W. Wu, Y.C. Tay, “AMRIS: a Multicast Protocol for Ad Hoc Wireless Networks”, Proceedings IEEE MILCOM’99, Atlantic City, Nov. 1999

[5] M.S. Corson, S.G. Batsell, “A Reservation Based Multicast (RBM) Routing Protocol for Mobile Networks: Initial Route Construction Phase”, ACM/Baltzer Wireless Networks, pages 1(4): 427-450, 1995

[6] L. Ji, M.S. Corson, “A Lightweight Adaptive Multicast Algorithm”, Proceedings IEEE GLOBECOM’98, Sydney, pages 1036-1042, Nov. 1998

[7] E. Bommaiah, M. Liu, A. McAuley, R. Talpade, “AMRoute: Ad-Hoc Multicast Routing Protocol”, Internet Draft, Aug. 1998

[8] S.J. Lee, M. Gerla, C.C. Chiang, “On Demand Multicast Routing Protocol”, Proceedings of IEEE WCNC’99, New Orleans, pages 1298-1302, Sept 1999

[9] J.J. Garcia-Luna-Aceves, E.L. Madruga, “A Multicast Routing Protocol for Ad-Hoc Networks”, Proceedings of IEEE INFOCOM’99, New York, pages 784-792, March 1999

[10] S. Lee, C. Kim, “Neighbor Supporting Ad Hoc Multicast Routing Protocol”, Proceedings of First Annual Workshop on Mobile Ad Hoc Network & Computing, MobiHOC 2000, Boston, pages 37-50, August 2000

[11] David B. Johnson, Davis A. Maltz, "Dynamic Source Routing in Ad Hoc Networks", Mobile Computing, T. Imielinski and H. Korth, Eds., Kulwer, 1996, pp. 152-81.

[12] David B. Johnson, David A. Maltz, and Yih-Chun Hu, “The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR),” draft-ietf-manet-dsr-10.txt, *IETF MANET Working Group*, July 2004

[13] VD Park and MS Corson "A highly adaptive distributed routing algorithm for mobile wireless networks", Proc. INFOCOM'97, Apr. 1997

[14] Chai-Keong Toh, "A novel distributed routing protocol to support Ad hoc mobile computing" Proc. 1996 IEEE 15th Annual Int'l. Phoenix Conference Computer and Communication, Mar. 1996,

[15] Chai-Keong Toh, "Long-lived Ad-Hoc Routing based on the concept of Associativity" IETF Draft, March 1999

[16] R. Dube et al., "Signal Stability based adaptive routing for Ad Hoc Mobile networks", IEEE Personal Communications, Feb. 1997, pp. 36-45.

[17] G. Aggelou and R. Tafazolli "RDMAR: A Bandwidth-efficient Routing Protocol for Mobile Ad Hoc Networks" , In Proceedings of The Second ACM International Workshop on Wireless Mobile Multimedia (WoWMoM), Seattle, Washington, August 20, 1999.

[18] C. Zhu, M.J. Lee and T. Saadawi, “A Border-aware Broadcast Scheme for Wireless Ad Hoc Network”, Proceeding of IEEE Consumer Communications and Networking Conference 2004, Las Vegas, Nevada, Jan. 5-8, 2004

[19] B. Williams, T. Camp, “Comparison of Broadcasting Techniques for mobile Ad Hoc Networks,” MOBIHOC’02, EPFL, Lausanne, Switzerland, June 9-11, 2002,

[20] S. Ni, Y. Tseng, Y. Chen, J. Sheu, “The Broadcast Storm Problem in a Mobile Ad Hoc Network,” MobiCom99’, Seattle, Washington, Aug. 15-20 1999

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

[22] J. Jetcheva, Y. Hu, D. Maltz, and D. Johnson. “A simple protocol for multicast and broadcast in mobile ad hoc networks”. Internet Draft: draft-ietf-manet-simple-mbcast-01.txt, July 2001.

[23] H. Lim and C. Kim. “Multicast tree construction and flooding in wireless ad hoc networks”. In Proceedings of the ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM), 2000.

[24] W. Peng and X. Lu. “On the reduction of broadcast redundancy in mobile ad hoc networks”. In *Proceedings of MOBIHOC*, 2000.

[25] Qayyum, L. Viennot, and A. Laouiti. “Multipoint relaying: An efficient technique for flooding in mobile wireless networks”. Technical Report 3898, INRIA - Rapport de recherche, 2000.

[26] W. Peng and X. Lu. “AHBP: An efficient broadcast protocol for mobile ad hoc networks”. Journal of Science and Technology - Beijing, China, 2002.

[27] W. Peng and X. Lu. “Efficient broadcast in mobile ad hoc networks using connected dominating sets”. Journal of Software - Beijing, China, 1999.

[28] J. Sucec and I. Marsic. “An efficient distributed network-wide broadcast algorithm for mobile ad hoc networks”. CAIP Technical Report 248 - Rutgers University, September 2000.

[29] IEEE 802.11 Standard, “Wireless LAN medium access control (MAC) and physical layer (PHY) specifications”, ISBN 1-55937-935-9, 1997

[30] T. S. Rappaport, “Wireless Communications”, 1996 by Prentice Hall PTR, ISBN 0-13-375536-3.

[31] C. Zhu, M.J. Lee, T. Saadawi, “On the Route Discovery Latency of Wireless Mesh Networks”, in the proceedings of IEEE Consumer Communication and Networking Conference 2005, Las Vegas, NV, Jan 3 – Jan 6, 2005

[32] C. Zhu, M.J. Lee, T. Saadawi, “A Smart Broadcast Scheme for Wireless Military Networks”, in the proceedings of IEEE Milcom 2004, Monterey, CA, Oct. 31 – Nov. 3, 2004

[33] Y. Liu, X. Hu, T. Kwon, C. Zhu, J. Zheng, M.J. Lee, “An Integrated Routing Protocol for Wireless Sensor Networks”, Special Issue of IEEE Wireless Communications on Wireless Sensor Networks: “Theory and Systems"

[34] C. Zhu, M.J. Lee and T. Saadawi, “RTT-based Optimal Waiting Time for the Best Route Selection in Ad Hoc Routing Protocols”, Proceeding of IEEE Milcom 2003, Military Communication Conference, Boston, MA, Oct. 13-16, 2003

[35] C. Zhu, M.J. Lee and T. Saadawi, "Multicast AODV - Problems and Improvements", Communications and Networks Conference 2003 Proceedings, Collaborative Technology Alliances, page 289-293, Maryland, MD, April 29-May 1, 2003

[36] C. Zhu, M.J. Lee, T Saadawi, "Simulation Study of Multicast AODV Protocol", Proceeding of OPNETWORK 2002, Aug. 26th-30th, 2002

[37] Jochen H. Schiller, "Mobile Communications," Addison Wesley, 2000

[38] Elizabeth M. Royer, Chai-Keong Toh, "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks," *IEEE Personal Communications*, Vol. 6, No. 2, pp. 46-55, April 1999

[39] Tarek N. Saadawi, Mostafa H. Ammar, and Ahmed El Hakeem, "Fundamentals of Telecommunication Networks," John Wiley & Sons Inc., 1994

[40] Andrew S. Tanenbaum, "Computer Networks," Prentice Hall, 2003

[41] C. E. Perkins and P. Bhagwat. "Highly dynamic Destination Sequenced Distance-Vector routing (DSDV) for mobile computers," in *Proceedings of the SIGCOMM'94*, pages 234-244, August 1994

[42] P. Jacquet *et al.*, "Optimized Link State Routing Protocol," draft-ietf-manetolsr-05.txt, Internet Draft, *IETF MANET Working Group*, Nov. 2000

[43] G. Pei, M. Gerla, and T.-W. Chen, "Fisheye State Routing: A Routing Scheme for Ad Hoc Wireless Networks," in *Proceedings of the ICC'2000*, New Orleans, LA, June 2000

[44] R. G. Ogier et al., "Topology Broadcast based on Reverse-Path Forwarding (TBRPF)," draft-ietf-manet-tbrpf-05.txt, *IETF MANET Working Group*, Mar. 2002

[45] I. Chakeres and L. Klein-Berndt, "AODVjr, AODV Simplified," *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R)*, Vol. 6, No. 3, July 2002

[46] Z. J. Haas and M. R. Pearlman, "The Zone Routing Protocol for Ad Hoc Networks," draft-ietf-manet-zone-zrp-02.txt, *IETF MANET Working Group*, June 1999

[47] F. Ye, H. Luo, J. Cheng, S. Lu and L. Zhang, "A Two-tier Data Dissemination Model for Large-scale Wireless Sensor Networks," in *Proceedings of the MOBICOM'2002*, Atlanta, September 2002

[48] Hakim Badis, Khaldoun Al Agha and Anelise Munaretto, "Quality of Service for Ad hoc Optimized Link State Routing Protocol (QOLSR)", IETF draft, draft-badis-manet-qolsr-00.txt, Washington, DC, USA, November 2004

[49] C. -K. Toh, "Maximum Battery life Routing to Support Ubiquitous Mobile Computing in Wireless Ad Hoc Networks," *IEEE Communication Magazine*, June 2001

[50] B. S. Manoj, R. Ananthapadmanabha, and C. Siva Ram Murthy, "Link life Based Routing Protocol for Ad hoc Wireless Networks," in *Proceedings of the 10th IEEE International Conference on Computer Communications 2001 (IC3N 2001),* October 2001

[51] R. S. Sisodia, B. S. Manoj, and C. Siva Ram Murthy, "A Preferred Link Based Routing Protocol for Ad Hoc Wireless Networks", *Journal of Communications and Networks*, Vol. 4, No. 1, March 2002, pp14-21

[52] Aggelou G, Tafazolli R. Rdmdr, "A Bandwidth-efficient Routing Protocol for Mobile Ad Hoc Networks," in *Proceedings of the ACM MobiCom99 /WoWMoM99*, August 1999

[53] R. Dube et al., "Signal Stability based adaptive routing for Ad Hoc Mobile networks", *IEEE Personal Communications*, Feb. 1997, pp.36-45

[54] D. Ganesan, R. Govindhan, S. Shenker, and D. Estrin, "Highly resilient, energy efficient multipath routing in wireless sensor networks," *Mobile Computing and Communications Review (MC2R)*, vol. 1, No. 2, 2002

[55] Ya Xu, John Heidemann, and Deborah Estrin, "Geography-informed energy conservation for ad hoc routing," in *Proceedings of the Mobicom'2001*, July 2001, pp.70-84

[56] S. Xu, T. Saadawi, "Does the IEEE 802.11 MAC protocol Work Well in Multi-hop Wireless Ad Hoc Networks?," *IEEE Communications Magazine*, Vol. 39, No. 6, pp. 130 - 137, June 2001

[57] Lakshmi Ramachandran, Manika Kapoor, Abhinanda Sarkar, Alok Aggarwal, "Clustering Algorithms for Wireless Ad Hoc Networks," in *Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications*, 2000

[58] S. Xu, T. Saadawi, "Revealing and Solving the TCP Instability Problem in 802.11 based Multi-hop Mobile Ad Hoc Networks," in *Proceedings of the IEEE VTC' 2001*, Vol. 1, pp. 257 –261, 2001

[59] L. Hester, Y. Huang, A. Allen, O. Andric, P. Chen, "neuRFon Netform: A Self-Organizing Wireless Sensor Network," in *Proceedings of the 11th IEEE ICCCN Conference*, Miami, Florida, Oct. 2002.

[60] Y. Huang, et al., "OPNET Simulation of A Multi-hop Self-organizing Wireless Sensor Network," in *Proceedings of the OPNETWORK 2002 conference*, Washington D.C., August 2002

[61] Y.-B. Ko and N.H. Vaidya, "Location-aided routing (LAR) in mobile ad hoc Networks," in *Proceedings of the MOBICOM'1998*, Dallas, Oct. 1998

[62] I. Stojmenovic and J. Wu, "Broadcasting and Activity-Scheduling in Ad Hoc Networks," *Ad Hoc Networking*, IEEE Press, 2004

[63] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," *ACM Wireless Networks*, Vol. 8, No. 2, March 2002.

[64] J. Cartigny, and D. Simplot, "Border node retransmission based probabilistic broadcast protocols in ad-Hoc networks," *Telecommunication Systems*, 22(1-4), 2003.

[65] M. Sun, W. Feng and T.H. Lai, "Location aided broadcast in wireless ad hoc Networks," in *Proceedings of the IEEE GLOBECOM 2001*, San Antonio, Texas, November 2001.

[66] A. Laouiti, A. Qayyum et L. Viennot, "Multipoint Relaying: An Efficient Technique for Flooding in Mobile Wireless Networks," in *Proceedings of the 35th Annual Hawaii International Conference on System Sciences,* Big Island, Hawaii, Jan., 2002

[67] I. Stojmenovic, M. Seddigh, J. Zunic, "Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, No. 1, January 2002

[68] R. Castenada, S. R. Das, and M. K. Marina, "Query localization techniques for on-demand routing protocols in ad hoc networks," *ACM/Kluwer Wireless Networks Journal*, Vol. 8, No. 2, March 2002

[69] Laurent Viennot, Philippe Jacquet, Thomas Heide Clausen, "Analyzing control traffic overhead versus mobility and data traffic activity in mobile Ad-Hoc network protocols," *Wireless Networks*, Volume 10, Issue 4, July 2004

[70] C. A. Santivanez, B. McDonald, I. Stavrakakis, and R. Ramanathan, "On the Scalability of Ad Hoc Routing Protocols," in *Proceedings of the IEEE INFOCOM 2002*, New York, 2002

[71] Tao Lin, Scott F. Midkiff and Jahng S. Park, "A Framework for Wireless Ad Hoc Routing Protocols," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, March 2003

[72] Ajay Chandra V. Gummalla and John O. Limb, "Wireless Medium Access Control Protocols," *IEEE Communications Surveys*, Second Quarter 2000

[73] F. A. Tobagi and L. Kleinrock, "Packet switching in radio channels: Part II – The Hidden terminal problem in carrier sense multiple-access and the busy-tone solution," *IEEE Trans. Communications*, vol. COM-23, pp 1417-1433, Dec. 1975

[74] P. Karn, "MACA – A new channel access method for packet radio," in *Proceedings of the ARRL/CRRL Amateur Radio 9th Computer Networking Conf.,* pp134 – 140,1990,

[75] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A media access protocol for wireless Lan's," in *Proceedings of the ACM SIGCOMM'1994*, pp. 212-225, London, Sept 1994

[76] C.L. Fuller and J.J. Garcia-Luna-Aceves, " Solutions to hidden terminal problems in wireless networks," in *Proceedings of the ACM SIGCOMM'97*, Cannes, French Riviera, France, Sept. 1997

[77] C.L. Fuller and J.J. Garcia-Luna-Aceves, "Floor acquistion multiple access (FAMA) for packet-radio networks," in *Proceedings of the ACM SIGCOMM'95*, pp. 262-273, Cambridge, Massachusetts, Aug. 1995

[78] Z. Hass, J. Deng, "Dual Busy Tone Multiple Access (DBTMA) – A multiple Access Control Scheme for Ad Hoc Networks," *IEEE Trans. on Communications*, Vol. 50, No. 6, June 2002

[79] R. Nelson and L. Kleinrock, "Spatial-TDMA: A collision-free multihop channel access control," *IEEE Trans. on Communications*, vol. 33, no. 9, Sept. 1985

[80] B. hajek and G. Sadaki, "Link shceduling in polynomial time," *IEEE Trans. Inform. Theory*, vol. 34, pp. 910-917, Sept. 1988

[81] L. Tassiulas and A. Ephremides, "Jointly optimal routing and scheduling in packet radio networks," *IEEE Trnas. Infom. Theory*, vol. 38, pp. 165-168, Jan. 1992

[82] A. Ephremides and T. V. Truong, "Scheduling broadcasts in multihop radio networks," *IEEE Trans. on Communications*, vol. 38, pp. 456–460, Apr. 1990

[83] R. Ramaswami and K. K. Parhi, "Distributed scheduling of broadcasts in a radio network," in *Proceedings of the IEEE INFOCOM'89*, pp. 497–504, Ottawa, Canada, April 1989

[84] C. X. Zhu and M. S. Corson, "A five-phase reservation protocol (FPRP) for mobile ad hoc networks," in *Proceedings of the IEEE INFOCOM'1998*, San Francisco, CA, March/April, 1998

[85] Z. Tang and J.J. Garcia-Luna-Aceves, "Hop-Reservation Multiple Access (HRMA) for Ad Hoc Networks," in *Proceedings of the INFOCOM'1999*, New York, March 1999

[86] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie, "Protocols for Self-Organization of a Wireless Sensor Network," *IEEE Personal Communications*, October 2000.

[87] IEEE Standard for Information Technology-Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low Rate Wireless Personal Area Networks (LR-WPANS), 2003

[88] Jianliang Zheng, Myung Lee, "Will IEEE 802.15.4 make ubiquitous networking a reality?," *IEEE Communications Magazine,* vol. 42, no. 6, pp. 140 – 146, Jun 2004

[89] J. Li, C. Blake, D.S.J.De Couco, H. Lee, R. Morris, "Capacity of Ad Hoc Wireless Networks," in *Proceedings of the ACM SIGMOBILE'2001*, Rome, Italy, July 2001

[90] P. Jacquet, A. Laouiti, P. Minet, and L. Viennot, "Performance of multipoint relaying in ad hoc mobile routing protocols," in *Proceedings of the Networking 2002*, Pise, Italy, 2002

[91] Richard Ogier, Fred Templin, Bhargav Bellur and Mark Lewis, "Topology Broadcast Based on Reverse-Path Forwarding (TBRPF)," presentation at IETF MANET WG meeting, March 2002, available at http://tbrpf.erg.sri.com/docs/tbrpf-manet-mar02.ppt

[92] Johnson, D. and Maltz, D. “Dynamic source routing in ad hoc wireless networks,” Mobile Computing (ed. T. Imielinski and H. Korth), Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996

[93] Z. Haas and M. Pearlman. “The Performance of Query Control Schemes for the Zone Routing Protocol,” in *Proceedings of the ACM SIGCOMM’1998*, Vancouver, Canada, September 1998

[94] C. Zhu, Y. Liu, M. Lee, *et al*, “Meshing the Wireless Personal Area Networks”, submitted to Communications Magazine, Jan, 2008