On improving the performance of multicasting in mobile ad hoc networks

Xie, Jiawei

2006

Xie, J. (2006). On improving the performance of multicasting in mobile ad hoc networks.
Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/47491>

<https://doi.org/10.32657/10356/47491>

Nanyang Technological University

Downloaded on 26 Apr 2023 14:55:58 SGT

On Improving the Performance of Multicasting in Mobile Ad Hoc Networks

Xie Jiawei

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Doctor of Philosophy

2006

TK
5105.77
X6
2006

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor, Prof. Anil Kumar Gupta, for his professional guidance, invaluable assistance, constant encouragement and patience throughout my graduate study at Nanyang Technological University. He always found time whenever I needed an intelligent discussion. I am grateful to Prof. Sukumar Nandi and Prof. Amitabha Das for their extensive and insightful discussions, comments and advice on various aspects throughout my research work.

I am eternally indebted to my parents, Mr. Xie Tingmen and Ms. Tan Anna, and my wife Ms. Huang Beilei for their love, encouragement and support, which the words cannot describe.

TABLE OF CONTENTS

ACKNOWLEDGMENTS.....	i
TABLE OF CONTENTS	ii
TABLE OF FIGURES.....	v
ABSTRACT	vii
LIST OF ABBREVIATIONS.....	ix
PUBLICATIONS FROM RESEARCH WORK	xii
 Chapter 1. Introduction.....	 1
1.1 MANETs and Multicasting	1
1.2 Medium Access Control (MAC) Protocol	3
1.3 Assumptions for Thesis.....	7
1.4 Research Objectives and Contributions.....	7
1.4.1 Research Objectives	7
1.4.2 Main Contributions	9
1.5 Organization of the Thesis.....	11
 Chapter 2. Literature Review of Multicast Routing Protocols for MANETs.....	 13
2.1 Multicast Ad Hoc On-Demand Distance Vector Protocol (MAODV).....	14
2.2 On-Demand Multicast Routing Protocol (ODMRP).....	16
2.3 Patch-ODMRP.....	18
2.4 Dynamic Core based Multicast routing Protocol (DCMP).....	21
2.5 Neighbor-Supporting Multicast Protocol (NSMP).....	23
2.6 Multicast Core-Extraction Distributed Ad hoc Routing (MCEDAR).....	25
2.7 Multicast routing protocol based on Zone Routing (MZR).....	28
2.8 Adaptive Dynamic Backbone Protocol (ADB).....	31
2.9 Adaptive Demand-Driven Multicast Routing (ADMR).....	33
2.10 Mobility-Based Hybrid Multicast Routing (MHMR) Protocol.....	37
2.11 Ad Hoc Multicast Routing (AMRoute) Protocol	39
2.12 Applying mobility prediction in ODMRP using GPS	41
2.13 Summary	43
 Chapter 3. The Gateway-Based Multicast Protocol (GBMP).....	 47
3.1 Data Structures.....	48
3.2 Protocol Description.....	49

3.2.1	Multicast Topology	49
3.2.2	Selection of Multicast Group Leader	52
3.2.3	Data Delivery	53
3.2.4	Passive Receive Mode (PRM).....	54
3.2.5	Bi-directional Local Link Repair (BLLR).....	56
3.2.6	Membership Management.....	63
3.3	Performance Evaluation	63
3.3.1	Performance Metrics	65
3.3.2	Simulation Results.....	68
3.4	Summary	79

Chapter 4. Literature Review of Reliable MAC Protocols for MANETs..... 81

4.1	IEEE 802.11 MAC DCF.....	81
4.2	Robust Broadcast	85
4.3	Broadcast Support Multiple Access (BSMA)	86
4.4	Broadcast Medium Window (BMW).....	88
4.5	Broadcast Scheme with Broadcast Acknowledgement (BACK).....	90
4.6	Batch Mode Multicast MAC (BMMM) Protocol	91
4.7	Multiple Access Collision Avoidance Protocol for Multicast Service (MACAM).....	93
4.8	Reliable Point-to-Multi-Point (RPMP).....	94
4.9	Summary	95

Chapter 5. Providing Reliable Broadcast Service at the MAC Layer..... 97

5.1	Data Structures.....	98
5.2	Round-Robin Acknowledge and Retransmit (RRAR).....	99
5.2.1	Overview of RRAR	99
5.2.2	Protocol Description.....	100
5.2.3	Performance Evaluation	102
5.3	Adaptive Round-robin Acknowledge and Retransmit (ARAR)	110
5.3.1	Overview of ARAR.....	110
5.3.2	Protocol Description.....	110
5.3.3	Performance Evaluation	114
5.4	Summary	123

Chapter 6. Literature Review of Congestion Control Schemes in Wireless Networks..... 125

6.1	Alleviating broadcast storm	125
6.2	Reliable ODMRP with Congestion Control.....	128
6.3	Congestion Bandwidth Algorithm (CBW).....	129
6.4	COngestion Detection and Avoidance (CODA).....	130
6.5	Weighted Load Metric	132
6.6	Congestion-controlled Adaptive Lightweight Multicast (CALM).....	133
6.7	Media Access Delay Regulator (MADR) Algorithm	134
6.8	Summary	135

Chapter 7. Coping With Congestion 137

7.1	Motivation	138
-----	------------------	-----

- 7.2 Queue Splitting (QS) 139**
- 7.3 Adaptive Retransmit (ADRT) 142**
- 7.4 Performance Evaluation 144**
 - 7.4.1 Value of MAX_DFO_PKT_ITVL 145
 - 7.4.2 Improvement by QS 147
 - 7.4.3 Improvement by ADRT 149
 - 7.4.4 Improvement by combination of QS and ADRT 150
- 7.5 Summary 152**

- Chapter 8. Conclusions and Future Work..... 155**

- 8.1 Main Research Contributions 155**
- 8.2 Future Work 158**
 - 8.2.1 Energy-efficiency and Power-awareness in Multicasting 158
 - 8.2.2 Quality of Service (QoS) Support for Multicast Routing Protocol 158
 - 8.2.3 Impact of High Bit Error Rate..... 159

- References 161**

- Annex: Location of the Source Code of the Simulation Models 169**

TABLE OF FIGURES

Figure 1.1. Multicast Operation.....	2
Figure 1.2. Hidden Terminal and Exposed Terminal Problems	6
Figure 2.1. MAODV Join Procedures	15
Figure 2.2. Construction of the mesh in ODMRP	18
Figure 2.3. An example of the patching procedure.....	20
Figure 2.4. Packet Header of NSMP	23
Figure 2.5. Data delivery in MCEDAR	27
Figure 2.6. Multicast tree creation inside a zone	29
Figure 2.7. Multicast tree propagation to the entire network.....	30
Figure 2.8. Tree flood and Network flood.....	34
Figure 2.9. Loosely structured multicast tree in ADMR for data delivery	35
Figure 2.10. Local sub-tree repair.....	36
Figure 2.11. User-Multicast Tree.....	40
Figure 3.1. Local Group Construction.....	51
Figure 3.2. A Possible GBMP Multicast Topology.....	52
Figure 3.3. Multicast Data Delivery	54
Figure 3.4. Passive Acknowledgment	55
Figure 3.5. Upstream-Node-Initiated Local Repair (UNILR)	59
Figure 3.6. Simulation Results of 1 Source, TP-3.1 Scenario	69
Figure 3.7. Simulation Results of 1 Source, TP-3.2 Scenario	70
Figure 3.8. Simulation Results of 3 Sources, TP-3.1 Scenario	74
Figure 3.9. Simulation Results of 3 Sources, TP-3.2 Scenario	75
Figure 4.1. Four-way handshake	83
Figure 4.2. Reliable Broadcast.....	85
Figure 4.3. Different Scenario of Collision Detection.....	86
Figure 4.4. Imprecise NAV Problem.....	89
Figure 4.5. Back Window.....	90
Figure 4.6. BMMM Protocol.....	92
Figure 4.7. MACAM RTS Frame Format	93
Figure 4.8. MACAM Operation	93
Figure 4.9. RPMP	95
Figure 5.1. Received Frame Bitmap (RFB).....	98
Figure 5.2 Modified Data Frame Header.....	100
Figure 5.3 Round-Robin Acknowledgement.....	101
Figure 5.4. BrAck Frame Format	101
Figure 5.5. Reliability Improvement of RRAR over IEEE 802.11 MAC and MACAM.....	106
Figure 5.6. Throughput Improvement of RRAR over IEEE 802.11 MAC.....	107
Figure 5.7. Normalized Byte Overhead	109
Figure 5.8. Multiple Packets Packed in One Frame.....	111
Figure 5.9. Reliability Improvement of ARAR over IEEE 802.11 MAC.....	117
Figure 5.10. Throughput Improvement of ARAR over IEEE 802.11 MAC.....	118
Figure 5.11. Performance Comparison of ARAR and RRAR	121
Figure 7.1. High delay introduced by ARAR under heavy traffic	139
Figure 7.2. The Queue-Splitting (QS) mechanism	141
Figure 7.3. Dynamic adjustment of the retransmission limit.....	143
Figure 7.4. Performance under Different MAX_DFQ_PKT_ITVL for ODMRP	146
Figure 7.5. Performance under Different MAX_DFQ_PKT_ITVL for GBMP	147

Figure 7.6. Performance comparison of different schemes 151

ABSTRACT

In recent years, Mobile Ad hoc Networks (MANETs) have attracted considerable research interests as they have numerous advantages, such as ease of deployment, robustness, self-organization and self-healing. However, due to the unique characteristics such as frequently varying topology, narrow bandwidth, moderate quantity of memory storage, finite battery power, and relatively lower processing ability etc., it is a very challenging task to perform multicasting in MANETs.

In this thesis, our main objective is to improve the performance of multicasting in MANETs at the Medium Access Control (MAC) and network layers. We achieve this objective in three main steps.

As our first step, we propose a new multicast routing protocol, called Gateway-Based Multicast Protocol (GBMP). GBMP improves upon the existing protocols in terms of the speed and the cost of the multicast tree repair, the transmission efficiency, and the control overhead. A number of novel features such as, global as well as local maintenance of group-shared trees, a bi-directional multicast tree repair mechanism, and the suppression of unnecessary acknowledgments are introduced in GBMP.

In the second step, we observe that many routing protocols rely heavily upon the MAC layer's broadcast service. Nonetheless, unreliable broadcast by the IEEE 802.11 MAC is detrimental to the upper layers' performance. We set forth to improve the IEEE 802.11 MAC's broadcast by proposing the Round-Robin Acknowledge and Retransmit (RRAR) mechanism. RRAR makes every effort to avoid the notorious "Ack storm" problem. An enhanced version of RRAR, called the Adaptive Round-robin Acknowledge and Retransmit (ARAR), is also proposed.

Contrary to the common belief that four-way handshake cannot be applied for broadcast and multicast, ARAR utilizes RTS/CTS/Data/BrAck frame exchange for reliable broadcasting. In addition, ARAR adjusts the Data frame length in response to different traffic loads such that the overhead for transmitting every Data frame can be reduced.

Traditionally, protocols of different layers are designed separately. However, the interaction between protocols in different layers is an important issue, especially for MANETs because of the physical constraints. One instance is that under heavy traffic, the retransmissions brought about by ARAR can exacerbate the state of congestion in the network. Therefore, as our third step, we address this problem by using an integrated approach where the routing layer and MAC layer issues are considered together. Specifically, we propose 1) a Queue Splitting (QS) mechanism, which splits the queue in-between the MAC and routing layers into two queues, the Control Queue and the Default Queue, and a priority scheduling is used at the MAC layer to alleviate congestion; 2) ADaptive ReTransmission (ADRT) that avoids congestion by dynamically adjusting the retransmission limit of ARAR based on the medium's busyness. The effect of these modifications is evaluated using the Gateway-Based Multicast Protocol (GBMP) and On-Demand Multicast Routing Protocol (ODMRP) for the network layer.

Hence, our work addresses the three key challenges involved in multicasting, the routing protocol, the MAC protocol and the congestion control, by proposing GBMP, ARAR (as an advanced version of RRAR), as well as QS and ADRT mechanisms. Through extensive simulations, it is demonstrated that, with the introduction of our proposed schemes, the performance of multicasting in MANETs can be improved substantially over the existing solutions.

LIST OF ABBREVIATIONS

AAck	Attach Acknowledgment
Ack	Acknowledgment
ADMR	Adaptive Demand-Driven Multicast Routing
ADRT	ADaptive ReTransmit
ANtf	Attach Notification
ARAR	Adaptive Round-robin Acknowledge and Retransmit
AReq	Attach Request
BcDQ	Broadcast Data Queue
BE	Bandwidth Efficiency
BLLR	Bi-directional Local Link Repair
BrAck	Broadcast Acknowledgment
CBO	Control Byte Overhead
CBR	Constant Bit Rate
CCA	Clear Channel Assessment
CDMA	Code Division Multiple Access
CO	Control Overhead
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CtrlQ	Control Queue
CTS	Clear To Send
CW	Contention Window
DCF	Distributed Coordination Function
DIFS	DCF Interframe Space
DftQ	Default Queue
DRA	Data Receive Acknowledgment
ECS	Enhanced Carrier Sensing
EIFS	Extended Interframe Space
FCS	Frame Check Sequence
FDMA	Frequency Division Multiple Access
FG	Forwarding Group

GBMP	Gateway-Based Multicast Protocol
GJQ	Global Join Query
GJR	Global Join Reply
GloMoSim	Global Mobile Information Systems Simulation Library
GN	Gateway Node
JQ	Join Query
JR	Join Reply
LJQ	Local Join Query
LJR	Local Join Reply
MANET	Mobile Ad Hoc Network
MAC	Medium Access Control
MACAM	Multiple Access Collision Avoidance Protocol for Multicast Service
MSDU	MAC service data unit
NAV	Network Allocation Vector
NL	Neighbor List
NPO	Normalized Packet Overhead
OCPL	Occurrence of Consecutive Packet Loss
ODMRP	On-Demand Multicast Routing Protocol
PCF	Point Coordination Function
PDU	Protocol Data Unit
PDR	Packet Delivery Ratio
PIFS	PCF Interframe Space
PRM	Passive Receive Mode
PsvAck	Passive Acknowledgment
QoS	Quality of Service
QS	Queue Splitting
RIAP	Receiver-Initiated Attach Procedure
Rack	Repair Acknowledgment
RFB	Received Frame Bitmap
RN	Repair Notification
RQ	Repair Query
RRAR	Round-Robin Acknowledge and Retransmit
RTS	Request To Send
SIFS	Short Interframe Space
TC	Traffic Category
TDMA	Time Division Multiple Access

TE	Transmission Efficiency
UNILR	Upstream-Node-Initiated Local Repair
WOCPL	Weighted Occurrence of Consecutive Packet Loss

PUBLICATIONS FROM RESEARCH WORK

Journal publications:

- “Gateway-Based Multicast Protocol – a Novel Multicast Protocol for Mobile Ad hoc Networks”, IEE Proceedings – Communications Volume 152, No. 6, December 2005.
- “Improving the Reliability of IEEE 802.11 Broadcast Scheme for Multicasting in Mobile Ad hoc Networks”, IEE Proceedings – Communications (accepted in May, 2005).

Conference publications:

- “Gateway-Based Multicast Protocol for Mobile Ad Hoc Networks”, The Fifth IFIP TC6 International Conference on Mobile and Wireless Communications Networks (MWCN 2003), Singapore, October 27th – 29th, 2003 (acceptance ratio: 40%).
 - “An Improvement to the Reliability of IEEE 802.11 Broadcast Scheme for Multicasting in Mobile Ad hoc Networks”, The First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2004), Santa Clara, USA, October 4th – 7th, 2004 (acceptance ratio: 19%).
 - “Improving the Reliability of IEEE 802.11 Broadcast Scheme for Multicasting in Mobile Ad hoc Networks”, IEEE Wireless Communications and Networking Conference (WCNC 2005), New
-

Orleans, USA, March 13th – 17th, 2005 (acceptance ratio: 30%);

- “Efficient Integration of the Multicast Routing Protocol and the Reliable MAC Protocol for MANETs”, 19th International Teletraffic Congress (ITC 19), Beijing, China, August 29th – September 2nd, 2005 (acceptance ratio: 40%);

Papers submitted for publication in journals:

- “On Providing a Reliable MAC Layer Broadcast Scheme for Multicasting in Mobile Ad hoc Networks”, submitted to Wiley Wireless Communications and Mobile Computing in November 2004;
 - “Alleviating Congestion caused by MAC layer Retransmission in Mobile Ad hoc Networks”, submitted to Elsevier Journal of Ad Hoc Networks in May 2005;
-

Chapter 1. INTRODUCTION

1.1 MANETs and Multicasting

Recently, Mobile Ad hoc NETWORKs (MANETs) [1] have attracted much research interest in the field of mobile computing. A MANET is comprised of wireless hosts or nodes, which may either move around freely and randomly or remain static. There is no pre-existing infrastructure through which these hosts can communicate with each other. These features brings along the advantages such as ease of deployment, robustness, self-organization and self-healing, which are especially useful in numerous scenarios that require rapid infrastructure-less local network deployment. Furthermore, when all the nodes in the network move, the network topology varies frequently and may consist of unidirectional or bidirectional links. In addition, similar to the other wireless networks, a MANET has very limited resources: narrow bandwidth, moderate quantity of memory storage, finite battery power, and relatively lower processing ability, etc. In particular, due to the effects of collision, interference and fading, the narrow bandwidth is further decreased as packet retransmissions are invoked by packet loss. Moreover, the transmission range is normally limited, which makes the communication between two nodes mostly through multiple hops. As a result of these challenges, the traditional routing protocols employed in the wire-line

networks are not suited for the MANET.

The above remarks are also true for the multicasting in a MANET. As defined in [2], multicasting in a network is the transmission of a packet to a subset of the hosts in the network. The subset makes up the multicast group specified by a multicast address. An efficient multicast facility provides packet delivery to the group of hosts (Figure 1.1) with lower overheads in comparison with broadcast to all hosts or unicast to each host within the group. Multicasting is able to improve the efficiency and robustness of distributed systems and applications such as those in the MANET, and efficiently utilizes the limited network resources while performing group communications. In order to receive data packets from a source, a node must first join the multicast group, i.e. become a member of the group. Of course, the node should be free to join or leave multicast groups at any time. One should note that a node that intends to send data packets to a multicast group may not necessarily be a member of the group. A node may also be a member of more than one multicast groups at any time.

Figure 1.1. Multicast Operation

Compared to unicasting, which is defined as a point-to-point communication in the network, multicasting is bandwidth efficient in that the source simply sends each data packet to the multicast group once (one session) and the data packets are

delivered to the group members; while in unicasting, the source needs to send the same data packet to different destination nodes individually (in multiple sessions). Considering the features of the MANET, the mobility may cause a path from a source to a destination to break frequently, making it more difficult to maintain multiple sessions if the unicasting is used to deliver data packets. Moreover, if the size of the multicast group is large, memory storage requirement for so many sessions is also a critical issue.

Compared to broadcasting, which is defined as a one-to-all communication in the network, multicasting allows multicast data packets to be delivered to a selected group of the nodes in the network; while in the broadcasting, all nodes are involved in data packet transmission. Also, frequent broadcasting of data packets produces congestion and collisions, which are not desirable, especially in the wireless networks. Multicasting solves these problems by alleviating the traffic redundancy and therefore reducing the probability of congestion and collisions. Consequently, in a MANET, multicasting is more attractive for a group communication.

There are various practical applications that can make use of the multicasting in the MANETs, which have been proposed in [3]: group-oriented mobile commerce, military communications and cooperation, distance education and entertainment services, indoor group audio/video conferences and intelligent transportation system, etc. Therefore, to perform research in the field of multicasting in MANETs is important and of realistic value.

1.2 Medium Access Control (MAC) Protocol

In wireless networks, a MAC protocol moderates access to the shared medium by defining rules that allow the nodes to communicate with each other in an orderly and efficient manner. The design of the MAC protocols is affected significantly by the wireless medium's unique properties [4], such as *half-duplex operation*, *time varying*

channel, burst channel errors and location dependent carrier sensing.

The reason for applying the half-duplex operation is that it is very difficult for a wireless system to receive data while its transmitter is sending data. When a node is transmitting data, the signal energy leakage from the transmitter to the receive path¹ generally has much higher power than the signal received from other nodes. Therefore, it is very difficult to detect and decode other nodes' signal while transmitting. This also implies that detecting collision while transmitting data, which is used in wire-line protocols, has not been possible so far.

The time varying radio channel is the result of superimposed time-shifted and attenuated versions of the transmitted signals. In order to mitigate the adverse effect brought about by the time-varying link quality, handshaking is widely utilized prior to data transmission: two nodes exchange small frames to verify the link status; if the handshake is successful, which indicates a good communication link, the data transmission follows.

As a result of time varying channel characteristics, channel errors are more likely to happen in wireless transmission than in wire-line. Consequently, the strategy of link-layer retransmission is widely used to deal with channel errors. This strategy uses acknowledgements (Acks) to detect frame errors: if a required Ack is not received following the previous Data frame, the retransmission is called upon.

Carrier sensing also plays an important role in the wireless communication in comparison to the wired communication. Carrier sensing is a function of the receiver's position relative to the frame's transmitter, which is due to signal strength's attenuation over the traversed distance. This yields two major problems: the *hidden terminal problem* and the *exposed terminal problem*. A hidden terminal

¹ This signal energy leakage is also referred to as self-interference.

(node) is one that is within the transmission range² of the intended destination but out of the transmission range of the sender. The simultaneous transmission from a hidden node and a sender can cause collisions at the destination. An exposed node is one that is within the transmission range of the sender but out of the transmission range of the destination. The exposed nodes are restrained by the sender from transmitting even though there will be no collision occurring at the destination. The exposed terminal problem results in underutilized bandwidth. As depicted in Figure 1.2 (a), while node A is transmitting to node B, node C cannot hear the transmission from A because it lies outside A's transmission range (it is also possible that there is a physical obstacle in between A and C). Therefore, during this transmission when node C senses the channel, it may mistakenly consider that the channel is idle. If C starts transmission, it interferes with the data reception at B. In such a situation, nodes A and C are referred as hidden terminals to each other. The exposed terminal problem is illustrated in Figure 1.2 (b), where B is transmitting data to A. Being within B's transmission range, C is able to sense B's transmission and it assumes that the media is busy. C is then restrained from attempting to transmit data to D. As a matter of fact, the transmission from C does not reach A (as A and C are out of each other's transmission range), and therefore will not interfere with the reception at node A; indeed, C can communicate with D in parallel with B's transmission to A. In this case, node C is called the exposed terminal to node B.

Most existing wireless MAC protocols can be classified into two categories, namely centralized and distributed. Centralized protocols such as Time Division Multiple Access (TDMA) [5], Frequency Division Multiple Access (FDMA) [6] and Code Division Multiple Access (CDMA) [7] require base stations, which are unavailable in ad hoc networks, to facilitate the inter-node communications.

² Transmission range is the range within which a frame can be successfully received if there is no transmission error.

Therefore, distributed MAC protocols such as Carrier Sense Multiple Access (CSMA) [8], Multiple Access Collision Avoidance (MACA) [9], MACA Wireless (MACAW) [10], Floor Acquisition Multiple Access (FAMA) [11] and the standardized IEEE 802.11 MAC Distributed Coordination Function (DCF) [12] are the candidates for ad hoc network's MAC layer.

Figure 1.2. Hidden Terminal and Exposed Terminal Problems

Most research works on wireless ad-hoc networks have adopted IEEE 802.11 MAC protocol, which is the de-facto industry standard for Wireless LANs, although it is not designed for multiple-hop networks. Our research work is also based upon IEEE 802.11, and we propose schemes to enhance the standardized protocol to make

it suitable for the multicasting in MANETs.

1.3 Assumptions for Thesis

The following assumptions are used throughout this thesis:

- All nodes have identical capabilities;
- All nodes are equipped with an omni-directional antenna, and the antenna cannot transmit and receive concurrently;
- The transmission power remains fixed; therefore, the transmission range is fixed. The covered area by a transmission is assumed to be the circle centered at the transmitter with the radius equal to the transmission range;
- The capture effect of the wireless system is not supported³;
- The modulation rate remains fixed;
- The battery power is infinite.
- Bit error rate is assumed to be very low and thus neglected; frames are corrupted only due to collision at the receiving nodes.

A high bit error rate will primarily reduce the effective bandwidth and is unlikely to have any other major impact on the performance of the protocols studied in the thesis. Verification of this assumption is left as a topic of further research.

1.4 Research Objectives and Contributions

1.4.1 Research Objectives

As mentioned, compared to unicast and broadcast, multicast is the most efficient to support the transmission of packets to a subset of the hosts in the network. However, multicasting in MANETs is a challenging task for several reasons. First, the limited

³ We will discuss in Chapter 4 that it is very difficult to make use of capture in multicasting scenarios.

and possibly asymmetrical radio bandwidth limits the number of control and data packets to travel through mobile nodes, which may cause inaccurate operations that rely heavily upon the timely delivering of the packets. Second, the nodes' mobility leads to an ever-changing topology, making the construction and maintenance of the multicast topology more difficult. Third, the unreliable radio links and arbitrarily changing topology also affects group membership management. Nodes may fail to send or receive membership management control packets. This may result in a node's wrong knowledge of others' membership (i.e. whether other nodes are members or not). Consequently, multicast topology may evolve incorrectly and may further prevent the multicast operations from being executed appropriately and efficiently.

In the past years, although a plethora of multicast protocols for MANETs exist in the literature to solve these problems, there are many areas where improvements are desirable and possible, e.g., the speed and the cost of the multicast topology repair mechanism, the transmission efficiency, the amount of control overhead, end-to-end delay, etc. In this thesis, one of our objectives has been to design a multicast routing protocol for MANETs aiming to improve in a number of facets. Note that the designing of the multicast routing protocol is based on the assumption that the underlying MAC layer is able to provide reliable transmission service.

However, simply designing a new multicast routing protocol may not be adequate. As the network layer must utilize the transmission service from the MAC layer, the performance of the MAC protocol affects the performance of the routing protocol significantly. The characteristics of the MAC layer protocol must be studied carefully and, if necessary, modified to fit the requirements of the routing protocols. Our research will then focus on improving the reliability of the IEEE 802.11 MAC DCF broadcast service, and consequently improve the performance of the network layer. This research objective is accomplished in two steps. In the first step, our work solely concentrates in providing reliable broadcast service at the MAC layer. Then,

in the second step, we examine the interaction between the MAC layer and the network (routing) layer. Based upon this, schemes will be designed to smoothly and efficiently integrate the MAC reliable broadcast with the multicast routing protocol.

1.4.2 Main Contributions

Our main contributions in this thesis are as follows:

Gateway-Based Multicast Protocol (GBMP): We propose a new multicast protocol for MANET, called the Gateway-Based Multicast Protocol (GBMP) that seeks to improve upon the existing protocols in terms of: (a) the speed and the cost of the multicast tree repair mechanism; (b) the transmission efficiency; and (c) the amount of control overhead. GBMP achieves these improvements by using a number of novel features such as: (a) both global and local maintenance of group-shared loosely structured trees; (b) a bi-directional multicast tree repair mechanism; and (c) the suppression of unnecessary acknowledgments. In addition, we have introduced a new metric called *Weighted Occurrence of Consecutive Packet Loss (WOCPL)* to measure the discontinuity in data packet delivery. Extensive simulation study shows that GBMP outperforms the more established On-Demand Multicast Routing Protocol (ODMRP) [13] and Adaptive Demand-Driven Multicast Routing (ADMR) [14] protocols in a number of important performance metrics under different traffic patterns and source node counts.

Round-Robin Acknowledge and Retransmit (RRAR): Broadcasting is one of the essential communication models of MANETs. Many MANET multicast routing protocols rely heavily upon the MAC layer's broadcast support. However, the broadcast mechanism of the standard IEEE 802.11 cannot provide reliable broadcasting service, and thus degrades the multicasting performance. Therefore, we propose an extension to the IEEE 802.11 broadcast mechanism, called Round-Robin Acknowledge and Retransmit (RRAR), to improve the broadcasting reliability. Different from other mechanisms, RRAR has a novel round robin acknowledge

scheme, in which the lost frames are reported by the neighboring nodes in a round-robin style to avoid the notorious *Ack explosion* problem. We employ a simple and effective Data/BrAck (Broadcast Acknowledgment) frame exchange sequence. Receiving the BrAck, the Data sender calculates and records the indices of the frames that are lost and retransmits them. The proposed scheme provides a reliable broadcast service to the routing layer, and the performance of multicast routing protocols can be improved significantly.

Adaptive Round-robin Acknowledge and Retransmit (ARAR): We further enhance the MAC layer's reliable broadcast mechanism by proposing the Adaptive Round-robin Acknowledge and Retransmit (ARAR) scheme. Based upon the observation that there is tremendous overhead for every Data frame (packet) transmitted in RRAR, ARAR packs multiple packets into one frame and thus reduces the overhead. For short frames, a simple and effective Data/BrAck (Broadcast Acknowledgment) exchange, as used in RRAR, is applied; for long frames, the *RTS/CTS/Data/BrAck four-way handshake* is utilized. Therefore, ARAR provides an even more reliable broadcast service than RRAR to the routing layer, and the performance of multicast routing protocols is further improved significantly.

Queue-Splitting (QS) for ARAR: Protocols of different layers in the protocol stack are traditionally designed separately. However, we observe that simple integration of MAC and network layers in the MANETs may result in poor performance especially under heavy traffic. We exemplify this by using the Adaptive Round-robin Acknowledge and Retransmit (ARAR) for the MAC layer, and the Gateway-Based Multicast Protocol (GBMP) and On-Demand Multicast Routing Protocol (ODMRP) for the network layer. In order to have a smooth and efficient integration, we propose a Queue Splitting (QS) mechanism, which splits the queue in-between the MAC and routing layer, into two queues, the *Control Queue* and the *Default Queue*. A priority scheduling is used at the MAC layer to serve these two queues, which alleviates the congestion and improves the performance substantially.

ADaptive ReTransmit (ADRT): The limited bandwidth is a critical limitation in MANETs. The Adaptive Round-robin Acknowledge and Retransmit (ARAR) scheme improves the reliability of the MAC layer's broadcast service using retransmissions. However, excessive retransmissions may cause congestion in the network. Bearing in mind the objective of efficiently integrating reliable MAC protocol within the routing protocol, we further modify the ARAR scheme by introducing ADaptive ReTransmission (ADRT) that dynamically adjusts the *retransmission limit* based on the medium's busyness so as to avoid congestion. The purpose of ADRT is to achieve a good trade-off between reliability and other performance metrics.

Therefore, with the GBMP as the multicast routing protocol, the ARAR at the MAC layer for providing reliable broadcast service to the network layer, and QS and ADRT utilized for the smooth integration between the MAC and network layer, we are able to provide high performance multicasting in MANETs.

1.5 Organization of the Thesis

The remainder of this thesis is organized as follows. In Chapter 2, a critical review on the existing multicast routing protocols for MANETs is given. In Chapter 3, our proposed Gateway-Based Multicast Protocol (GBMP) is presented. As the broadcast reliability at the MAC layer is essential to the performance of multicasting, in Chapter 4, we review protocols that aim to improve IEEE 802.11 MAC DCF's broadcast. Then, Chapter 5 presents the Round-Robin Acknowledge and Retransmit (RRAR) and Adaptive Round-robin Acknowledge and Retransmit (ARAR), which aim at providing reliable broadcast at the MAC layer. Chapter 6 reviews the works on congestion control schemes in wireless networks, which is followed by Chapter 7, where we present the Queue Splitting (QS) and ADaptive ReTransmission (ADRT) mechanisms. This thesis is concluded in Chapter 8. Some future research work is

also described in this last chapter.

Chapter 2. LITERATURE REVIEW OF MULTICAST ROUTING PROTOCOLS FOR MANETs

In this chapter, multicast protocols that have appeared in the literature and are related to our work are critically examined, and their advantages and disadvantages are discussed. However, first we give an overview of the issues that are of concern while designing multicast protocols.

In the broadcast-oriented mobile wireless network, due to interference, noise, signal attenuation and mobility, a wireless user may not receive the queries from the multicast sources and vice versa. This leads to failure of the user to enter a multicast group. Therefore, while designing the membership management schemes, the network's mobility and broadcast nature must be taken into account.

Furthermore, the multicast routing should also take care of the link breakages due to the network's mobility. There are two ways to handle this problem. One is to construct an appropriate multicast topology for data packet delivery. A topology that is robust to the dynamic environment will reduce the occurrence of link failures. In addition, there are times when delivery failures occur frequently in a highly mobile network, and even redundant routes are unable to alleviate the problem. In these cases, a repair mechanism, which is the other method, must be provided enabling the

receivers to continue receiving their data. Furthermore, a robust multicast topology may also help in repairing the broken links. However, considering the scarce radio resources, control overheads incurred should be limited to a minimum.

2.1 Multicast Ad Hoc On-Demand Distance Vector Protocol (MAODV)

MAODV [15] is a *tree-based* protocol. MAODV establishes a group shared multicast tree originated from the Group Leader connecting multicast group members and non-group nodes in-between.

When a node, say node L, wants to join a multicast group, in order to discover the route to the group, L starts by broadcasting a *route request (RREQ)* with the group's network address as the destination and the "J" flag set indicating that it wishes to join. If node L is the first one to join the group, it is certain that there will be no reply. After waiting for a given duration, the node creates the new group and declares itself as the Group Leader by periodically broadcasting a *hello message (Group Hello)* with a group sequence number. This group sequence number is initialized and incremented periodically by the group leader. Member nodes store this sequence number and use it when they need to respond to an RREQ. In the Group Hello message, hop count information is also included. The nodes on the tree are able to know the distance from the Group Leader when they receive a hello message.

If a node, say node R (see Figure 2.1), decides to join a group, which already exists, the node may receive multiple *Route Reply (RREP)* from the existing tree nodes in response to its RREQ. Other nodes rebroadcast the RREQ if they do not know the route to the group leader or if they are not on the multicast tree until it reaches a multicast tree node, and this tree member node will send back the RREP. Each RREP implies a route from the multicast group to the node R. Because of the

tree topology used in MAODV, the node R must explicitly choose one of the routes and activate it by unicasting a *multicast activation (MACT)* message. This message causes every node along the path to add itself to the multicast tree and form a new multicast branch that terminates at node R. Though the nodes along the path may not necessarily be multicast members, they must reside on the tree to act as routers for the data delivery.

Figure 2.1. MAODV Join Procedures

The nodes in the multicast group are free to leave the group at any time. However, a node that is not a leaf on the multicast tree must remain on the tree to continue acting as a router. For example, if Node A in Figure 2.1 decides to leave the group, it will continue to act as a router. On the other hand, if a multicast member, say node R, which is already on the multicast tree, decides to leave the group, because it is a tree leaf, it explicitly prunes itself from the tree by sending a special MACT message to the group. After receiving the MACT, nodes which are on the same tree branch as the node R but not multicast members (e.g., node T), remove themselves from the tree, and forward the MACT to its parent until this message reaches another multicast member, say node A.

Due to the mobility, tree links are liable to break. The node that is the first one

downstream from the link break point is in charge of initiating the link repair by sending an RREQ message with the 'J' flag set. Only a multicast tree node whose hop count towards the group leader is less than or equal to that of the initiating node and that has a fresh enough routing information to the group leader is allowed to respond.

The tree-based topology used in the MAODV is not suitable for MANETs because high mobility produces frequent link breakages that trigger route repairs very often. Excessive control overheads will be incurred in maintaining the multicast tree, which will reduce the available bandwidth for data delivery. Moreover, as hard state is used in the routing table, every tree member keeps track of the link leading to its parent and acts upon the link breakage. The repair mechanism of MAODV produces sub-optimal routes between the group leader and the multicast members. As a result, more nodes will be included in the multicast tree, lengthening the branches, and therefore, the branches are more vulnerable to link breakages in the case of high mobility. Consequently, the MAODV suffers from poor packet delivery ratio.

2.2 On-Demand Multicast Routing Protocol (ODMRP)

ODMRP [13] is an effective *mesh-based* multicast protocol. A mesh is established among the sources and the destination nodes. The mesh is constructed using the concept of *Forwarding Group (FG)* [16]. Any node in an FG is in charge of forwarding (broadcast) multicast data packets of a multicast group, i.e., a node in an FG forwards a multicast packet if it is not a duplicate one. A soft state approach is applied in the FG maintenance, i.e., once a node becomes an FG node, a timer is associated with it, and before the timer expires, the node should forward every multicast packet it has received; after that, it quits from the FG until the next time it becomes an FG node again.

Nodes lying in-between the sources and the destination nodes may act as FG

nodes. In ODMRP, group membership and multicast routes (multiple paths provided by the mesh) are established and updated by the sources on demand. If a multicast source has data to send to the group, it broadcasts a *Join Query (JQ)*. When a node receives a non-duplicate Join Query, it stores the upstream node's ID (it may be used in setting up a reverse path back to the source), and rebroadcasts it. When this Join Query reaches a multicast receiver, the receiver creates a *Join Reply (JR)*, broadcasts it to the neighbors (of course, if the TTL of the Join Query is not decreased to zero, it is still forwarded). When a node receives a Join Reply, if the "next node ID" of one of the entries matches its own ID, the node realizes that it is on the path from a receiver to (one of) the sources and thus is part of the forwarding group. It then broadcasts its own Join Reply by checking its route table for the stored next hop addresses (the upstream node's ID) to the sources. The Join Reply is therefore propagated by each forwarding group member along the path until it reaches the multicast source. The query and the reply processes construct (or update) the multicast topology from the sources to the receivers and build the forwarding group—the mesh of nodes in between. Yet, considering the mobility, which may cause link breakages even though a mesh is there to provide moderate redundant paths, periodic broadcasting of the Join Query by the multicast senders is necessary to refresh the membership information and update the routes. This process is illustrated in Figure 2.2. Multicast senders S1 and S2 broadcast Join Query. Receiver nodes R1, R2 and R3 respond by sending Join Reply back to the sources. A mesh, including nodes B, C and R1, is then created. Notice that node R1 is now acting both as an FG node and a receiver.

The mesh provides topology redundancy that is able to properly cope with mobility. When a route from a multicast sender to one of the receivers breaks, other paths available will be used in forwarding multicast packets automatically. As depicted in Figure 2.2, when the link between node B and R1 breaks, node R1 is still able to receive data packets from node S1 via node C. However, a node may only be

included in the FG by periodic flooding of Join Query, which is relatively costly. Consequently, if a node moves to an area where less FG nodes exist, the ratio of the delivered multicast data packets will be reduced. Moreover, ODMRP suffers from high overheads. When either the number of the nodes in the network or the number of nodes in the multicast group increases, ODMRP scales poorly due to the broadcasting of the Join Query throughout the network and due to the delivery of data packets in the mesh topology [62]. When the number of multicast senders increases, the number of nodes in the FG also increases, leading to higher occurrences of congestion and collisions.

Figure 2.2. Construction of the mesh in ODMRP

2.3 Patch-ODMRP

Patch-ODMRP [17] extends ODMRP by deploying a local patching scheme instead of having very frequent mesh reconfigurations, i.e. frequent sending of the Join Query by the multicast sources.

In Patch-ODMRP, each FG node monitors if there is a possibility of being separated from the mesh. An FG node utilizes MAC layer *BEACON*, which is

defined in IEEE 802.11, to check who are in its neighborhood. By comparing this information with the upstream FG node information in the Forwarding Group table, the FG node is able to detect if an upstream FG node loses contact with it. If this is the case, the detecting FG node starts the patching procedure by flooding an *advertisement (ADVT)* packet, telling that the sending node has lost contact with the specified upstream FG node. The ADVT also contains fields specifying the multicast group ID, the multicast source ID, and the hop count the sending node is away from the multicast source. It is sent with limited hops (2 or 3 hops) in order to limit the scope of the flooded message.

When a node receives the ADVT packet, it updates the routing information to the sending node. Further, if it is an FG node serving the same multicast group and the same multicast source as the node that sent the ADVT, and if it is closer to the specified multicast source (by checking the hop count included in the ADVT), it generates a *PATCH* packet and sends (or forwards) it back to the node that sent the ADVT. The nodes along the path that the PATCH packet traversed are selected as temporary FG nodes for the multicast group. If the node receiving the ADVT is not qualified to connect the sending node to the FG, it forwards the ADVT to its neighbors provided the TTL of the ADVT is greater than 0.

The node that initiated the ADVT may receive several PATCH packets. It chooses the shortest path to the source that one of the PATCH packets specified.

Figure 2.3 illustrates the patch procedure. Suppose node F is an FG node having lost contact with its upstream FG node, node A. Nodes B and D are both FG nodes around it. Node F broadcasts the ADVT packet asking for patching to the FG. Both nodes B and D receive this packet. However, only node B replies by sending a PATCH packet, as node B is closer to the source node S than node F is. Both nodes D and G are farther away from the source node S than the node F is. Upon receiving the PATCH packet, node F chooses node B as its upstream node.

Figure 2.3. An example of the patching procedure

Patch-ODMRP outperforms ODMRP when operating in a network consisting of a small number of multicast sources, which results in a sparse mesh (a sparse mesh acts very similar to a tree, which is not able to cope well with mobility). Using the patching procedure, whenever a mesh is likely to separate, FG nodes are patched to the mesh for the subsequent data packet delivery, and therefore, frequent flooding of the Join Query to refresh the overall multicast topology can be avoided.

The flooding of Join Query by the multicast source is used as the last resort to refresh the entire topology. However, as a downstream node initiates the patching procedure, if the procedure fails, the multicast sources have no idea of the failure, and thus are unable to react to the patching failure appropriately. The patching procedure also introduces new nodes into the FG. With the increase in the number of FG nodes, more data packets for the multicast group will be forwarded. While this improves the packet delivery ratio, the transmission efficiency degrades. Longer interval in-between flooding of two consecutive Join Queries can reduce the control overhead; however, the lifetime for an FG node is lengthened, resulting in a higher number of FG nodes, which is a situation leading to inefficient topology and degradation of the transmission efficiency.

2.4 Dynamic Core based Multicast routing Protocol (DCMP)

DCMP [18] is another extension to ODMRP that tries to reduce control overhead by dynamically classifying the sources into *Active* and *Passive* categories.

Sources are ranked into three categories: *Active Sources*, *Core Active Sources* and *Passive Sources*. Active Sources are similar to the sources in ODMRP in the sense that they flood Join Query control packet regularly. Core Active Sources (Core Nodes) are those that act as a core for one or more Passive Sources. The core nodes are responsible for creating the shared mesh on behalf of the Passive Sources associated with them. A Passive source does not flood the Join Query for creating the multicast mesh; rather it depends on a nearby Core Active Source to forward its multicast packet to the receivers. That is, in DCMP, the shared mesh is constructed amongst the Core Nodes and the receivers. Whenever a passive source wants to send multicast packets, it sends them to its associated core node and the core node in turn will forward them to the mesh.

The maximum number of Passive sources a core node is able to support is limited by a parameter *MaxPassSize*. The hop distance between a core node and its Passive source is bounded by the parameter *MaxHop*. These two parameters are used to limit the number of core nodes and therefore to guarantee the basic robustness to produce moderate redundancy in routing paths (if the number of core nodes making up the mesh is too small, the redundancy in the mesh is not sufficient to guarantee reliable transmission).

An additional flag *CoreAcceptance* is included into the Join Query. It is set when the sending node (a multicast source) is able to support Passive sources; otherwise, it is reset. When an Active Source receives a Join Query from another source, it changes its status to Passive Source when all the following conditions are satisfied:

- The *CoreAcceptance* flag is set,
- Hop distance traveled by the Join Query is less than or equal to *MaxHop*,

- Its node ID (hereafter, this source node is called a *ToBePassive* source) is less than the node's ID that sent the Join Query (this node is called a *ToBeCore* source).

Then, this *ToBePassive* source sends *PassReq* to the *ToBeCore* source, setting the *CoreReq* field and put its own ID into the *Passive Source ID* field in the packet. The *PassReq* is forwarded to the *ToBeCore* node. If the *ToBeCore* node is able to support the Passive Source, it sends back a *Confirm* packet. It becomes the core of the Passive Source in forwarding multicast data packets to the network. If the counter of the supported Passive sources in a core node reaches the threshold *MaxPassSize*, in the successive Join Queries, the core node will reset the flag *CoreAcceptance* until the counter becomes less than *MaxPassSize*. When the *ToBePassive* source receives the *Confirm* packet, it changes its status from Active to Passive Source. It will no longer flood a Join Query until it becomes an Active Source again.

Passive Sources are required to refresh their status by sending *PassReq* to their core nodes in response to a flooded Join Query. If a core node fails to receive a *PassReq* from one of its supported Passive Source after sending a Join Query, it deletes the information stored in its memory concerning the Passive Source, and decreases the counter that counts the number of Passive Sources it is now supporting. If the counter reaches zero, this core node becomes an Active Source. On the other hand, if a passive node receives its core node's Join Query, which has traversed a hop distance larger than *MaxHop*, it changes its status to Active and sends a *PassReq* with the *CoreReq* flag reset to its core node. When the core node receives this packet, it deletes from its memory related information concerning the Passive Source, and decreases the counter that counts the number of Passive Sources it is now supporting.

DCMP reduces control overhead by reducing the number of the Active Sources. However, reducing the number of active sources may weaken the robustness of the mesh, as the redundancy provided by the mesh is proportional to the number of

sources in the network. If a Passive Source lies in between the receivers and its Core Node, sub-optimal routes for forwarding data packets to the receivers exist, leading to higher delay in the packet delivery. Moreover, instability may occur due to frequent changes in the status from passive to active, and vice versa. Note that a Passive Source becomes an Active Source when it receives a Join Query that has traversed a hop distance larger than MaxHop; an Active Source becomes a Passive Source whenever it finds a qualified Core Active Source. Now using an example, we explain how the instability may lead to inefficiency in transmission. Suppose a source, say node A, is a Passive Source supported by a Core Node, say node B. After a while, due to mobility, the node A becomes a Core Node. By sending its own Join Query, a mesh (containing node A) is extended by introducing a node, say node C, into the FG. Later on, it is possible that node A becomes Passive Source supported by another Core Source, say node D, after performing local computation on a received Join Query. Node A now does not send Join Query any more. However, node C continues to act as an FG node before its lifetime expires, leading to the transmission inefficiency.

2.5 Neighbor-Supporting Multicast Protocol (NSMP)

NSMP [19] adopts a mesh structure to enhance resilience and robustness against nodes' mobility. It also utilizes nodes' locality to reduce the overhead of route failure recovery and mesh maintenance.

Type	Sequence Number	Group Address	Source Address	Upstream Node	FC	NC
------	-----------------	---------------	----------------	---------------	----	----

Figure 2.4. Packet Header of NSMP

The packet header of NSMP packets is demonstrated in Figure 2.4. The field *Upstream Node* contains the address of the node which sends or forwards the current received packet; the field *FC* (Forward Count) denotes number of forwarding nodes along the path that the packet travels; and the field named *NC* (Non-forward Count)

is the number of non-forwarding nodes the packet passes. The forwarding nodes are those in the multicast mesh that forward packet from the sources to the receivers.

When a node becomes a multicast source, it first broadcasts a FLOOD_REQ packet. When an intermediate node receives the FLOOD_REQ packet, it updates its routing table using the information in the packet header. It updates the Upstream Node field with its own address and forwards the packet. When a receiver receives the first non-duplicate FLOOD_REQ packet, it stores the information of the packet header into a *ReqCache*, and waits for a delay for possible other FLOOD_REQs from the same source through other paths. For every redundant FLOOD_REQ received within the delay, the receiver computes the weighted path length, $(1-\alpha) \times FC + \alpha \times NC$, where $0 \leq \alpha \leq 1$ is the relative weight. It chooses the shortest (reverse) path to send a REP back to the source (if two paths have the same length, the one with more forwarding nodes is selected). Hence, NSMP balances routing efficiency and path robustness by giving preference to paths containing more forwarding nodes. Nodes on this reverse path become forwarding nodes of the group.

In this way, when there are multiple sources in the group, a multicast mesh can be constructed. Nodes in the multicast mesh are called mesh nodes. Others are called non-mesh nodes. Note that the global flooding is also performed by a group leader (the source with the smallest address value in a multicast group) at FLOOD_PERIOD interval to refresh the overall network in case of partitions exist.

All the nodes that detect the REP packets (except the mesh nodes) identify themselves as *neighbor nodes* of the group. Also, when a non-mesh node finds that one of its neighbors is a source, it becomes a neighbor node.

Neighbor nodes are utilized to reduce overhead when NSMP performs local route discovery. All sources send LOCAL_REQ at REQ_PERIOD interval, which is shorter than FLOOD_PERIOD, to recover from possible route failures. Forwarding of the LOCAL_REQs is performed by mesh nodes and neighbor nodes only, such that the propagation of the packets is restricted to areas near the mesh. Receivers

receiving the LOCAL_REQs should reply with REPs through the best path, which is calculated in the same way as sending a REP in response to a FLOOD_REQ. During the process that the REPs (in response to LOCAL_REQs or FLOOD_REQs) are transmitted back to the sources, the nodes' status, mesh node and neighbor node, are refreshed.

Compared to ODMRP, NSMP reduces control overhead by introducing the concept of neighbor nodes. Rather than performing global topology refreshing as ODMRP does, NSMP performs frequent local route recovery together with infrequent global topology refreshing in case of network partitions. This is a tradeoff between the efficiency and robustness. Also, NSMP tries to use paths that contain more existing forwarding nodes so as to improve route efficiency while retaining robustness. However, if the network is dense, i.e. each unit area contains many nodes, and many non-mesh nodes become neighbor nodes. Thus the efficiency improvement achieved by introducing local route discovery is limited. Moreover, NSMP proactively performs local route discovery, in which control messages are exchanged. On one hand, if the link breakages occur infrequently (e.g. in some cases, nodes are moving slowly or in local areas), these overheads are not necessary and thus they are not cost-efficient. On the other hand, if a receiver moves quickly to an area where no other mesh or neighbor nodes can be contacted, rather than reactively initiating route discovery towards the mesh, it has to wait for the next FLOOD_REQ (or possibly, LOCAL_REQ) to arrive. This reduces the number of data packets delivered.

2.6 Multicast Core-Extraction Distributed Ad hoc Routing (MCEDAR)

MCEDAR [20] is the extension to the CEDAR [21] architecture and provides the robustness of mesh-based routing protocols and the efficiency of tree based forwarding protocols.

CEDAR dynamically establishes a *core* using selected nodes of the network, and then incrementally propagates the link state of stable high bandwidth links to the nodes of the core. A set of nodes is elected in a distributed and dynamic manner to form the core of the network by approximating a *minimum dominating set* of the ad hoc network using only local information. Each node periodically broadcasts a beacon, which contains the information such as degree, effective degree (defined as the number of the sending node's neighbors who have chosen it as their dominator) and the ID of its dominator, pertaining to the core computation. Nodes that need to find a dominator select the highest degree node with the maximum effective degree in its first neighborhood as its dominator. Ties are broken by node ID. Considering the effect of the network mobility, if a node loses connectivity with its dominator, after listening to beacons from its neighbors, the node may either find its original dominator, or join another core that is dominated by another neighbor, or itself becomes a dominator. The broadcast of route probes to discover the location of a destination node, and the broadcast of some topology information amongst the cores is achieved by a *Core Broadcast* mechanism, which utilizes the cooperation between the MAC layer and the network layer.

MCEDAR uses two of the CEDAR's components: the architecture of core and the core broadcast. MCEDAR adopts the CEDAR's mechanism of electing core nodes. The graph consisting of all these core nodes is called a *core-graph*. MCEDAR extracts a *sub-graph* of the core-graph to function as the routing infrastructure. This sub-graph is a mesh topology (called *mgraph*) used to provide redundant paths, i.e. only core nodes compose the mesh for multicast packet delivery in a multicast group, as is described in Figure 2.5. When a non-core node wants to join a multicast group, it requests its dominator (core node) to perform the join operation. A core node initiates the join operation by core broadcasting a join request, *JOIN(Multicast group address, joinID)*, in which joinID contains the time a core node joins a multicast group and in this case, it is set to infinity. The JOIN message

is forwarded until it reaches a core node that has already joined the multicast group. This core node sends a $JOIN_Ack(Multicast\ group\ address, joinID)$ only if its joinID is smaller than the joinID contained in the arriving JOIN. The core node that sends a JOIN_Ack continues to forward the JOIN further down.

Figure 2.5. Data delivery in MCEDAR

Once the mgraph is extracted for a multicast group, data forwarding is done on the mgraph using the core broadcast mechanism. Core nodes in the mesh are in charge of forwarding the multicast packets they receive. When a data packet arrives at an mgraph member (a core node), the node attempts to forward the data packet only to those nearby core nodes that are also members of the same mgraph. This forwarding mechanism eliminates redundant transmissions and hence saves bandwidth while improving reliability when the nodes are very mobile.

The advantage of MCEDAR is that nodes are grouped into cores, each one of which is dominated by a core node (dominator). These core nodes together construct a multicast mesh for delivering data packets from one core to the multicast receivers in the other cores. A special core broadcast is exploited to suppress unnecessary data

forwarding between the core nodes.

Nonetheless, nodes need to broadcast beacons periodically, which may collide with each other. Failure to receive the beacons may trigger the *ping-pong effect* in electing core nodes, i.e., the reselection of core nodes is liable to occur from time to time. The core-broadcast builds a source-based tree implicitly in the mgraph. Based on the joinID (the time a core node joins the group), this tree defines a global ordering of the members on the mgraph. Consequently, any changes in the joinID of one mgraph member can potentially trigger a cascade of joinID changes in the entire mgraph.

2.7 Multicast routing protocol based on Zone Routing (MZR)

MZR [22] is a source-initiated, on-demand multicast routing protocol. It makes use of the zone construction and query mechanism of ZRP (*Zone Routing Protocol*) [23] to build a source-based multicast tree. Each node constructs a zone around itself with limited zone radius measured in hop count. Every node periodically broadcasts an *ADVERTISEMENT* packet identifying itself. The TTL value of the packet is set to the zone radius. When a node receives the *ADVERTISEMENT*, a route entry for the sending node is created in its zone routing table. A soft state is associated with the route information such that the stale routes will be purged. A route entry expires if the *ADVERTISEMENT* from a certain node is not periodically refreshed. Using this method, every node gets to know the nodes in its defined zone.

A multicast source initiates the creation of the multicast tree. First, the source forms a multicast tree in its zone by sending a *TREE-CREATE* to each node in its zone through the unicast routes obtained from its zone routing table. Reverse route entries are created in respective nodes as the *TREE-CREATE* propagates in the zone. When a multicast receiver inside the zone of the source node receives the *TREE-CREATE* packet, it replies by sending a *TREE-CREATE-Ack* back. Nodes along the path that the *TREE-CREATE-Ack* travels back to the source become the members

on the multicast tree in the zone. As illustrated in Figure 2.6, node S initiates the tree creation by sending TREE-CREATE packets to all the nodes in its zone. Node A, B and D are multicast receivers and they respond to node S with TREE-CREATE-Ack messages to set up the tree branches. Node C becomes a node on the multicast tree branch, connecting node S and D, although node C is not a multicast group member.

Figure 2.6. Multicast tree creation inside a zone

Secondly, the source tries to spread the multicast tree to the entire network. Thanks to the zone routing table, the source is able to identify all the *border nodes* (border nodes are those whose hop distance from the zone center node is equal to the zone radius) in its zone and unicasts a *TREE-PROPAGATE* packet to each one of them. When a border node receives it, even though it may not be a multicast receiver, the border node must send a TREE-CREATE packet to all its zone nodes. If a node in the border node's zone is a multicast group member, a TREE-CREATE-Ack is sent as a response. The source node's border node, in turn, unicasts TREE-CREATE-Ack packets to the multicast source. Then, the border node sends TREE-PROPAGATE messages to its own border nodes to propagate the tree creation. This procedure is continued until the TREE-CREATE packets are spread to the whole network. As demonstrated in Figure 2.7, the source node S initiates the multicast tree creation, and the border nodes E, P and N extend the tree inside their respective zones and to the rest of the network.

Figure 2.7. Multicast tree propagation to the entire network

When the multicast tree is established, multicast data packets are transmitted from the source along the tree branches to the multicast receivers.

In MZR, downstream nodes are in charge of detecting link breakages and reconfiguring the tree. By checking the changes in its neighbor table, a downstream node can easily detect the breakage. The downstream node initiates a global search for the multicast tree by using the zone routing mechanism. It first sends a *JOIN* packet to all its zone nodes with the TTL set to the zone radius. If any of the nodes in its zone is on a multicast tree, a *JOIN-Ack* is replied to set up the new tree branch. Otherwise, if no reply is received, the downstream node propagates the join procedure by sending a *JOIN-PROPAGATE* to all of its border nodes, which in turn send their own *JOIN* packets to the nodes in their zones. If they get a reply from any of their zone nodes, a *JOIN-Ack* is sent back to the node that initiated the join process. Otherwise, the *JOIN-PROPAGATE* is transmitted again to propagate to the entire network until a reply is received in response.

MZR enjoys the feature of propagating requests via border nodes so that the procedures can be performed quickly and efficiently. This is based on the correct

zone routing information each node collects by receiving others' ADVERTISEMENTS. Nevertheless, if the nodes are moving fast, the information concerning the zone may turn out to be obsolete rapidly, resulting in less efficient propagation of the messages. On the other hand, in order to keep the zone routing information fresh enough, the frequency of the ADVERTISEMENT beacon should be high, leading to high control overheads. Also, with the number of the nodes in the network increased, even though the flooding of the beacon message is hop limited, control overhead is still tremendous. The tree that is created in MZR also suffers from numerous link breakages as in other tree-based protocols, and therefore requires great efforts in repairing the broken links.

2.8 Adaptive Dynamic Backbone Protocol (ADB)

ADB [24] adopts a *two-tier* topology with an adaptive backbone infrastructure to integrate the robustness of the flooding scheme and the efficiency of the tree-based scheme. All the backbone nodes make up a mesh. And each backbone node also acts as the root for the local tree.

ADB selects the backbone nodes by relaxing the dominating set property. Even when it is more than one hop away from the closest backbone node, a node is allowed to attach to the backbone if the *accumulated link failure frequency* from that node to the backbone does not exceed the specified threshold, i.e. a node decides whether to associate with a backbone or not based on the dynamics it senses from the environment. The metric "accumulated link failure frequency" is used to measure the dynamic condition of a node's surrounding area.

In the neighbor discovery process of the ADB, when first started, each node sets itself to be a backbone node (core). Each node needs to periodically broadcast a hello message to let others discover itself in the surrounding area. In this hello message, a parameter, *nlff* (the accumulated *Normalized Link Failure Frequency*) is included. The *nlff* carries the accumulated NLFF of every hop on the path from the sending

node to the core. It is defined as,

$$nlff_{curr} = \frac{\# \text{ of link failures detected}}{NLFF_TIME_WINDOW \times (\# \text{ of neighbors})}$$

$nlff = \alpha \cdot nlff_{curr} + (1 - \alpha)nlff$, where α is the smoothing factor lying between 0 and 1.
Initially $nlff$ is set to zero.

The core selection process in each node begins after the node has started for some time and has got enough information of its surrounding area. Then, by calculating a height considering the value of $1/nlff$, the degree and the ID of the node itself and the nodes around, each node decides whether it should continue to serve as a core, or become a child of an existing core. If a node has the largest height among its known neighbors, it is considered as a local optimal node and therefore should continue to act as a core. Otherwise, it chooses the local optimal node as its core and memorizes it as its parent. Hop limit and the upper threshold value for $nlff$ are used to constrain the range of the local group originated from a core node. If either of these two metrics in a received hello message (from a node's parent node) exceeds its threshold, the receiving child node must either choose another node as its parent or itself becomes a backbone node. Under different mobility conditions, this process will generate different number of cores and different depths of local trees.

Cores get the route information to other nearby cores from their child nodes. Child nodes are required to periodically update routing information of other cores from the received hello messages and inform their own cores of it.

In order for a node to join a multicast group, it must send a *Join Request* message to its parent in the local group. If a receiving parent is not already on the multicast forwarding tree, it forward the request towards the direction of the core and marks itself as a forwarder. Otherwise, it simply ignores the request. In this way, a multicast tree is created in the local group, and the data forwarding is performed by the nodes marked as forwarders. The multicast forwarding status must be periodically refreshed to deal with mobility, node failures and nodes' leaving of the multicast group.

Data forwarding in the local group is performed solely by nodes marked as the *multicast group forwarders*. Every time a forwarder receives a data packet, it simply forwards the packet to surrounding neighbors that are group members or forwarders performing a local broadcast of the packet.

When a core receives a data packet, it must forward it to other cores on the backbone. By utilizing the routing information provided by its child nodes, a core node encapsulates the received packet into a *CORE_FWD* message and forwards it to every other core nodes it knows.

ADB establishes a hierarchical topology for multicast routing, which takes the surrounding nodes' mobility into account by sensing accumulated link failure frequency. This mechanism handles the mobility well in the sense that different mobility results in different depth of the local tree and thus different reactive speeds to fix the link breakages. However, as the backbone selection process is based on a beacon mechanism, failure in receiving the beacon message will generate instability in the backbone structure. This instability produces inaccurate routing information being collected, which eventually leads to frequent switching of core responsibility between the nodes. As ADB adopts the tree topology for multicast packet delivery, changing of the core may require the multicast member to rejoin the multicast group by sending a Join Request again. In a highly mobile network, this kind of instability refreshes the multicast topology either partially or entirely, resulting in either low packet delivery ratio or low transmission efficiency.

2.9 Adaptive Demand-Driven Multicast Routing (ADMR)

ADMR [14] is an on-demand multicast routing protocol that attempts to reduce the proactive protocol components. Multicast routing state is dynamically established and maintained only for active groups and only in nodes located between multicast senders and receivers.

There are two types of floodings defined in ADMR: *tree flood* and *network*

flood. Tree flood is performed only by nodes that are on a source specific multicast tree, while network flood is carried out by all the nodes in the network. The two types of floodings are illustrated in Figure 2.8.

Figure 2.8. Tree flood and Network flood

When a node S wants to send data packets to a multicast group G, it sends a packet with an ADMR header as a network flood. Each multicast packet originated by node S for the multicast group contains this header, which includes a sequence number uniquely identifying a packet, hop count specifying the distance a packet has traversed, the previous hop address used for the receiving node to know its upstream node's MAC address and inter-packet time used by nodes to monitor (or estimate) the receipt of new data packets. Every node that receives the packet from the source should memorize its upstream node's MAC address by checking the previous hop address field in the ADMR header. This information may be used soon in setting up the multicast tree branch. A multicast receiver should send back a *Receiver Join* message to the upstream node specified in the ADMR header. The upstream node in turn forwards the *Receiver Join* to its next hop node leading back to the source and sets itself as a tree node. The next hop node towards the source is then its upstream node on the tree.

After this process, a multicast tree having a multicast source as its root is constructed. However, this tree is *loosely structured*. Multicast data packets are

propagated using tree flood, such that packets are forwarded only among nodes belonging to the multicast forwarding tree. When a node receives such a packet, it checks the source ID and its Membership Table entry for this group to determine whether it should forward the packet. If it decides to forward, it just broadcast this packet in its local area. Otherwise, this packet is discarded. The packet thus flows along the tree from the sender to the group receivers but is not constrained to follow specific branches in the tree. This strategy enables a packet to be forwarded automatically around temporarily broken links or failed forwarding nodes. As described in Figure 2.9, two packets, packet X and Y, are sent from node S to the two receivers, node R1 and R2, via intermediate tree nodes. Due to a link break occurring between node B and node R1, packet Y cannot be propagated from node B to node R1. Nevertheless, as data receipt is not restricted to a certain branch, node R1 is still able to receive the packet Y from a nearby tree node, D, on another tree branch.

Figure 2.9. Loosely structured multicast tree in ADMR for data delivery

Absence of packets within a given period, which is a multiple of the inter-packet time, is an indication of forwarding tree disconnection. For example (see Figure 2.10), when a forwarding node C, for source S and group G, does not receive packets from S via its former upstream node B within the multiple of the inter-packet time, it performs a local repair procedure to reconnect to the multicast tree. It first sends a *Repair Notification* packet to other downstream nodes, H, D, R1 and R2, on the current tree branch. Unlike the forwarding of data packets, the forwarding of the

Repair Notification is strictly along the tree branch. The Repair Notification serves two purposes: first is to prevent the downstream nodes from initiating the repair mechanism by telling them that a repair has already begun upstream; and second is to check again whether the link to the upstream tree node is really broken. After sending the Repair Notification, node C waits for a duration before proceeding with its local repair. If after the duration, no Repair Notification is sent from its parent node on the tree, node C sends a hop-limited *Reconnect* packet as a form of network flood. This Reconnect packet either will be broadcast to the source if the receiving nodes are not aware of the route back to a multicast source, or will be unicast to a multicast source if the receiving node knows the route to the multicast source (as in Figure 2.10, the Reconnect packet is forwarded to S via nodes F and A). If the Reconnect packet reaches the source, it replies with a *Reconnect Reply* packet, which is unicast back to the repair node C along the path that the Reconnect has traveled. This process is illustrated in Figure 2.10.

Figure 2.10. Local sub-tree repair

ADMR employ a loosely structured multicast tree for data delivery. Data receipt is not restricted to a specific branch. This feature makes the data delivery resilient to mobility. Failures in receiving packets from a parent node will not affect the overall delivery, and thus, its implementation is also simple. On the other hand, as has been mentioned before, a tree topology is ultimately not suitable in a highly dynamic network, as link breakages occur frequently due to the mobility, leading to high control overheads in repairing the broken links. In ADMR, only an affected

downstream node tries to repair the link break. This requires the node to wait for a certain amount of time before initiating its repair procedure so that no other upstream or downstream node on the same branch initiates the repair at the same time. The disadvantage is that the responsibility of repairing a sub-tree lies on a single node. When the traffic load is heavy, a lot of the data packets may be lost due to the waiting period involved. Secondly, the repair is not local, even though when a node, which receives the Repair Notification packet, has the route information back to the source. The source, after receiving the Repair Notification, will try to find a new path leading to the repairing node. This two-way transmission (from the repairing node to the source and then backwards) is time consuming. Also, the control overheads are high because of the broadcast plus unicast scheme when link breakages are occurring often.

2.10 Mobility-Based Hybrid Multicast Routing (MHMR) Protocol

MHMR [25], [26] uses mobility-based clustering and hierarchical structure. The mobility computation utilizes the mobility information (individual mobility, group mobility) and the position information obtained via GPS [27].

The clustering method is based on the combination of the mobility computing and the distributed clustering algorithm. Each node is assumed to be able to get its mobility and position information through GPS and periodically disseminates the information to its neighboring nodes. A node M upon receiving node N's velocity information, calculates the relatively speed between itself and node N. A node with the lowest ID among the nodes, from which the node M is able to receive the periodic updates of mobility information, and whose mobility relative to node M is under the predefined threshold, is selected as a tentative *cluster head*.

All the cluster heads then build a mesh amongst them to provide robust packet delivery. Before a node is able to send or receive data packets of a multicast group, it

must join the group. To achieve this, the node requests its cluster head to join the multicast group. If the cluster head is not a member of the group, it advertises a *Join Request* to its neighboring cluster heads using a mechanism similar to ZRP [23], i.e. each cluster head sends the Join Request only to its border nodes to limit the propagation (the border node information can be proactively maintained by each cluster head). When an existing multicast cluster head member receives the Join Request, it responds to the source cluster head, which is the cluster head that initiates the Join Request, with a *Join Reply*, creates or updates the entry in its multicast member table, and forwards the Join Request further down to other cluster heads. The Join Request from the source cluster head should be received regularly by all the existing multicast cluster heads, otherwise, the entry corresponding to it will be deleted on expiry. A cluster head can leave the multicast group by ceasing to send the Join Request.

For multicast data forwarding, a source-based tree is created dynamically over the mesh made up by the cluster heads. Data packets are first sent to the cluster head, which dominates the source node, and then, they are forwarded by the cluster head to its neighboring cluster heads. If a downstream cluster head receives a packet from one of its upstream cluster heads, it will inform its other upstream cluster heads not to forward any more packets to it by sending a *Forward Reject* message. When the other upstream cluster heads receive the Forward Reject, they do not forward data packets any more to this downstream cluster head in the next time period (this time period depends on the network stability). Using this method, a tree can be dynamically created to delivery data packets.

MHMR tries to create a stable underlying topology for multicast using the concept of relative mobility. The calculation is based on the information (speed, direction and position) obtained from GPS. Other mechanisms, multicast topology building and data packet delivery, are performed based upon this relatively stable topology. In order to let every node gather up-to-date information of each other, the

broadcast of the mobility and position information is unavoidable, and consequently the processing overheads are high. In the data delivery, if a downstream cluster head receives data packets from an upstream cluster head, the downstream one explicitly asks the other upstream cluster heads to stop forwarding data packets to it for a following period of time. The purpose of this mechanism is to reduce redundant data packet transmission and hence improve transmission efficiency. However, the relative mobility calculated here is only within the cluster, but not between the cluster heads. The time period for the upstream cluster head to keep silent is hard to decide based upon the overall relative mobility because cluster heads are generally more than one hop away, and the calculation of their relative mobility is not directly related to the deduction of the silent period.

2.11 Ad Hoc Multicast Routing (AMRoute) Protocol

AMRoute [28] improves the robustness and efficiency by introducing two features: a) building a user-multicast tree, where packets' replication and forwarding is only performed by group members over unicast tunnels; b) updating core nodes dynamically according to group membership and network connectivity.

The user-multicast tree, which includes only the multicast group senders and receivers, is based on a mesh that encompasses all the group members. Each group has at least one core node that is responsible for constructing the multicast topology. Each group member begins by identifying itself as a core of a one-node mesh. It broadcasts JOIN_REQ packets with increasing TTL (using expanding ring search) to discover other group members or meshes. When a member receives a JOIN_REQ from a core of another mesh, the nodes responds back with a JOIN_ACK. In this way, a bi-directional tunnel is established between the core and the responding node of another mesh. The number of core nodes is reduced by letting the node with a higher IP address become the core of the newly merged mesh.

When the mesh is constructed, a core node starts to build (or refresh) the

spanning tree by sending periodic TREE_CREATE messages along all the mesh links (i.e., the unicast tunnels) attached to it. Group members receiving non-duplicate TREE_CREATEs forward them on all mesh links except the incoming one, and mark the outgoing and incoming links as *tree links* of the multicast group. Data packets can then be forwarded through these tree links. Figure 2.11 shows an example of the user-multicast tree. Node A is the core node. It constructs a multicast tree extending to node B, C and D via the virtual link A-B, A-C and A-C-E-D.

Figure 2.11. User-Multicast Tree

However, due to nodes' mobility, it is possible that data packets may reach a link that does not contain any group members. In this case, a TREE_CREATE_NAK will be sent back along the incoming link. On receiving a TREE_CREATE_NAK, a group member marks the incoming link as a *mesh link* (not a tree link). Thus each non-core node considers the link along which a non-duplicate TREE_CREATE message was received and every other link along which no TREE_CREATE_NAK message was received to be part of the tree for a specific group.

If a node leaves a group, it sends out a single JOIN_NAK message to its neighboring nodes. If it subsequently receives any data or signaling message for that group, it can send out further JOIN_NAK messages. It is also possible that nodes' leaving the group results in splitting of the mesh.

Indeed, each member expects to receive periodic TREE_CREATE messages. In

case this message is not received within a specific period, the node designates itself to be the core after a random time. The node, whose timer expires the earliest, succeeds in becoming the core and initiates the processes of discovering other disjoint meshes as well as of tree creation. Multiple cores that may arise in this case are resolved by the core resolution procedure, i.e. the node with the highest IP address wins the competition.

The advantage of AMRoute is that since the user-multicast tree is an overlay tree based on the underlying mesh, each group member is aware of its neighbors (other group members) on the tree only; multicast state is maintained by the group members only, and is not required to be maintained by other nodes; also, the logical tree topology need not be changed frequently due to the mobility, which improves robustness and reduces signaling overhead. Nevertheless, the overlaying tree can also lead to reduction in packet forwarding efficiency and may increase the end-to-end delay. The reason is, it is very difficult to let multiple unicast tunnels share the same physical links, which results in redundant traffic on the physical links. As shown in Figure 2.11, although the data packets can be forward from A to D through node C, due to the separate unicast tunnels constructed from A to C and A to D, the same data packet is sent to C and to D in two different transactions, which causes inefficiency.

2.12 Applying mobility prediction in ODMRP using GPS

In [13], the adaptation of the frequency for sending the Join Query via mobility prediction has been specified, and all the nodes are supposed to be equipped with GPS. The location and movement information obtained via GPS is utilized to predict the duration of the time that routes will remain valid. With the predicted time of route disconnection, Join Queries are flooded only when route breaks are imminent. The rationale is that as soon as a single link on a path is about to get disconnected, the entire path is invalidated, and the path should be rebuild.

When a source sends a Join Query, it appends its location, speed, and direction. It sets the *MIN_LET* (*Minimum Link Expiration Time*) field to the predefined value *MAX_LET_VALUE*. The next hop neighbor, upon receiving the Join Query, predicts the link expiration time between itself and the previous hop by calculating their relative speed and the distance between them. The minimum between this value and the *MIN_LET* indicated by the Join Query is included in the packet. When a multicast receiver receives the Join Query, it predicts the LET of the last link on the path. The minimum between the last link expiration time and the *MIN_LET* value specified in the Join Query is the *RET* (*Route Expiration Time*).

The multicast receiver can choose a route that is the most stable (i.e., the one that will remain connected for the longest duration of time). To do this, it must wait for an appropriate amount of time after receiving the first Join Query so that all possible routes and their LET will be known. The receiver then chooses the most stable route, which has the largest LET, and broadcasts a Join Reply. The RET value is enclosed in the Join Reply that is broadcast. If a forwarding group node receives multiple Join Replies with different RET values, it selects the one with the minimum RET and sends its own Join Reply. When the source receives Join Replies, it selects the minimum RET amongst all the Join Replies received. Then the source can build new routes by flooding a Join Query before the estimated route break occurs, which is specified by the minimum RET. Of course, the lower bound and the upper bound for flooding a Join Query should be pre-defined so as to prevent too short or too long refresh interval from being applied.

By calculating the route expiration time, the sources are able to adjust the frequency of refreshing the entire topology. However, in order to get the “best” route, almost every node along the path, from a receiver back to the source, should wait for a certain amount of time (at a receiver, it should wait for various Join Queries to know the route with the largest RET; at a potential forwarding group node, it should wait in order to reduce the redundant forwarding of Join Reply with different RET

values; at the source side, it should wait for all possible Join Replies to choose the route with the least RET). Furthermore, as nodes are moving randomly, an instability (large relative speed between two neighboring nodes) on a single hop affects the LET and RET of the entire route, i.e., the entire path is invalidated due to the instability on a single hop. Thus, in practice, choosing routes based on relative mobility may not help a lot in the sense of adjusting the Join Query frequency. In fact, the frequency for flooding the Join Queries is always high due to the randomness in the mobility.

2.13 Summary

The protocols are summarized in Table 2.1, which have been reviewed in this chapter.

It is clear that in flat topologies, there are two basic topologies: the tree and the mesh. The tree topology used in MAODV, ADMR, *AMRIS (Ad Hoc Multicast Routing Protocol Utilizing Increasing ID Number)* [28] and *LAM (Lightweight Adaptive Multicast)* [30], etc., enjoys the characteristic of higher transmission efficiency as the data packet transmission is performed mainly by the tree members. However, the tree suffers from frequent link breakages in a mobile environment. Excessive control overheads are needed in repairing the tree branches. On the other hand, the mesh topology, used in ODMRP, PatchODMRP, DCMP, and *CAMP (Core-Assisted Mesh Protocol)* [31], etc., has moderate redundancy to cope with the link breakages. However, the excessive redundancy adversely affects the transmission efficiency, making the multicasting similar to the flooding.

In order for the multicast protocol to be scalable, hierarchical topology has been introduced in MCEDAR, ADB, MHMR, *AMRoute (Ad Hoc Multicast Routing Protocol)* [28], and *PAST-DM (Progressively Adapted Sub-Tree in Dynamic Mesh)* [32], etc. In a hierarchical topology, nodes are organized into local groups or clusters. Either a mesh or a tree based on the mesh is constructed among the local groups or

clusters. The mesh is used to provide the fundamental redundancy for packet delivery to the local structures, and then the tree is usually applied in the local groups or clusters so that transmission efficiency can be achieved in local structures.

In constructing the local topology, many protocols depend on the beacon mechanism for nodes to get the knowledge of each other's existence. The grouping or clustering is then performed based upon the received information. In a network that is relatively less dynamic or that has light traffic load, this kind of proactive scheme is able to work reasonably well. However, when the network is highly mobile or is of high density (many nodes located in a relatively small area) or the traffic load is high, the beaconing mechanism will not work satisfactorily, as either it has high control overheads in updating the topology, or frequent flooding of beacon messages may potentially produce contention, congestion and collisions. As a result, nodes get incomplete information of their surrounding areas, which brings instability in the formation of the local groups or clusters. This instability, in turn, results in imprecise routing information, which adversely affects the packet deliveries.

Utilizing the mobility information, such as speed and direction, may be promising in the MANETs, as done in MHMR and *PBM (Position-Based Multicast Routing)* [33]. It may enhance the protocols' robustness and efficiency. The mobility and location information can be provided by GPS or any other equipment. However, if the location service is included, extra control overheads in finding the locations of the receivers should not be neglected when the number of receivers is large. Also, the propagation of the location information of individual nodes throughout the network produces large overheads. Consequently, using only the relative mobility information seems to be a simple and effective way to tackle the inherent mobility between the nodes.

Table 2.1. Brief Summary of Reviewed Multicast Routing Protocols

Protocol	Topology	Dependency on Unicast Protocol	Periodic Topology Refresh	Main Disadvantage
MAODV	Tree	Yes	No	Tree links liable to break in high mobility
ODMRP	Mesh	No	Yes	High control overhead
Patch-ODMRP	Mesh	No	Yes	Patching introduces extra FG nodes that lead to reduced efficiency
DCMP	Mesh	No	Yes	Sub-optimal routes and reduced robustness under mobility
NSMP	Mesh	Yes	Yes	High control overhead in dense networks
MCEDAR	Hierarchical	Yes	Yes	Topology instability likely
MZR	Hierarchical	No	Yes	High control overhead
ADB	Hierarchical	No	Yes	Topology instability likely
ADMR	Loosely-structured tree	No	No	Time-consuming link repair mechanism
MHMR	Hierarchical	No	Yes	Relative stability calculation not accurate for multi-hop case
AMRoute	Tree over mesh	Yes	Yes	Reduced packet forwarding efficiency

Chapter 3. The Gateway-Based Multicast Protocol (GBMP)

Multicasting is able to improve the efficiency and robustness of distributed systems and applications such as those in the MANETs. Multicasting efficiently utilizes the limited network resources while performing group communications. However, performing multicasting in Mobile Ad hoc Networks (MANETs) is challenging due to the networks' unique characteristics such as frequently varying topology, narrow bandwidth, moderate quantity of memory storage, finite battery power, and relatively lower processing ability, etc. Though much work exists in literature focusing on multicast protocols for MANETs, improvements, such as increasing transmission efficiency and reducing control overhead, are still possible.

In this chapter, we describe our proposed multicast protocol, called Gateway-Based Multicast Protocol (GBMP), for the MANETs and study its performance through extensive simulations. In GBMP, the multicast model we use is *Any-Source Multicast (ASM)* [34]. In this model, an IP datagram is transmitted to a host group (multicast group) which consists of a set of zero or more end-hosts (receivers and possibly other senders) identified by a single IP destination address (multicast address); it supports one-to-many and many-to-many communications; end-hosts may join or leave the group at any time; there is no restriction on their location or

number; any end-host may transmit to a host group, even if it is not a member of the group; also, the receivers in a multicast group receive data packets from all the sources in the group.

GBMP has the following features: 1) it constructs a group-shared loosely structured multicast tree for data forwarding to obtain moderate topology redundancy; 2) it applies the Passive Receive Mode (PRM) to improve transmission efficiency, rather than applying a ‘pruning’ mechanism as other protocols do; 3) it employs a novel Bi-directional Local Link Repair (BLLR) strategy to repair broken links, i.e., link repair is performed locally from upstream and downstream directions in a cooperative manner. In order to measure the discontinuity in data packet delivery, we also introduce a new metric called Weighted Occurrence of Consecutive Packet Loss (WOCPL). The following terminologies are used in describing the protocol:

Local group: is a set of multicast receivers and Forwarding Group (FG) [16] nodes, which are close to each other (e.g. within 3 hops), for a long enough period of time.

Multicast group leader: is the multicast source node that has the lowest ID among all other multicast sources in a multicast group.

Gateway node: is a multicast receiver node that receives a broadcast message (Global Join Query) from the multicast group leader first within its neighborhood. It is also the root of the local group-shared tree.

Normal node: a node that is neither a multicast source node, nor a multicast receiver node, nor a FG node.

3.1 Data Structures

The following data structures are used to describe the GBMP protocol:

- **Multicast Source Table:** stores all the multicast source IDs for every

multicast group a node belongs to.

- **Route Table:** stores the routing information.
- **FG node status Table:** stores the status of whether a node is a FG node or not for a certain multicast group.
- **Responsibility Table (Rsp Table):** stores all the downstream nodes' IDs that a node is responsible for their receipt of data packets.
- **Monitor Table (MNT Table):** stores IDs of all the downstream nodes whose data packet receipt status a node is monitoring.

3.2 Protocol Description

3.2.1 Multicast Topology

GBMP builds a *group-shared multicast tree* rooted at the multicast group leader, which is the source that has the lowest ID among all the sources in a multicast group. The intermediate nodes on the tree forward only those data packets of the multicast groups in which they are involved. In the following text, we name these nodes also as the Forwarding Group (FG) nodes.

Each multicast source must broadcast a *Global Join Query (GJQ)* when it wants to start sending data to a multicast group. If this multicast source is the multicast group leader, the GJQ triggers the construction or refreshing of the multicast topology. The format of GJQ is as follows:

Global Join Query (Message Type, Multicast Group ID, Global TTL, Local TTL, Sequence Number, Gateway Node ID)

In the GJQ, the “Local TTL” field is initialized with a pre-defined value *INVALID_TTL*, and is used to limit the range of a local group being set up. When a receiver receives a GJQ originated from the multicast group leader, and if the value of this field is equal to *INVALID_TTL*, the receiver is eligible to become a *Gateway*

Node (GN) of a local group. It starts the construction of a local group by setting the “Local TTL” to *MAX_LOCAL_GROUP_RANGE*, which is the range of a local group (the default value is 3)⁴; and by putting its own ID into the “Gateway Node ID” field, thereby claiming that it has become a GN. It then forwards the GJQ. After that, the GN sends a *Global Join Reply (GJR)* back to the source. The format of GJR is as follows:

Global Join Reply (Multicast Group ID, Source ID, Next Hop ID)

Every intermediate node receiving the GJR checks whether its own ID is equal to the “Next Hop ID” field in the GJR message. If this is the case, the node becomes an FG node, and then it chooses its own next hop address to forward the message until it reaches the multicast source.

If in the GJQ initiated by the multicast group leader, a receiver finds that the “Local TTL” is less than or equal to *MAX_LOCAL_GROUP_RANGE*, it knows that it is within the range of a local group. Then, the receiver stores the ID of the GN and the path towards it in the route table, decreases the “Local TTL” by one and broadcasts it. If the value of the “Local TTL” is equal to zero, which means this GJQ has reached the border of a local group, then before broadcasting the GJQ, it is reset to *INVALID_TTL*.

Receivers in a local group should send a *Local Join Reply (LJR)* to its GN in order to set up the local group-shared tree. Nodes along the path, which are traveled by the LJR back to the GN and are specified by the Next Hop ID field in the message, become FG nodes of the multicast group. The format of LJR is as follows:

Local Join Reply (Message Type, Multicast Group ID, Gateway Node ID, TTL, Sequence Number, Next Hop ID)

The process of constructing a local group is illustrated in Figure 3.1. Nodes A,

⁴ A local group with too many hops is hard to maintain; while with too few hops, there will be too many local groups in the network, which defeats the purpose of the hierarchical topology.

B, E and F are receivers. First, node A receives a GJQ originated from the multicast group leader, S, through the upstream node H. Node A finds that the Local TTL field is equal to INVALID_TTL. It immediately sends back a GJR to S via H (H then becomes a FG node); it forwards the GJQ downstream to B and C, which also forward it further down. Because B is a receiver, it replies to A by sending back a LJR. So do nodes E and F. The forwarding of the LJRs back to A makes nodes C and D becoming FG nodes. Note that A will not forward any LJRs to H. Therefore, A becomes the GN for its downstream nodes B, C, D, E and F; also, the route from S to A is established.

Figure 3.1. Local Group Construction

In between two successive GJQs from the multicast group leader, the GNs should maintain their local groups by sending *Local Join Query (LJQ)*. The format of LJQ is

Local Join Query (Message Type, Multicast Group ID, Gateway Node ID, TTL, Sequence Number,)

The nodes, which belong to a local group, should respond by sending LJR to their GN.

Hence, the global group-shared tree is further divided into *local trees (local groups)*, which are the sub-trees rooted at the GNs. The introduction of the local tree concept has the following advantages: 1) The GNs maintain the local trees, making the interval of maintaining the global topology to be longer, and therefore save

control overheads; 2) When a receiver is eligible to become a GN, it sends back the GJR to the source at once (rather than waiting for its downstream nodes' GJR or LJR and then sending its own GJR). This accelerates the construction of the multicast tree and shortens the delay between the multicast topology construction and the sending of the succeeding data packets from a source. Figure 3.2 presents an example of multicast topology of GBMP after the global and local trees have been constructed.

Figure 3.2. A Possible GBMP Multicast Topology

3.2.2 Selection of Multicast Group Leader

All the sources should compete to become the multicast group leader. When a source wants to send data packets to the network, it must first try to broadcast a GJQ. If it receives a GJQ from another multicast source that has a lower ID, it cancels broadcasting its own GJQ and forwards the received ones. Like the receivers, the sources (other than the multicast group leader) must send either Global Join Reply (GJR) or Local Join Reply (LJR) in response to the GJQ to take part in constructing the tree.

If a source successfully receives GJRs in response to its GJQ, it becomes the multicast group leader. It will then broadcast the GJQ regularly to refresh the group-

shared tree.

As the other sources also take part in the multicast topology construction, they do not have to send their own GJQs. However, if the next GJQ is missing or a *Source Leave* message from the multicast group leader (which indicates that the multicast group leader will stop sending data) is received, the sources will again compete for the responsibility of the multicast group leader.

If, in-between two GJQs from the multicast group leader, a new source wants to send data packets to the multicast group, it broadcasts a GJQ. If its ID is smaller than the existing multicast group leader, this source becomes the multicast group leader and the multicast topology is rebuilt. Otherwise, when this GJQ reaches either a group member or an FG node, say node A, then node A unicasts a *Join Ack* message to the new source. When the new source receives the Join Ack message, it unicasts back a *Join Notification* message to set up the FG nodes along the path.

3.2.3 Data Delivery

We choose the same strategy as described in the ADMR for data delivery within the multicast tree, i.e., we do not restrict that the packet delivery must be on the specific branches. A node may receive data packets from any branch. As illustrated in Figure 3.3, node C can receive a multicast data packet from either A or B, depending on from whom the packet arrives first (i.e. each multicast data packet is forwarded from S along the shortest-delay path to the receiver C). When a node receives a data packet, it checks its “FG node status Table”. If it is an FG node of the group, it just forwards the received packet. Hence, the packet flows along the tree branches from the sources to all the receivers.

Figure 3.3. Multicast Data Delivery

3.2.4 Passive Receive Mode (PRM)

In order to improve the transmission efficiency, we propose the Passive Receive Mode (PRM). The PRM is different from the *pruning* mechanism (as employed by ADMR, in which a FG node stops forwarding data packets when it has no downstream node). Pruning is effective in increasing forwarding efficiency when the network is static or the mobility is very low. Nevertheless, when the mobility is high, FG nodes may need to forward packets to various newly-arrived downstream nodes from time to time. Therefore, the control overhead incurred by performing first pruning and then attaching (i.e. becoming an FG node again) repeatedly can be very high. However, in the PRM, if there is temporarily no downstream node to forward data packets to, an FG node forwards one data packet every few seconds while remaining on the multicast tree. Whenever required, the FG node can resume forwarding every data packet.

To implement this, in each data packet header, we add three fields: *last node ID*, *previous node ID* and a Boolean variable *InPRM*. The “last node ID” contains the ID of the node that sends the current data packet. The “previous node ID” contains the ID of the node from which the last node (specified by the ‘last node ID’ field) receives this data packet. If “InPRM” is set to FALSE, it means that the node which sends this packet is not in PRM; otherwise, it is in PRM and this packet is used as a

heartbeat (which is explained later).

The PRM utilizes the *Passive Acknowledgment (PsvAck)* technique [59]. As shown in Figure 3.4, when an FG node, say B, receives a non-duplicate data packet, it moves the content of the data header field “last node ID” to the field “previous node ID”, puts its own ID into the “last node ID” and then forwards this data packet. When its upstream node, A, receives the PsvAck (a duplicate data packet) from B and if in the “previous node ID” it finds its own ID, this implies that A is responsible for the data receipt at B. Then A stores the “last node ID” (which contains B’s ID) in its RSP Table, and discards the packet. Otherwise, it stores the “last node ID” in its MNT Table (the use of MNT Table will be illustrated in Section 3.2.5.1).

Figure 3.4. Passive Acknowledgment

A receiver, which does not have any downstream node, broadcasts a *Data Receive Acknowledgment (DRA)* message containing its upstream node's ID within one hop to provide an explicit acknowledgment for the data packets received. Receiving the DRA, the upstream node would be responsible for the data packet delivery at the receiver.

Whenever node A forwards a data packet, it waits for the PsvAck from B. If node A finds that node B is no longer receiving data packets from it but receiving from another node, say D, node A moves the entry for node B from its RSP Table to its MNT Table. After every movement of the entry, node A checks if its RSP table is

empty. If this is the case, A enters Passive Receive Mode, and broadcasts an *Enter PRM* message within one hop to inform its upstream nodes. When A's upstream node, which is responsible for data receipt at A, receives the "Enter PRM", it lengthens the time out duration for receiving PsvAcks from node A for subsequent data packets. The node A, which is in the PRM, will forward data packets from its upstream node with longer interval, say, one data packet every 2 seconds, which is commonly known as heartbeat⁵. In the header of each heartbeat (data packet), the InPRM is set to TRUE. An FG node in PRM sends heartbeats to inform its upstream nodes of its existence. The FG node also looks for PsvAcks or DRAs in response to its heartbeats, which indicate that there are some nodes in the neighbourhood that need to receive data packets from it. If a PsvAck or DRA is received, the FG node leaves the PRM, and then forwards every subsequent data packet, setting the InPRM field in the data header to FALSE. When its upstream node receives the PsvAck with InPRM set to FALSE, the upstream node automatically resets the timeout value for receiving a PsvAck. From that point onwards, it waits for the PsvAck for every subsequent data packet.

3.2.5 Bi-directional Local Link Repair (BLLR)

Most multicast protocols, which possess link repair schemes (e.g. MAODV, Patch-ODMRP, MZR and ADMR reviewed in Chapter 2) repair a broken link from the downstream direction (let us call it *downstream repair*). It may seem difficult to repair a broken link from the upstream direction (let us call it *upstream repair*) as there are multiple destinations (receivers) and it is difficult for the upstream node to find the paths to each one of them. Therefore, the existing schemes take the view that the only way is to let the nodes from the downstream direction initiate the repair,

⁵ The average distance between two communicating neighbouring nodes is half the transmission range, i.e. $250/2 = 125\text{m}$. When nodes move at medium speed, i.e. less than 20 m/sec, the relative speed between them is less than 40 m/sec. Then, the average duration for the link between them to break is about 3 seconds. Thus, the heartbeat interval of 2 seconds is reasonable.

though the disadvantage is that the excessive number of nodes may be included into the multicast topology. Consequently more data packets are forwarded unnecessarily, leading to reduction in the transmission efficiency.

However, we find that the local upstream repair is applicable in the multicast protocols that use the concept of FG. Through PsvAck, an upstream node is able to know its downstream nodes, and if the node loses contact with any one of them, it may be able to repair the link through the other nodes that are in the multicast topology. As a result, by exploiting the existing topology redundancy, fewer nodes are introduced into the multicast topology, and consequently, the transmission efficiency can be improved.

Moreover, downstream repair schemes usually flood control packets locally or globally, which generates high control overhead. Hence, if an effective local upstream repair is applied, the number of initiations of the downstream repair can be reduced and the overall control overhead can be decreased. Of course, the local upstream repair cannot guarantee repairing of every link breakage in a mobile network, and therefore the downstream repair should also be retained.

Based upon the above analysis, we propose the Bi-directional Local Link Repair (BLLR) strategy. It is applicable for multicast protocols that use the concept of forwarding group and use broadcasting for packet delivery. In BLLR, first a local upstream repair is performed; if it fails, then a downstream repair is initiated. In the implementation, provided that the traffic comes from upper layers continually, the loss of certain number of PsvAcks, say N_{LPA} , indicates a link breakage. This triggers the upstream repair. Meanwhile, the downstream node also considers the loss of N_{LPA} packets to be the result of a link breakage. It waits for a given duration for the upstream repair to take effect. If the upstream repair succeeds in time, the downstream node does not have to do anything; otherwise, it initiates the downstream repair.

The BLLR is implemented in GBMP by proposing an upstream repair

mechanism, called Upstream-Node-Initiated Local Repair (UNILR) and a downstream repair mechanism, called Receiver-Initiated Attach Procedure (RIAP).

3.2.5.1 Upstream-Node-Initiated Local Repair (UNILR)

MNT Table: Let us illustrate the use of this table through an example. In Figure 3.4, node D receives a PsvAck from node B and finds node A's ID in the "previous node ID" field, then it stores B's ID into its MNT Table. This means that D is monitoring the data receipt at B.

Node Status: In order to facilitate the cooperation between the UNILR and RIAP, four status of nodes (including the sources which can be receivers or FG nodes for other sources) are defined:

- **FG node only:** a node that only acts as an FG node, and it is not a receiver.
- **Receiver only:** a node that only acts as a receiver, and it is not an FG node.
- **FG node as well as Receiver:** a node that acts as both an FG node and a receiver.
- **Normal node:** a node that is neither an FG node, nor a Receiver, nor a source.

A field *enNodeStatus* specifying the "node status" is added in each of the following messages: the data header, the control message "Enter PRM" and "Data Receive Ack" (see subsection 3.2.4 for the usage of these two control messages).

Figure 3.5 demonstrates how UNILR works. Suppose, at first, node A is responsible for the data receipt at node B, and the node D, being in PRM, is monitoring the data receipt at node B. Through previous PsvAcks, both nodes A and D know that B is "FG node as well as Receiver" and is not in PRM. Assume that B moves in the direction shown by the dashed arrow and thus it cannot receive data

packets from A any more. Having not received $MAX_NO_PSV_ACK_OR_DATA$ ⁶ PsvAcks from B, node A initiates the UNILR procedure. Node A first *disables* (not deletes) the RSP Table entry for B. Secondly, it broadcasts within one hop a *Repair Query (RQ)*, asking for help from other FG nodes.

Figure 3.5. Upstream-Node-Initiated Local Repair (UNILR)

Upon receiving the RQ, the FG nodes of the same multicast group are eligible to respond by sending *Repair Acknowledgment (RAck)* if they know that B is in their neighbourhood. However, in order to suppress excessive control packets, different priorities are assigned to these nodes to respond, according to their knowledge of the *queried node* (node B, in our example): a) whether one of these nodes itself is the queried node; b) whether the queried node is in their neighbourhood; c) whether they are responsible for or are monitoring the data receiving at the queried node. A higher priority means smaller back-off time before sending a RAck. The priorities are assigned as follows from the highest to the lowest:

1. The queried node has the highest priority and does not wait before sending a RAck;
2. A node that is responsible for the data receipt at the queried node has the next lower priority, and is allocated the back-off window of $[1, 31]$;
3. The next lower priority is assigned to a node that is monitoring the data

⁶ The value of $MAX_NO_PSV_ACK_OR_Data$ can be decided dynamically by $(T/Ipkt)$, in which T is the timeout delay for initiating the UNILR and $Ipkt$ is the average data packet inter-arrival time. T is assigned according to upper layer's requirements, and in our simulations, it is 300 milliseconds.

receipt at the queried node and is not in PRM. The back-off window for this priority is [32, 63];

4. A node that is monitoring the data receipt at the queried node and is in PRM has the lowest priority. The allocated back-off window is [64, 127].

If node A receives the RACK sent by B, it realizes that B is still in its neighbourhood. It enables the entry for B in its RSP Table, and will not act upon other RACKs that arrive later. Any node that knows B randomly selects a back-off value from the back-off window (of its current priority), and then waits accordingly (the larger the value, the longer a node must wait before sending its RACK). During this delay, if it overhears a RACK towards A, the node cancels the operation; otherwise, it sends out a RACK. The unit of the back-off window should be large enough, such that a RACK sent by another node can be heard and redundant RACKs can be suppressed to the greatest extent. However, a very long back-off is also undesirable as the data delivery will suffer from the long delay, which possibly results in application failure. Therefore, in our implementation, the unit is set to 1 millisecond (e.g., if the selected back-off value is 31, a node waits for 31 milliseconds before sending).

Next, let us suppose that node D successfully sends out a RACK towards A because it is now monitoring the data receipt at B. Node A receives the RACK from D first, and responds to it by sending a *Repair Notification (RN)*. Then, A deletes the entry for B in its RSP Table. When node D receives the RN, it leaves PRM and from then on forwards every data packet for node B.

It is possible that node A does not receive any RACK in response to its RQ. In this case, it will perform UNILR multiple times (a typical value is 3). After that, if still no RACK is received, node A gives up hope of repairing using UNILR. In this scenario, the RIAP *may* be triggered by a node downstream the broken link as described in the next two subsections.

We note that whenever the downstream node is in the PRM, a single loss of

PsvAck at the upstream node is considered to be quite severe, and thus the UNILR is immediately triggered.

UNILR attempts to make good use of the limited redundancy provided by the loosely structured tree for link repair. If another FG node is responsible for the data receipt at the queried node, UNILR does not need to do anything. However, if the queried node is monitored by another FG node, which is in PRM, UNILR requires the monitoring node to leave the PRM.

3.2.5.2 Receiver-Initiated Attach Procedure (RIAP)

RIAP is triggered by a receiver when it detects the loss of several consecutive data packets during a multicast session (if the multicast session terminates, the upper layer informs the routing layer protocol and the receiver node simply leaves the multicast group). RIAP requires that, while receiving data packets, every receiver stores the data packets' arrival time and their sequence numbers. Using the information, a receiver is able to calculate the packets' average inter-arrival time, and estimate the arrival time of the next data packet. When $PKT_LOSS_THRESHOLD^7$ estimated data packets are lost, the receiver initiates the RIAP by broadcasting an *Attach Request (AReq)* with TTL set to two hops⁸.

All the nodes on the tree are allowed to respond to the AReq provided that they have received data packets recently⁹. However, these nodes have different priorities (higher priority means smaller back-off time before sending an *Attach Acknowledgment* in response to the AReq). The priorities are as follows:

1. The FG nodes have a higher priority than the non-FG nodes;

⁷ The loss of $PKT_LOSS_THRESHOLD$ consecutive packets indicates that the link is probably broken, and thus the repair is initiated. Receivers with different node status have different values for this threshold, which will be discussed in the next subsection.

⁸ In order to reduce the control overhead, the TTL is set to two, and the repair is performed in the local area only.

⁹ Otherwise, a node just decrements the value of TTL by one and broadcasts it until TTL is equal to 0, at which the packet is discarded.

2. An FG node that is not in PRM has a higher priority than the ones in PRM.

Before sending the *Attach Acknowledgment (AAck)*, a node waits for a random delay according to its priority. During the wait, if it does not receive any AAck meant for the initiator of the AReq, the node sends out the AAck; otherwise, it does not send any. The node sending the AAck is called the *Attach Point Node (APN)*. The receiver that initiated the AReq responds to the first AAck it receives by sending an *Attach Notification (ANtf)* towards the APN. The intermediate node on the path towards the APN, if it is not already an FG node, becomes one for this multicast group; otherwise, it leaves PRM if it is in this mode. Then the intermediate node forwards the ANtf to the APN. When the APN receives the ANtf, if it is in PRM, it leaves this mode. Thus, the receiver resumes receiving data packets from a new path.

3.2.5.3 Cooperation between UNILR and RIAP

The bi-directional link repair is achieved by the cooperation between UNILR and RIAP. The following rules are applied for this purpose:

1. When an upstream FG node fails to receive MAX_NO_PSV_ACK_OR_DATA (set to three)¹⁰ consecutive PsvAcks from one of its downstream nodes (which is not in PRM) that it is responsible for, and the downstream node's status is "FG node as well as Receiver" or "FG only", the upstream node initiates the UNILR for this downstream node; otherwise, UNILR is not initiated.
2. When an "FG node as well as Receiver" fails to receive 6 data packets¹¹, it initiates the RIAP.
3. When an upstream FG node fails to receive *one* PsvAck from one of its

¹⁰ The loss of 3 consecutive packets is considered to be the result of a link breakage. So the UNILR is initiated after 3 consecutive passive acknowledgments are lost.

¹¹ The gap of 3 more data packets loss for an "FG node as well as Receiver" to initiate the RIAP is used to separate the initiation of UNILR and RIAP.

downstream node (the downstream node is in PRM) that it is responsible for, the upstream node initiates the UNILR for this downstream node.

4. When a “Receiver only” node fails to receive 3 consecutive data packets, it initiates the RIAP procedure. As the receiver’s upstream node should be able to know the receiver’s status through DRAs, its upstream node will not start the UNILR procedure.

Therefore, for a link breakage, BLLR is achieved in this way. GBMP first initiates the UNILR if the downstream node is not a “Receiver only” one. If the UNILR fails, it is up to the downstream node to decide whether or not to execute RIAP according to its status. If the downstream node is “Receiver only”, it initiates the RIAP without waiting for the upstream node to execute UNILR.

3.2.6 Membership Management

Nodes are free to enter or leave the multicast group at any time. A node may enter the multicast group by broadcasting an Attach Request message or by responding to a GJQ by sending a GJR or LJR. A multicast receiver node may leave the multicast group silently. A multicast source node is also allowed to leave the multicast group at any time. If it wants to do so, it must flood a message *Source Leave* to the whole network. Every node should have a source table storing IDs of all the multicast source nodes in a multicast group. When the multicast group leader leaves, the multicast source, whose ID is the least larger than the former leader, becomes the multicast group leader.

3.3 Performance Evaluation

We evaluate the performance of our proposed protocol GBMP, and compare it with ODMRP and ADMR using GloMoSim [35]. Both ODMRP and ADMR are mature protocols and have their respective IETF drafts. The ODMRP has to flood the Join

Query frequently so as to refresh the multicast topology timely to cope with node's mobility. Also, the lifetime for an FG node should not be too long; otherwise, the transmission efficiency decreases due to too many redundant packets. Thus, we choose 3 seconds for the Join Query interval and 4 seconds for the lifetime of an FG node. For ADMR, a source sends a data packet every 30 seconds using network flooding. For GBMP, GJQ is sent every 60 seconds, and LJQ is sent every 9 seconds. The large durations for multicast topology refreshing are chosen for ADMR and GBMP since these two protocols employ link repair mechanisms. Table 3.1 lists the parameters used in simulations.

In our simulations, all multicast members retain their membership throughout the simulation. The sources start transmitting CBR traffic at the 5th second of the simulation time to avoid transient effect, which may occur due to membership formation at the beginning of the simulation. For simplicity, multicast sources are randomly selected from the 20 group members and these sources are also receivers. All the nodes move at the same speed, which varies from one simulation to another.

Table 3.1. Simulation Environment Parameters of Routing Protocols

Parameter	Value
Number of Nodes	50
Number of Multicast Group	1
Number of Multicast Group Member	20
Area	1000 m × 1000 m
Mobility Model	Random-Way-Point
Nodes' Speed	5 ~ 30 m/s (5 m/s increase per step)
Pause Time	5 seconds
Radio Transmission Range	250 m
Propagation Path-Loss Model	Free Space
Channel Capacity	2 Mbps
Simulation Time	300 seconds
ODMRP Join Query Interval	3 seconds
ODMRP FG node lifetime	4 seconds
ADMR Data Flooding Interval	30 seconds
GBMP Global Join Query Interval	60 seconds
GBMP Local Join Query Interval	9 seconds
MAC Layer Protocol	IEEE 802.11

Two different traffic patterns are defined in Table 3.2.

Table 3.2. Traffic Patterns (TP-3.1 and TP-3.2)

Traffic Pattern	Description
TP-3.1	CBR, constant packet inter-arrival time of 250 milliseconds, 256 bytes per packet
TP-3.2	CBR, constant packet inter-arrival time of 100 milliseconds, 256 bytes per packet

The two traffic patterns do not represent any particular application, but are defined to test the protocols' capability in delivering data packets. As the broadcast scheme of IEEE 802.11 MAC is known to be unreliable, when there are multiple sources, the starting time of the CBR traffic emission is dispersed within a second interval according to the number of sources (e.g., if there are three sources, the traffic starting time at the sources is staggered by 1/3 second) so that the unnecessary collisions are reduced, otherwise the performance would be over pessimistic. Each simulation run was conducted 10 times with different seed numbers. The reported data were averaged over these runs.

3.3.1 Performance Metrics

The following metrics are used to compare the performance of the three protocols:

- **Packet delivery ratio (PDR):** the percentage of data packets that is delivered to (or received by) all the receivers, which is defined as

$$\frac{\sum_{i=1}^{N_R} (\text{the number of data packets actually delivered to receiver } i)}{\sum_{i=1}^{N_R} (\text{the number of data packets should be delivered to receiver } i)} \times 100\% ,$$

where N_R is the number of receivers. PDR demonstrates a protocol's ability of delivering data packets and is directly related to the performance of upper layers.

- **Number of data packets delivered per data packet transmitted (Transmission Efficiency, TE):** indicates the multicast efficiency of the multicast topology, which is defined as

$$\frac{\sum_{i=1}^{N_R} (\text{the number of data packets actually delivered to receiver } i)}{\sum_{j=1}^{N_{NW}} (\text{the number of data packets transmitted by Node } j)},$$

where, N_{NW} is the number of nodes in the network.

- **Control Overhead (CO):** the number of control packets transmitted per data packet delivered to the receivers, which is defined as

$$\frac{\sum_{j=1}^{N_{NW}} (\text{the number of control packets transmitted by Node } j)}{\sum_{i=1}^{N_R} (\text{the number of data packets actually delivered to receiver } i)}.$$

- **Control Byte Overhead (CBO):** the number of control bytes transmitted per data packet delivered to the receivers, which is defined as

$$\frac{\sum_{j=1}^{N_{NW}} (\text{the length in bytes of control packets transmitted by Node } j)}{\sum_{i=1}^{N_R} (\text{the number of data packets actually delivered to receiver } i)}.$$

Note the difference between TE and CO/CBO. TE reflects the effectiveness of forwarding data packet within the multicast topology, while CO/CBO reflects the cost (ineffectiveness) of constructing, maintaining and repairing the topology (i.e., the higher the CO/CBO, the less effective a protocol is in topology construction, maintenance and repair).

- **Normalized Packet Overhead (NPO):** the total number of all the data and control packets transmitted per data packet delivered to the receivers, which is defined as

$$\frac{\sum_{j=1}^{N_{NW}} (\text{data packets transmitted by Node } j) + \sum_{j=1}^{N_{NW}} (\text{control packets transmitted by Node } j)}{\sum_{i=1}^{N_R} (\text{data packets actually delivered to receiver } i)}$$

- **Consecutive Packet Loss:** the distribution of the number of data packets lost consecutively at the receivers. We define the Weighted Occurrence of Consecutive Packet Loss (WOCPL) to be:

$$WOCPL = \frac{\sum_{i=4}^N (i^2 \cdot OCPL_i)}{\sum_{i=1}^N (i \cdot OCPL_i)}$$

where $OCPL_i$ stands for the number of occurrences of consecutive i data packets loss. We let $N = 30$ in our simulation; we do not record loss of more than 30 consecutive packets as that implies that partitioning has occurred, which has nothing to do with the performance of the protocols. The denominator represents the total number of data packets lost due to link breakages, while the numerator is the weighted number of data packets lost (normally, a loss of a small number of data packets is not a serious problem for real-time traffic, so this number is calculated from 4 to N). WOCPL measures the discontinuity in the data packet delivery. The smaller the value of WOCPL for a protocol is, the less the protocol is suffering from consecutive packet loss, and the more suitable it is for real-time traffic. To exemplify WOCPL, let us consider any two protocols, called P1 and P2. Table 3.3 and Table 3.4 show OCPL for protocols P1 and P2 for i number of consecutive data packets lost. Therefore, the WOCPLs of the two protocols are calculated as

$$WOCPL_{P1} = \frac{4^2 \times 15 + 5^2 \times 11 + 6^2 \times 10}{1 \times 5 + 2 \times 8 + 3 \times 10 + 4 \times 15 + 5 \times 11 + 6 \times 10} \approx 3.87$$

$$WOCPL_{P2} = \frac{4^2 \times 20 + 5^2 \times 20 + 6^2 \times 15}{1 \times 3 + 2 \times 5 + 3 \times 5 + 4 \times 20 + 5 \times 20 + 6 \times 15} \approx 4.56$$

Since $WOCPL_{P1} < WOCPL_{P2}$, we observe that, under the same network configuration, the protocol P1 is able to deliver data packets more smoothly than P2 (of course, in order to compare the two protocols, we need to evaluate other metrics also).

Table 3.3. Example of OCPL for Protocol P1

<i>i</i>	1	2	3	4	5	6
<i>OCPL_i</i>	5	8	10	15	11	10

Table 3.4. Example of OCPL for Protocol P2

<i>i</i>	1	2	3	4	5	6
<i>OCPL_i</i>	3	5	5	20	20	15

3.3.2 Simulation Results

3.3.2.1 Impact of Mobility and Traffic Pattern

Figure 3.6 and Figure 3.7 show the performance of the three protocols under two different traffic patterns when there is one source in the network. In most of the cases, GBMP has the highest packet delivery ratio with the lowest control overheads, control byte overhead and normalized packet overhead among the three.

GBMP and ADMR have higher Packet Delivery Ratio (PDR) than ODMRP in most of the mobility scenarios. The reasons are as follows. Firstly, for single source case, the network topology is actually a source-based tree. For GBMP and ADMR, because they employ loosely structured multicast forwarding trees, the number of sources in the network has very little effect on the topology. However, in ODMRP, in the case of a single source, there is not enough redundancy in the topology. Secondly, ODMRP requires the source to flood the Join Query frequently in order to cope with the link breakages and the sub-optimal routes, and it has no mechanism to repair broken links. GBMP and ADMR, nevertheless, have their own mechanisms for link breakage repair, so that broken links can be repaired promptly and therefore the packet loss decreases. Thirdly, ODMRP's frequent sending of broadcast packets (Join Query) increases congestions and collisions (due to IEEE 802.11 MAC layer broadcast mechanism's potential unreliability), which decreases the number of data packets received. On the other hand, with the help of the link repair mechanisms, both GBMP and ADMR need not refresh the multicast topology as often as ODMRP

does, leading to higher packet delivery ratio.

(a) PDR, 1 src, TP-3.1

(b) TE, 1 src, TP-3.1

(c) CO, 1 src, TP-3.1

(d) CBO, 1 src, TP-3.1

(e) NPO, 1 src, TP-3.1

(f) WOCPL, 1 src, TP-3.1

Figure 3.6. Simulation Results of 1 Source, TP-3.1 Scenario

(a) PDR, 1 src, TP-3.2

(b) TE, 1 src, TP-3.2

(c) CO, 1 src, TP-3.2

(d) CBO, 1 src, TP-3.2

(e) NPO, 1 src, TP-3.2

(f) WOCPL, 1 src, TP-3.2

Figure 3.7. Simulation Results of 1 Source, TP-3.2 Scenario

Note that the PDR of ADMR is lower than that of GBMP. This is due to ADMR's pruning mechanism, in which a node N prunes itself from the multicast tree if it has no downstream nodes. With the pruning mechanism, ADMR is able to achieve higher transmission efficiency by forwarding lesser data packets, especially

when the mobility of nodes are low (see Figure 3.6 (b) and Figure 3.7 (b) speeds 5 m/s and 10 m/s). Nevertheless, in a mobile network, it is possible that after some time, a node is required to forward data packets again for its “new” downstream nodes. In ADMR, the only way for node N to re-embark on the multicast forwarding tree is to rely on its downstream nodes’ local or global repair mechanism, which incurs extra control overheads (see Figure 3.6 (c) and Figure 3.7 (c)). In contrast, GBMP uses the PRM at the FG nodes to reduce data packet forwarding while remaining on the tree; also, GBMP sends Heartbeat packets, so that newly arriving downstream nodes can be quickly discovered and node N returns to normal forwarding status earlier. In this manner, although the transmission efficiency of GBMP is lower, packet delivery ratio is improved and control overhead is reduced. PRM thus makes a good trade-off between the transmission efficiency and packet delivery ratio.

In the traffic pattern TP-3.1, the control overheads and normalized packet overheads of GBMP and ADMR are less than those of ODMRP (Figure 3.6 (c) and (e)). The reason is that the flooding intervals of GBMP and ADMR (to optimize the multicast topology) are longer than that of ODMRP. Although the local repairing mechanisms of GBMP and ADMR generate extra overheads, the benefits outweigh the overheads. Nevertheless, in the heavier traffic pattern TP-3.2 (Figure 3.7 (c) and (e)), the NPO of ADMR is higher than that of ODMRP, and the CO of ADMR and GBMP approach (ADMR exceeds) that of ODMRP as nodes’ speed increases. The reasons for the above performance in TP-3.2 are as follows. Under the heavier traffic load, all three protocols suffer from frequent data packet loss, and therefore the link repairing procedures are initiated more frequently in GBMP and ADMR. GBMP employs only local repair mechanism, and its control overheads increase moderately, and therefore, the control overheads of GBMP are less than ODMRP and ADMR. ADMR, however, utilizes a global repair mechanism once the local repair fails, and as a result the control overheads increase fast, especially at high speeds. The increase

in control overheads can also be observed through the metric CBO (see Figure 3.6 (d) and Figure 3.7 (d)).

In both TP-3.1 and TP-3.2, we note that ADMR has higher control overheads than GBMP. This is mainly due to ADMR's global repair mechanism. Let us recall that whenever a link breakage occurs, ADMR lets the node, say node N, which is immediately downstream the broken link, initiate a local repair. The nodes that are on the sub-tree "below" node N start a timer to monitor the result of the local repair. When the timer expires and if there is still no data packet coming from their upstream nodes, the receivers on this sub-tree will initiate their own global repair. When the global repairs are triggered, the control overheads incurred are proportional to the number of receivers on the sub-tree and the number of nodes in the network. In GBMP, however, whenever a link breakage cannot be repaired by UNILR occurs, the affected receivers initiate the RIAP independently. Both the UNILR and the RIAP are restricted in local areas. Therefore, the overheads are only proportional to the number of nodes in the nearby areas (within two hops) and the number of receivers affected by the link breakage. Consequently, the control overheads of GBMP are less than that of ADMR.

The control byte overhead (CBO) performance varies in different traffic patterns. For TP-3.1, GBMP has the least CBO (Figure 3.6 (d)) except at 5 m/sec, ADMR has the least. This is due to its reduced control overheads. However, under TP-3.2, GBMP's CBO approaches ODMRP's (Figure 3.7 (d)). This is because in heavy traffic load and high mobility, the possibility of losing data packets increases due to increased collisions. Hence, GBMP uses more control packets. However, considering that, in practice, not all the nodes move with the same high speed (in contrast, the nodes in our simulations do), therefore normally there will be less control packets, and consequently reducing CBO.

Figure 3.8 and Figure 3.9 demonstrate the performance of the three protocols under the two traffic patterns when there are three sources in the network. Under TP-

3.1, the packet delivery ratio is the highest in ODMRP (Figure 3.8 (a)), while under TP-3.2 it is highest in GBMP (Figure 3.9 (a)). In a relatively lighter traffic load (i.e., TP-3.1), the advantage of the mesh used in ODMRP is evident. With the provision of topology redundancy, ODMRP is able to cope with mobility well. For GBMP and ADMR, the topology redundancy is less than that of ODMRP, which leads to lower packet delivery ratio than ODMRP. On the other hand, ODMRP's high PDR is achieved at the cost of lower transmission efficiency and more overheads. Moreover, under higher traffic load (i.e., TP-3.2), GBMP gives the best PDR performance due to its effective topology and local repair mechanisms. The degradation in PDR for ODMRP under heavy traffic load is because of the increased collisions, which occur due to more packets being forwarded in the mesh.

Note that in Figure 3.9 (a), ADMR has much lower PDR than the GBMP and ODMRP. By observing Figure 3.9 (c) and (e), it is obvious that in order to resume packet forwarding, many control packets have been transmitted. Indeed, this makes things worse, since too many control packets take up much bandwidth and this further reduces the number of data packets delivered. This can also be deduced by observing Figure 3.9 (d) for the actual CBO posted.

(a) PDR, 3 src, TP-3.1

(b) TE, 3 src, TP-3.1

(c) CO, 3 src, TP-3.1

(d) CBO, 3 src, TP-3.1

(e) NPO, 3 src, TP-3.1

(f) WOCPL, 3 src, TP-3.1

Figure 3.8. Simulation Results of 3 Sources, TP-3.1 Scenario

Figure 3.9. Simulation Results of 3 Sources, TP-3.2 Scenario

Note that GBMP’s transmission efficiency is slightly better than that of ADMR (see Figure 3.9 (b)), which proves that the PRM performs well under heavy traffic load.

Under TP-3.1, both GBMP’s and ADMR’s control overheads and normalized

THE GATEWAY-BASED MULTICAST PROTOCOL (GBMP)

packet overheads are much less than ODMRP's. However, under TP-3.2, the control overhead and control byte overhead of GBMP is very close to ODMRP, which should be acceptable, considering that 1) in our simulation runs, nodes' speeds are much higher than those likely to be in practical scenarios, and 2) GBMP yields higher packet delivery ratio, higher transmission efficiency and lower normalized packet overheads.

3.3.2.2 Consecutive Packet Loss

In order to study the discontinuity in delivering data packets, Figure 3.6 (f) and Figure 3.7 (f) demonstrate the Weighted Occurrence of Consecutive Packet Loss (WOCPL) for the single source case under traffic patterns TP-3.1 and TP-3.2, respectively; Figure 3.8 (f) and Figure 3.9 (f) demonstrate the WOCPL for the three-source case (only the WOCPL of one source is shown here).

GBMP has the lowest WOCPL under traffic pattern TP-3.1 for both 1-source and 3-source scenarios. because the reason is that in TP-3.1, the data packets' inter-arrival time is 250 milliseconds, which is long enough for the BLLR to take effect and thus reduce the OCPL.

The performance of ADMR is not as good as GBMP. ADMR has the lowest WOCPL only in the scenario "1 source TP-3.2" (Figure 3.7 (f)) (in this case, however, ADMR's PDR is much lower than GBMP), while the highest in other cases. ADMR's repair mechanism relies on the operation of a single node (say node N) for the local repair; and after the local repair fails, affected receivers initiate their own global repairs. A delay exists between the initiation of local repair and global repair, during which receivers may continue to suffer from packet loss. Furthermore, even though the local repair succeeds, because of the mobility in the network, the sub-tree "below" node N may have other link breakages. Consequently, the timers, which are used by the receivers on this sub-tree to monitor the result of the local repair, may expire before all the broken links are repaired. Therefore, some of the

receivers may still initiate the global repairs independently. So, this process delays the resumption of the data packet delivery, without reducing control overheads significantly.

ODMRP’s WOCPL is less than that of GBMP and ADMR in the scenario of “3 sources, TP-3.2” (Figure 3.9 (f)). In this case, however, ODMRP’s TE is much lower than that of GBMP. Also, ODMRP’s average end-to-end delay is much higher than GBMP (see next section for comparison of the average end-to-end delay). Therefore, although ODMRP does not suffer from high WOCPL in this scenario, the frequent flooding (every 3 seconds) by the sources lead to inefficient use of the limited bandwidth. This problem will deteriorate when the number of sources increases.

3.3.2.3 Average end-to-end delay

One of the major concerns in designing routing protocol in MANETs is that a repair scheme may introduce latency, which will prevent upper layer applications from operating properly. Therefore, we study the average end-to-end delay of the three protocols under different scenarios in Table 3.5 to Table 3.8.

Table 3.5. Average End-to-end delay (sec), 1 source, TP-3.1 under different speeds

	5m/sec	10m/sec	15m/sec	20m/sec	25m/sec	30m/sec
ODMRP	0.010466	0.010664	0.01077	0.010698	0.010537	0.010478
GBMP	0.012528	0.013131	0.013812	0.013798	0.01378	0.013819
ADMR	0.03587	0.036687	0.038603	0.039617	0.038453	0.038741

Table 3.6. Average End-to-end delay (sec), 1 source, TP-3.2 under different speeds

	5m/sec	10m/sec	15m/sec	20m/sec	25m/sec	30m/sec
ODMRP	0.010254	0.01052	0.010645	0.010611	0.010221	0.010267
GBMP	0.01249	0.013065	0.013959	0.013949	0.013786	0.014061
ADMR	0.040092	0.044293	0.048819	0.051036	0.051305	0.052784

Table 3.7. Average End-to-end delay (sec), 3 sources, TP-3.1 under different speeds

	5m/sec	10m/sec	15m/sec	20m/sec	25m/sec	30m/sec
--	--------	---------	---------	---------	---------	---------

ODMRP	0.015541	0.016266	0.016278	0.016706	0.016835	0.017084
GBMP	0.016762	0.016898	0.016852	0.016864	0.016821	0.01732
ADMR	0.044341	0.052759	0.056028	0.061629	0.088928	0.103704

Table 3.8. Average End-to-end delay (sec), 3 sources, TP-3.2 under different speeds

	5m/sec	10m/sec	15m/sec	20m/sec	25m/sec	30m/sec
ODMRP	0.096505	0.094059	0.090464	0.090968	0.100822	0.103325
GBMP	0.031769	0.030344	0.029927	0.030916	0.03739	0.037688
ADMR	0.656745	0.810818	0.908373	0.815258	0.737636	0.836201

For the single source cases (Table 3.5 and Table 3.6), the average end-to-end delays of GBMP are slightly higher than those of ODMRP, but much less than those of ADMR. For the 3-srouce cases, with TP-3.1 (Table 3.7), the performance of GBMP and ODMRP does not deviate considerably from each other; with TP-3.2 (Table 3.8), GBMP’s average end-to-end delays at different speeds are much lower than the others. Indeed, although the execution of BLLR has to rely on link breakage detection (the loss of several consecutive data packets) and by exchanging control packets, the other mechanisms in GBMP help reduce the delay thus incurred. For example, the loosely-structured tree topology facilitates the packet forwarding via shortest paths; the PRM helps improve transmission efficiency such that less congestions and collisions occur. Even the BLLR itself tries to repair broken links in a timely manner (first performs UNILR, and then performs RIAP if necessary). Therefore, the introduction of link repair mechanisms in GBMP is effective and does not incur much increase in latency.

Also note that in Table 3.8, ADMR has the highest end-to-end delay. By referring to Figure 3.9, it is not difficult to understand that ADMR suffers from huge packet losses in this heavy traffic scenario. More importantly, ADMR’s higher CO and CBO (Figure 3.9 (c) and (d)) indicate that a lot of control packets have been generated for the purpose of link repair. They not only take up a large portion of the bandwidth, but bring about congestion as well. As a result, the end-to-end delay

increases tremendously.

3.4 Summary

Performing multicast in MANETs is challenging due to wireless networks' physical constraints as well as nodes' mobility. In this chapter, we have proposed a new multicast protocol for MANETs, called the Gateway-Based Multicast Protocol. By constructing local trees (local groups), the delay between the building of the multicast topology and the sending of the succeeding data packets is reduced. Receivers are grouped into local groups and the links among them are maintained locally to reduce the control overheads. The Passive Receive Mode is utilized to suppress unnecessary data packet forwarding. Upstream-Node-Initiated Local Repair and Receiver-Initiated Attach Procedure are initiated from upstream and downstream directions, respectively, to repair the broken links locally. We define a new performance metric, called Weighted Occurrence of Consecutive Packet Loss (WOCPL), to measure the discontinuity in the data packet delivery, which indicates whether or not a protocol is suitable for real-time traffic. Simulation results prove that GBMP is able to achieve high packet delivery ratio with low overheads and high transmission efficiency in high mobility and high traffic load scenarios. The WOCPL performance shows that GBMP is suitable for real-time traffic.

Please note that the design of GBMP is based on the assumption that the MAC layer is able to provide reliable broadcast service to the routing layer. However, as we will see in the following chapters, this cannot be achieved by simply using IEEE 802.11 MAC DCF. Therefore, we improve the broadcast reliability at the MAC layer in Chapter 5 and incorporate it with GBMP in Chapter 7.

Chapter 4. Literature Review of Reliable MAC Protocols for MANETs

In this chapter, as a preliminary, we first describe the IEEE 802.11 MAC protocol's unicast and broadcast schemes. After analyzing its limitation in supporting reliable broadcast service, we then critically review some reliable broadcast MAC protocols aiming to improve the IEEE 802.11 MAC's broadcast reliability.

4.1 IEEE 802.11 MAC DCF

The IEEE 802.11 standards include two MAC Protocols, the *Distributed Coordination Function (DCF)* and the *Point Coordination Function (PCF)*. With respect to MANETs' infrastructure-less characteristic, all the nodes are operating in a distributed way. Therefore, DCF, which requires no infrastructure, can be applied in MANETs.

DCF allows for automatic medium sharing between compatible physical layers through the use of Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) and a random backoff time following a busy medium condition (known

as Collision Resolution). Furthermore, positive acknowledgement scheme is utilized to schedule retransmission by the sender if the previous transmission fails.

Interframe Space (IFS): IEEE 802.11 defines interframe space, which is the time interval between frames. Four different IFSs¹² are defined to provide priority levels for access to the wireless media: short interframe space (SIFS), PCF interframe space (PIFS), DCF interframe space (DIFS) and extended interframe space (EIFS). SIFS is used before the transmission of CTS, Data or Ack frame. DIFS is applied before RTS transmission or back-off. If an erroneous frame is received, the EIFS is used. The PIFS is used in PCF and is not of our concern.

Carrier Sense: IEEE 802.11 performs both physical and virtual carrier sense. The physical carrier sense is performed in the physical layer, while the virtual carrier sense is in the MAC layer. In order to implement virtual sense, all the frames contain a field named *Duration*, which is the time duration that the medium is reserved for the following frame exchange sequence. The *Duration* field is used by an overhearing node to update its Network Allocation Vector (NAV) on conditions that: 1) the containing frame is correctly received; 2) the value of the *Duration* field is greater than the node's current NAV value. A node should not transmit information on the medium if either the physical carrier sense indicates the medium is busy or its NAV value is above zero.

Collision Avoidance: Due to self-interference (see Section 1.2), collision detection is very difficult in wireless networks. Therefore, IEEE 802.11 DCF defines a four-way handshake to avoid collisions. The four-way handshake, which includes RTS/CTS/Data/Ack frame exchange sequence (see Figure 4.1), is applied if the Data frame length is above a pre-defined threshold *dot11RTSThreshold*. Prior to sending a unicast Data frame, a sender sends *Request To Send (RTS)* to the destination node.

¹² In IEEE 802.11e, which is a supplementary draft providing QoS support, a new IFS, called Arbitration Interframe Space (AIFS) is introduced. The AIFS is at least DIFS, and can be enlarged individually for various Traffic Categories (TCs).

Using the Duration field of RTS, it informs other nodes in its transmission range of the medium reservation. In response to the RTS, the destination node replies with a *Clear To Send (CTS)* frame if the medium is determined to be idle. Nodes outside the transmission range of the source but inside the range of the destination will update their NAV according to the overheard CTS. If a CTS is received, the source node sends out the Data frame and awaits the Ack from the destination; otherwise, if either the CTS or the Ack is lost, the source node will reinitiate a four-way handshake for the same Data frame. One should note that the use of RTS/CTS is able to reduce the probability of collision of the Data frame. However, it cannot eliminate the hidden terminal problem completely as the transmission of the RTS frames from the sender may collide with that from the hidden nodes at the destination node. Therefore, the RTS/CTS handshake cannot always succeed.

The RTS/CTS and the Ack mechanisms *cannot* be used for broadcast or multicast, since there are multiple destinations for the RTS and Data, which incur potentially multiple concurrent transmissions of CTSs and Acks. As a result, in IEEE 802.11 DCF, broadcast/multicast Data frames are sent using the CSMA/CA only, which is unreliable.

Figure 4.1. Four-way handshake

Contention Resolution: IEEE 802.11 adopts the Binary Exponential Back-off (BEB) algorithm for contention resolution. Each node maintains a Contention Window (CW) and a back-off counter. After the medium is sensed to be idle for DIFS or EIFS time, a node generates a random backoff period according to Equation

(4.1) for an additional deferral time before transmitting, unless the backoff counter already contains a non-zero value, in which case it is not needed to generate a new random number.

$$\text{Backoff Time} = \text{Rand}() \times \sigma \quad (4.1)$$

where σ is the duration of an empty slot time, and the function $\text{Rand}()$ returns a random number that is drawn from a uniform distribution over the interval $[0, CW]$. The CW parameter takes an initial value of CWmin, and takes the next value in the series every time a retransmission is initiated, until the value of CWmax is reached. The CW is reset to CWmin after every successful attempt to transmit a MAC service data unit (MSDU) or a MAC management Protocol data unit (MMPDU). That is, for the i th retransmission, let us suppose when i reaches m' , CW reaches CWmax; however, when i further reaches the retransmission limit m , the pending frame is dropped and CW is reset to CWmin. Therefore, the value of CW is given by:

$$CW = \begin{cases} (CW_{\min} + 1) \times 2^i - 1 & 0 \leq i < m' \\ (CW_{\min} + 1) \times 2^{m'} - 1 & m' \leq i \leq m \end{cases} \quad (4.2)$$

If a node wants to compete for the media, it must first defer for DIFS period after the media is idle. Then, it generates a backoff counter and backs off before initiating a transmission. During the deferring and backoff, if the media is sensed to be busy, the node freezes its backoff counter. The count down of the backoff counter will be resumed after the media is sensed to be idle once again and remains idle for DIFS time, unless the backoff counter is already reduced to zero, in which case, no backoff is needed and a transmission can be initiated right away. However, if a collision occurs, before reinitiate a transmission, a new backoff counter value has to be generated.

In the following sections, we critically examine the reliable broadcast MAC protocols proposed in the literature.

4.2 Robust Broadcast

The principle of Robust Broadcast [36] is to require another node (the collision detector) to detect collision and to feedback this information to the Data sender. A RTS/CTS/Data handshake is performed for broadcasting.

When a node wants to send a broadcast packet, as a first step, it must finish the collision avoidance phase. Then, it tries to get the address of the collision detector (the method of getting the collision detector’s address will be described later). If the collision detector is available, a RTS is addressed to it; otherwise, the whole scheme is disabled and the Data frame is broadcasted directly. Corresponding to the RTS, if a CTS is received from the collision detector, the node broadcasts the pending Data frame (see Figure 4.2); otherwise, it performs a backoff procedure and retries later as it would do for a unicast frame.

Figure 4.2. Reliable Broadcast

There are two methods proposed to choose the collision detector. For networks with a base station, the scheme can use the base station as a collision detector. For ad hoc networks, the source node (other than the current sending node) of the latest message sent over the medium can be a collision detector candidate, since that node is guaranteed to exist and is active.

The problem with this mechanism is that the RTS/CTS exchange can only avoid the collisions in the neighboring area of the collision detector but not at other neighbors. As is shown in Figure 4.3, the node S sends a RTS addressed to the

collision detector, node D. Let us assume that node D successfully replies with a CTS and node S receives it. Therefore, the collision at nodes A and E, resulting from node C's concurrent transmission with node S, can be avoided. However, since only one node is chosen to detect collision and transmit back the CTS, collision from node G at node B cannot be prevented. Hence, the avoidance of hidden terminal problem in the sender's neighborhood is quite limited.

Figure 4.3. Different Scenario of Collision Detection

4.3 Broadcast Support Multiple Access (BSMA)

Broadcast Support Multiple Access (BSMA) [37] assumes that the radio has DS (Direct Sequence) capture ability¹³. It incorporates the RTS/CTS/NAck control frames exchange. After successfully finishing the collision avoidance, the sender transmits a RTS to all of its neighbors (addressed to the broadcast address) and waits for WAIT_FOR_CTS time units for CTS from its neighbors. If not being in YIELD state (in which either the virtual or physical carrier sense prevents a node from

¹³ A radio with capture ability has the capability to lock onto a sufficiently strong signal in the presence of other interfering, less powerful signals. If the ratio of the arriving frame's signal strength over the sum of all colliding packets is larger than the threshold value, the frame is deemed to be received successfully while other colliding frames are dropped.

transmitting), the neighbors are supposed to send their respective CTSs to eliminate the hidden terminal condition. After transmitting the CTS, the neighbors wait for WAIT_FOR_DATA time units for the following Data frame. If the sender “captures” a CTS from any of its neighbors, it broadcasts the pending Data frame and waits for a NACK within WAIT_FOR_NACK time unit; otherwise, if it fails to receive any CTS, the sender must backoff and access the medium again to send a RTS. At the neighbor side, if no Data frame is received after WAIT_FOR_DATA time units, a NACK is sent addressed to the RTS sender. When the sender receives (or captures) a NACK from any of its neighbors, it should re-access the medium following the same procedure described above and retransmit this lost Data frame; otherwise, a broadcast process is considered to be completed and the sender may access the medium for transmitting the next broadcast frame following the same procedure described above.

This scheme relies upon the physical layer’s capture ability to receive the CTS and NACK. However, as analyzed in [41], a successful capture can only occur when the strongest frame’s Signal-to Interference Ratio (SIR) is 10db or more [10]. This requires that, if there are two nodes sending CTS or NACK simultaneously, the ratio of the two nodes’ distance from the RTS or Data sender must be at least 1.5. Obviously, when more than two nodes are sending CTS or NACK, the equivalent 10db SIR and distance ratio of 1.5 are difficult to achieve. Moreover, as discussed in [42], when nodes (or traffic) are uniformly distributed, the capture effect occurs with a probability of about 0.55 when there are two colliding frames and the required SIR is 10db. This probability further drops to 0.3 with the existence of 5 competing nodes, and the limit of it is 0.2 if the number of the contending frames further increases. Hence, in BSMA, relying solely on the capture effect may lead a sender to 1) miss the colliding CTS frames and attempt to access the medium again for transmitting the RTS, which lowers the protocol’s efficiency; 2) miss the colliding NACK frames and consequently, miss the necessary retransmissions, which lowers

the reliability of the broadcast service.

A similar scheme reported in [38] utilizes a similar RTS/CTS mechanism along with collision avoidance. However, it does not require the radio to have direct sequence capture ability. After performing collision avoidance and transmitting a RTS to all its neighbors for a pending Data frame, during the period that the sender expects to receive the CTS, if the medium is sensed to be busy, it is deemed to be the result of simultaneous (possibly colliding) CTSs from the neighbors. A Data frame is transmitted once the medium becomes idle. However, the busy medium may not necessarily be caused by collided CTSs, especially in a network with heavy traffic and hidden terminal problem. In this case, a Data frame is sent before the medium is reserved successfully.

4.4 Broadcast Medium Window (BMW)

Broadcast Medium Window (BMW) [40] treats broadcasting as multiple reliable transmissions to the neighbors in a round robin fashion. Upon completing the collision avoidance phase, a source node sends a RTS to one of its neighbors. The format of RTS is modified to further contain 1) the sequence number of the earliest transmitted frames stored in the source (SN_E), and 2) the sequence number of the latest frame waiting to be sent (SN_C). These information implies that the frames whose sequence number lies in the range (SN_E, SN_C) are all stored in the source node. The corresponding CTS from the addressed neighbor contains either the sequence number of a previously missed frame or the latest one. The other neighbors should adjust their NAV according to the time duration specified in the RTS or CTS heard, such that the following Data/Ack exchange can be performed successfully. After receiving the CTS, the source transmits the Data frame, whose sequence number is specified in the CTS, and anticipates an Ack. Upon finishing the transmission of one frame to this neighbor, the source node will continue the RTS/CTS/Data/Ack

exchange (skipping the collision avoidance phase) with it until the latest Data frame is transferred. After that, the source will initiate a dialogue with another neighbor. During the process, the other neighboring nodes receiving the Data frames also update their records of received frames' sequence numbers, which may reduce unnecessary future transmissions.

The main problem of BMW is the *imprecise NAV problem*. It occurs if the packets have different lengths. When the source node sends the RTS addressed to one of its neighbor, it has no idea of which frame the neighbor needs, and therefore, is unable to decide the value assigned to the Duration field in the RTS. A safe solution is to set the Duration according to the longest frames stored in the source. At the neighbor's side, if it has missed a previous frame, then of course, it has no idea of the missed frame's length. It can only assign the value of the Duration field in the CTS according to what is specified in the RTS. The other neighbors adjust their NAV according to the RTS and CTS heard. Since the Duration field is calculated based on the longest stored frame, if the length of the actual transmitted frame is less than the longest one stored in the source, bandwidth is wasted (see Figure 4.4).

Figure 4.4. Imprecise NAV Problem

Moreover, if the involved source and its neighbor update their NAVs according to the actual shorter Data frame, they will start the backoff procedure earlier than other neighboring nodes, which makes them easier to win the competition for the medium. This condition further generates the problem of unfair access to the

medium, since the source may not give in the control of the medium until all the neighbors have been served. However, setting the Duration field to any value less than the maximum length of the stored frames may result in the Data frame to collide with other frames, leading to more retransmissions and lower efficiency.

4.5 Broadcast Scheme with Broadcast Acknowledgement

(BACK)

As proposed in [43], a node broadcasts a Data frame according to the standard IEEE 802.11 MAC DCF. However, in order to ensure that the sender is aware of whether a Data frame has been received by all the neighbors or not, all the neighbors are required to respond right away in the DIFS following the Data frame. That is, after the Data frame, the time duration starting at the end of SIFS and terminating at the end of DIFS is defined to be Backoff Acknowledgement (BACK) window (see Figure 4.5).

Figure 4.5. Back Window

The BACK window is divided into minislots and each neighbor should select one of them to transmit a BACK. Two x-bit patterns $p(x)$ are defined to identify the BACK message:

$$p(x) = \begin{cases} 0, & \text{ACK for a newly received broadcast packet} \\ 1, & \text{ACK for a duplicatedly received broadcast packet} \end{cases}$$

These two patterns are designed to inform the sender of the newly or

duplicatedly received Data frame. If a neighboring node receives a broadcast frame successfully, it should randomly choose a BACK window minislot and transmit the corresponding pattern bits according to whether the frame is a new or a duplicated one. The number of bits that is used in the patterns is determined by the link quality: if the channel quality is poor, a longer pattern length should be applied and less minislots will be allocated. Hence, the number of minislots in a BACK window is determined by

$$\frac{DIFS - SIFS}{x/W}, \quad \text{where } W \text{ is the network's}$$

bandwidth

Due to its design's limitation, the BACK messages are simply for notification to the sender of the broadcasted frame. In the case that the number of received BACK is less than the number of neighbors, a rebroadcast is performed. Moreover, a maximal broadcast retry threshold, MBRT, is defined to limit the bandwidth wastage when the sender fails to collect sufficient number of BACKs even after many rebroadcasts.

The main problem of this scheme is that as long as there are not enough BACKs collected (due to the collision of two BACKs, or some neighbor's moving away from the sender's transmission range), a rebroadcast is performed. Especially in the case that there are more neighbors than the number of minislots, and all the neighbors send their own BACKs to acknowledge the Data frame received, collision of two or more BACKs is certain to happen. Therefore, sender will have to perform rebroadcast unnecessarily until MBRT threshold is reached, which is a severe wastage of the bandwidth.

4.6 Batch Mode Multicast MAC (BMMM) Protocol

The main idea of BMMM [41] protocol is, rather than using at least n rounds of DCF-like unicasts for a multicast/broadcast frame intended for n neighbors, by

consolidating the n contention phases into one, the required time to serve a multicast/broadcast can be significantly reduced.

As illustrated in Figure 4.6, the sender of the Data frame utilizes RTSs to sequentially instruct each intended receivers (its neighbors) to transmit their CTSs. Once the multiple RTS/CTS exchanges have been finished and the sender receives at least one CTS, it broadcasts the pending Data frame. Then, the Ack transmissions from those neighbors who have sent their CTSs are coordinated. BMMM introduces a new control frame, called Request for Ack (RAK) to accomplish the Ack coordination. The neighboring node that is addressed by a RAK should send its Ack to the source. The neighbors, whose Acks are not received by the RAK sender, will be included in sender's next round of medium access for this Data frame's rebroadcast.

Figure 4.6. BMMM Protocol

Although BMMM decreases the expected number of contention phases, which tries to minimize the time required to serve a Data frame's broadcast reliably, the cost of introducing extra control overhead is high. In the case that the source fails to receive some CTSs or Acks (due to collision or some neighbors are restrained from responding as in a YIELD state), a next round of medium access and costly RTS/CTS/Data/RAK/Ack frame exchanges will be initiated.

4.7 Multiple Access Collision Avoidance Protocol for Multicast Service (MACAM)

MACAM [44] resort to RTS/CTS/Data/Ack frame exchange to provide reliable broadcast service. When the pending Data frame is longer than a predefined threshold length, the sender broadcasts a RTS to its neighbors. Figure 4.7 shows the format of the revised RTS frame.

Figure 4.7. MACAM RTS Frame Format

A list of intended recipients (specified by the RA fields) is included in the RTS. Receiving the RTS, if its address is indicated by one of the Receiver Addresses (RA), a neighbor should send a CTS (see Figure 4.8). The neighbors' CTS transmission sequence is equivalent to the order of RAs specified in the RTS. After at least one CTS is received, the sender transmits the pending Data frame. However, if the number of received CTSs is less than that of the neighbors, the sender will contend for the medium to rebroadcast the Data frame to nodes whose CTSs are missing.

Figure 4.8. MACAM Operation

Nevertheless, in order to limit the length of a RTS frame, a pre-defined

threshold is used for the maximum number of RAs that can be included. Therefore, if the number of neighbors exceeds this threshold, only the threshold number of neighbors are included in the RTS. The remaining nodes have to receive the Data frame through retransmission by the same sender.

The main problem with this protocol is that in order to hold a list of RAs, the length of the RTS frame is increased considerably. Thus, the loss and consequent retransmission of the RTS become costly. Moreover, although the number of RAs included in a RTS is limited, the neighbors excluded from the first RTS may receive the Data frame already in the first transmission. However, the sender has no idea of this condition. Consequently, for the next round of RTS transmission, the previously excluded nodes will be included in a new RTS, which increases overhead considerably. Thirdly, the retransmission of the RTS is based on whether the number of received CTSs equals the number of RAs in the prior RTS. If a node is restricted from sending due to virtual carrier sensing, it cannot transmit the CTS. Hence, even though this node can receive the Data, a retransmission is still performed, which may be unnecessary.

4.8 Reliable Point-to-Multi-Point (RPMP)

RPMP [45] proposes to send Multicast-Request-to-Send (MRS) prior to the Data. MRS addresses all the neighbors who are involved in a point-to-multi-point transmission session. The neighbors in turn answer with identical Multicast-Clear-to-Send (MCS) at the same time. For the reception process of the MCSs, it is proposed to use the orthogonal frequency division multiplexing (OFDM)¹⁴. With OFDM the reception of more than one simultaneously transmitted MCS packet becomes

¹⁴ OFDM is based on frequency division multiplexing (FDM), but is much more spectrally efficient by spacing the sub-channels much closer together (until they are actually overlapping). This is done by finding frequencies that are orthogonal, allowing the spectrum of each sub-channel to overlap each other without interference.

possible.

Figure 4.9. RPMP

RPMP assumes that by using MCSs, it is possible to reserve the media around the neighbors successfully. However, sometimes it is not so, since, if the MRS collides with other frames, a neighbor will not be able to reply with an MCS, and consequently, the hidden terminal problem cannot be avoided.

4.9 Summary

In this chapter, we have reviewed the existing works on providing reliable broadcast at the MAC layer. The main consideration that leads IEEE 802.11 DCF to employ only CSMA/CA, rather than RTS/CTS/Data/Ack mechanism, for broadcasting is that there are multiple destinations for the RTS and Data, and thus potentially multiple concurrent senders of the CTSs and Acks in response. Multiple CTSs collide at the sender and the sender will have no idea of whether or not all the neighbors have reserved their neighboring medium successfully. Same consideration applies to Ack frame transmission.

Therefore, in existing works, the sender either (a) addresses only one neighboring node for medium reservation (Reliable Broadcast); or (b) address neighbors one by one for multiple transmissions of the Data frame (BMW); or (c) address the neighboring nodes sequentially or together preceding one transmission of the Data frame (BMMM, MACAM and RPMP) using different techniques; or (d)

use different Ack or NAck schemes (BSMA and BACK). Nonetheless, all of them have their drawbacks concerning reliability and/or efficiency.

Hence, it can be concluded that for MAC layer reliable broadcast protocol, Acks for Data frames should be used as it is natural and robust. In addition, considering the hidden terminal problem, using RTS/CTS to reserve medium so as to reduce retransmission is also needed for long frames. In this case, the overhead of using control frames must be taken into account such that the protocol will not suffer from low efficiency. Furthermore, in broadcast and multicast, the problem of CTS or Ack storm, which is caused by their simultaneous transmission by potentially multiple senders, must be dealt with carefully.

Chapter 5. Providing Reliable Broadcast Service at the MAC Layer

Broadcasting is one of the essential communication models of MANETs. Many MANET multicast routing protocols rely heavily upon MAC layer's broadcast service for data delivery, multicast topology construction and maintenance. However, the broadcast mechanism of the IEEE 802.11 standards cannot provide reliable broadcasting support, and therefore, satisfactory performance of the multicast protocols cannot be guaranteed. In this chapter, we first propose an extension to IEEE 802.11 broadcast mechanism, called Round-Robin Acknowledge and Retransmit (RRAR), to address this shortcoming. RRAR has a novel round robin acknowledge scheme for the neighboring nodes to report lost frames, based on which the Data frame sender calculates and records the lost frames' indices (or sequence numbers), and performs retransmissions. Based on RRAR, we further propose Adaptive Round-robin Acknowledge and Retransmit (ARAR). ARAR adapts to the traffic load by adjusting the length of a Data frame to reduce the control overhead. Accordingly, different handshake sequences are adopted to broadcast short and long frames.

The RRAR and ARAR perform retransmissions in each hop. One may argue that rather than performing the hop-by-hop retransmission (HHR), the end-to-end

retransmission (EER, i.e. the individual links on a path do not provide link-layer retransmissions; reliable transfer is achieved by requiring the end receivers to report lost frames and then the sources perform retransmissions) can also be used. Indeed, as analyzed and compared in [38], under identical scenario, the energy consumption by EER is at least an order of magnitude larger than HHR even for moderate values of link error rate. This, although is not our main focus in this thesis, justifies our work of providing hop-by-hop link layer reliability in multi-hop, ad-hoc wireless networks.

5.1 Data Structures

In order to keep a record of the Data frames that are lost or received, each node maintains a 32-bit *Received Frame Bitmap (RFB)* for every neighboring sender. With reference to the latest received Data frame (of sequence number m) from a sender, if an earlier frame is lost, the corresponding bit in the RFB is set to '0'; otherwise, it is '1' (see Figure 5.1). Of course, the bitmap is deleted after having not received any Data from a sender for a long period of time.

Figure 5.1. Received Frame Bitmap (RFB)

For the purpose of retransmission, every Data frame is stored in a queue at the sender, called the *Broadcast Data Queue (BcDQ)*, after its first-time broadcasting. The earliest Data frame is deleted when a new one is to be queued and the BcDQ is full. With each stored frame, a counter, called the retransmission counter, is associated. This counter counts the number of times the frame has been retransmitted. If the counter's value exceeds *MAX_RETX_LIMIT*, the corresponding frame is

simply deleted from the BcDQ. The length of the BcDQ is set to 64, which is twice the number of frames that an RFB can represent. Also, a flag, called *ReTxFlag*, is associated with each BcDQ entry specifying whether or not the current Data frame needs retransmission.

A *Neighbor List (NL)*, which contains the addresses of neighbors, is maintained in each node. If a “new” node is overheard to be broadcasting a frame, it is added into the NL. If a neighbor has not been heard from for a long time, it is deleted from the NL along with the RFB for it.

5.2 Round-Robin Acknowledge and Retransmit (RRAR)

5.2.1 Overview of RRAR

We assume that there are always multicast packets available for transmission. Also, the paths to all nodes have similar propagation characteristics.

In order to enhance the reliability of the broadcast communication, RRAR introduces an acknowledgment and retransmit scheme. Nevertheless, RRAR avoids the well-known problem of *Ack explosion* and collision by requiring the Ack from only one of the neighboring nodes receiving the broadcast message. Thus, after finishing the collision avoidance phase, the sender requires, in the broadcast Data frame header, one of its neighbors to send back an acknowledgment, called *Broadcast Acknowledgment (BrAck)*. The BrAck contains the sequence number of the latest received frame, s , and a bitmap specifying the previous frames that are received or lost with reference to s . The selection of the node for a BrAck is performed in a *round-robin* style among all the neighbors that have been heard on the medium. Then, by checking the bitmap in the received BrAcks, the Data sender calculates and records those lost frames and retransmits them if they are still stored.

5.2.2 Protocol Description

When the MAC layer of a node, say A, has to broadcast a packet coming from its upper layer, RRAR first performs CSMA/CA to avoid collisions in the neighborhood. If the collision avoidance phase is passed, node A broadcasts the Data frame directly. We modify the header of the Data frame by inserting a field BrAck Node Address. This field contains the address of the node that is requested to send a BrAck (see Figure 5.2).

Figure 5.2 Modified Data Frame Header

The selection of the BrAck node is based on the sender’s knowledge of its neighbors and is done in a round-robin style. For each new packet to be broadcast, the sender selects a different neighbor for replying with a BrAck. In the case that the traffic has just begun and no frame has been overheard on the medium, the sender does not know of its neighbors’ existence and the NL is empty. In this case, as well as when the current frame is a retransmitted one, the BrAck Node Address is set to *all 1s*. In the following parts of this thesis, we assume that all nodes always have a non-empty NL.

Suppose node A selects node B as the BrAck node and broadcasts the Data frame *m* (as is depicted in Figure 5.3). Frame *m* is then stored in A’s BcDQ. Suppose all the neighbors, B, C and D, receive this frame. By checking this received frame’s sequence number, all of them update their respective RFB; by checking the BrAck Node Address in the Data frame header, node B generates a BrAck and sends it to node A after SIFS time; node C and D update their NAV according to the *Duration/ID* field in the frame header so as to remain silent during the BrAck

transmission by B.

Figure 5.3 Round-Robin Acknowledgement

The format of the BrAck frame is illustrated in Figure 5.4. The *Latest Sequence Number* field contains the sequence number of the latest received Data frame from the sender whose address is contained in the “Destination Address”; and the “Received Frame Bitmap” contains the RFB with reference to the “Latest Sequence Number”.

Figure 5.4. BrAck Frame Format

If node A receives the BrAck successfully, by checking the Latest Sequence and the Received Frame Bitmap fields, the sequence numbers of the Data frames lost at node B can be calculated. If these frames are still stored in the BcDQ, they will be retransmitted. Note that the retransmitted frames are simply discarded by those nodes who have already received them.

Similarly, when the data packet $m+1$ arrives from the upper layer, node A selects another neighbor, say node C, for sending BrAck, as is shown in Figure 5.3.

It should be noted that no BrAck is required for retransmitted Data frames. This

is due to the following reasons. If one neighbor lost a frame, it is possible that some other neighbors also lost it. Therefore, the frame's retransmission is likely to benefit several neighbors. After the retransmission, if there are neighbors who still fail to receive it, they may well report it through later BrAcks. Moreover, one of the nodes that reported the lost frame may move away before this frame is retransmitted. So we do not require nodes to send BrAcks for retransmitted frames.

It is sometimes possible that after broadcasting the Data frame, no BrAck is received. This can be due to the collision either at the recipient of the Data or at the recipient of the BrAck (i.e. the sender of the Data), or due to the fact that the neighbor moves away from the sender. However, this does not pose any problem since, by checking the bitmaps in the BrAcks of later Data frames from other neighbors, the sender is able to decide whether some previous frames need retransmission or not. Therefore, when sending the next Data frame, even though the BrAck of the previous one is not received, the sender will choose another neighbor from the NL, provided that there are multiple neighbors in its NL.

Another special case is that when the traffic load is not heavy, the interval between the broadcasting of two successive Data frames might be long. In this situation, if there is no frame that needs transmission, a sender may transmit a *BrAckQuery* frame. This control frame asks one of the neighbors (also in a round-robin style) to send a BrAck and checks if there is any previously transmitted frame that needs retransmission. If it is the case, a retransmission is performed immediately. Otherwise, a BrAckQuery is sent addressing another neighbor until all the neighbors have been queried once in one round.

5.2.3 Performance Evaluation

We evaluate the performance using GloMoSim. For convenience, Table 5.1 lists the parameters used in our simulation if not mentioned otherwise.

Table 5.1. Simulation Environment Parameters of RRAR

Parameter	Value
Number of Nodes	50
Area	1000 m × 1000 m
Mobility Model	Random-Way-Point
Nodes' Speed	20 m/s
Pause Time	5 s
Radio Transmission Range	250 m
Sensing Range	500 m
Propagation Path-Loss Model	Free Space
Channel Capacity	2 Mbps
Simulation Time	300 s
CBR Data Packet Size	64 octets
ODMRP Join Query Interval	3 s
ODMRP FG node lifetime	4 s
Number of Sources	3
Broadcast Data Queue Length	64 frames
MAX RETX LIMIT	3

We have also implemented the MACAM (see Section 4.7) protocol for comparison with RRAR. In MACAM, by using RTS/CTS frame exchange prior to the Data transmission, the hidden terminal problem can be alleviated. However, due to the fact that a long RTS will introduce excessive control overhead, in our simulations, the maximum number of Receiver Addresses (RAs) included in a RTS is set to three (we vary the number of RAs from one to three, specified by the data series *MACAM-1RA*, *MACAM-2RA* and *MACAM-3RA*, such that the impact of this parameter can be observed easily). For fairness in comparison, the maximum retransmission limit for a Data frame is set to three, which is the same as in RRAR.

In the routing layer, ODMRP is used due to the reason that ODMRP relies solely on MAC layer's broadcast service and has no link repair mechanism. Therefore, an enhancement in the MAC layer broadcast reliability leads directly to the network layer's performance improvement.

One may argue that with multiple paths provided by the mesh in ODMRP, if a node loses one packet from a neighbor, it may well receive the packet from another neighbor, and consequently, reliable broadcast support from the MAC layer may be unnecessary. However, as the number of packets forwarded in the network increases,

collisions are likely to increase, and as a result the data delivery suffers. Therefore, if we improve the reliability of the MAC layer broadcast, the multicast protocol's performance can be improved significantly.

In the simulations, all the three sources start to send CBR traffic at the same time (rather than dispersing the starting time of various traffic flows as in section 3.3). The intervals between two successive CBR data packets sent by the three sources are always the same. We use this scenario in order to test the performance when there are many collisions. Moreover, mobility (nodes are moving at 20m/s) is applied to all the nodes in order not to lose generality for MANETs. Each simulation run was conducted 10 times with different seed numbers. The reported data were averaged over these runs.

5.2.3.1 Reliability

To measure the reliability, we use the metric, Packet Delivery Ratio (PDR). Figure 5.5 (a) shows the packet delivery ratio of a 20-member group under different traffic loads.

In all the cases, RRAR provides much higher PDR to ODMRP than IEEE 802.11 and MACAM. Approximately 52%~ 56% data packets are delivered to the receivers by IEEE 802.11, because the simple CSMA/CA mechanism always encounters severe collision problem under our simulation scenario.

MACAM improves PDR over IEEE 802.11. Different number of RAs results in different outcomes. Under heavy traffic (e.g., when the packet inter-arrival duration at a source is 50 milliseconds), the fewer RAs included in a RTS, the higher the reliability MACAM provides. As shown in Figure 5.5 (a), the PDR performance of MACAM-2RA approaches that of MACAM-1RA as traffic load becomes lighter. The reason is as follows. It is obvious that the more RAs included in a RTS, the more difficult for the RTS sender to collect all the required number of CTSs from the neighbors. Therefore, not receiving enough CTSs, the Data sender will retransmit

(the sender transmits the Data when at least one CTS is received; and it performs retransmission for the rest of the neighbors whose CTSs were not received). This is based on the assumption that if a CTS from a node is not received, that node is probably not able to receive the following Data. However, this assumption does not hold if a) the non-responding neighbor may have been restricted from sending due to carrier sense; or b) the CTS collides with another frame at the RTS sender. In both cases, the neighbor may have received the Data correctly, and retransmissions of RTS and CTS are unnecessary. Then the PDR performance is degraded as the unnecessary RTSs and CTSs take up a large portion of the limited bandwidth and generate severe collision and congestion. Therefore, MACAM-1RA makes a tradeoff between reliability and number of retransmissions under heavy traffic. However, when the traffic intensity decreases, MACAM-2RA's performance improves since more RTS retransmissions are scheduled due to failure in collecting enough CTSs. Nevertheless, the fact that MACAM-2RA provides a higher PDR than MACAM-3RA also shows that long RTS is not preferred as the retransmission of RTS is costly.

In all the cases, RRAR performs the best. RRAR can provide a PDR close to 90%, even though it encounters the same problem of severe collisions. With the bitmap included in the BrAcks, a sender is able to identify the frames that are lost and perform retransmissions for Data frames. This directly increases the possibility that frames are delivered to receivers, and consequently, PDR is improved. Also, RRAR is able to know lost frames and perform necessary retransmission in an efficient way. Retransmission is performed only when required through BrAcks in RRAR, rather than counting on the number of received CTSs (as in MACAM). Hence, RRAR utilizes the limited bandwidth more efficiently and provides a higher reliability (through just enough number of retransmissions) than MACAM.

Figure 5.5. Reliability Improvement of RRAR over IEEE 802.11 MAC and MACAM

We notice that when the data interval is 50 milliseconds, the PDR of RRAR is much lower than higher data interval cases. This is mainly due to the fact that in this traffic pattern, the medium is always very busy. In the IEEE 802.11, the available capacity is allocated solely to Data frames; while in RRAR, BrAcks are transmitted. These control frames together with those first-time-transmitted Data frames take up a larger portion of the bandwidth and reduces the bandwidth for retransmissions. Hence, the PDR is lower when the network is congested.

Figure 5.5 (b) presents the packet delivery ratio when the multicast group size varies and the data interval remains fixed at 100 milliseconds. RRAR outperforms IEEE 802.11 and MACAM in all the cases. With the group size increasing, the performance of IEEE 802.11 improves slightly. It is easy to imagine that when the number of receivers increases, more nodes become FG nodes. Therefore, the mesh provides more optional routes from the sources to the receivers, and therefore increases the PDR. Nevertheless, for RRAR, from group size of 20 to 40, the PDR decreases slightly. This is because, firstly, with increasing group size, more control packets (for multicast topology construction and maintenance) are transmitted. Secondly, with the increase in the FG nodes, a packet is transmitted more number of times. These two factors together make the traffic load in the network heavier. As a

result, the performance degrades slightly. In spite of this, RRAR delivers more frames than IEEE 802.11.

5.2.3.2 Throughput

Figure 5.6 depicts the average throughput each multicast receiver gets when different traffic load and different group size is concerned. As more data packets are delivered to the receivers by introducing RRAR, it is obvious that each receiver enjoys higher throughput compared to the IEEE 802.11 broadcast scheme and MACAM.

Figure 5.6. Throughput Improvement of RRAR over IEEE 802.11 MAC

5.2.3.3 Average end-to-end delay

Table 5.2 shows the performance of the average end-to-end delay under different traffic intensities. IEEE 802.11 always has the lowest value due to two reasons. First, there is no retransmissions of Data frames. Hence, the average value of end-to-end delay is often small. Second, due to IEEE 802.11 MAC broadcast's unreliability, the data packets that travel to the far away receivers are less than those in RRAR and MACAM, i.e., more data packets are received by the receivers that are nearer to the sources. Therefore, the computed end-to-end average delay is always small.

RRAR's average end-to-end delay is much lower than MACAM when traffic load is heavy (i.e., data inter-arrival duration is 50 milliseconds and 100 milliseconds). When the traffic load decreases, MACAM-1RA's end-to-end delay

outperforms RRAR's. This is because when the traffic load is light, collisions happen less often, and MACAM-IRA can collect enough number of CTSs (1 CTS for MACAM-IRA) more easily. Also, no retransmissions of Data frames are performed even if they are lost (this can be proved by MACAM-IRA's lower PDR than RRAR). Therefore, the end-to-end average delay is low, which is computed for the received packets. RRAR, however, performs retransmission for the lost Data frames as long as it is able to do so. When retransmissions are performed, the average end-to-end delay would surely increase.

Table 5.2. Average end-to-end delay (sec) under different traffic intensity

	50msec	100msec	150msec	200msec	250msec
802.11	0.027041	0.01805	0.017748	0.018758	0.018055
RRAR	3.579784	0.481167	0.29711	0.321792	0.416898
MACAM-IRA	9.322607	2.84434	0.259481	0.082842	0.066083
MACAM-2RA	20.2821	16.68648	11.93135	7.08658	3.96719
MACAM-3RA	27.37652	26.83296	22.67402	15.8168	13.60986

We notice that when the traffic load decreases, the delay in RRAR increases. This is because the lost frames are reported only when: a) a node receives a newly transmitted Data frame and is required to send back a BrAck; or b) a BrAckQuery is received. As the traffic intensity decreases, the interval between the broadcasting of two successive Data frames or the time duration before a BrAckQuery arrives is likely to be longer, resulting in longer latency before the lost frames are reported and their retransmissions are effected.

Table 5.3 illustrates the average end-to-end delay of the protocols with different number of group members. With the increase of the number of group members, the traffic load in the network increases even if the data inter-arrival time at the source does not change. This is because more packets have to be forwarded by increased intermediate nodes (FG nodes) to the receivers. Studying PDR performance (Figure 5.5 (b)) with Table 5.3 shows that the RRAR is able to achieve higher reliability with lower delay than that in MACAM.

Table 5.3. Average end-to-end delay (sec) under different traffic intensity

	10	15	20	25	30	35	40
802.11	0.01612	0.01699	0.01805	0.01891	0.02035	0.02176	0.02240
RRAR	0.27686	0.28157	0.48117	1.39029	2.25225	3.66804	4.17185
MACAM-1RA	0.68219	1.79172	2.84434	3.61672	4.09542	5.10235	5.97573
MACAM-2RA	13.2275	15.0766	16.6865	18.7909	19.5003	20.7292	20.7273
MACAM-3RA	22.0621	25.1815	26.8330	27.5862	29.4595	30.5367	30.9326

5.2.3.4 Normalized Byte Overhead (NBO)

In order to evaluate the overhead of the protocols, we define the NBO as follows:

$$NBO = \frac{\sum_j^{N_{NW}} (\text{Number of bytes transmitted by node } j)}{\sum_i^{N_R} (\text{Number of data bytes actually delivered to receiver } i)}$$

where the numerator denotes the overhead of all *control and Data frames* (seen from the MAC layer) in the network, and the denominator is the summation of all *data packets* received (in bytes, seen from the routing layer) by all the receivers. Therefore, NBO reflects the cost (in terms of bytes transmitted) for each byte of data packet received. Figure 5.7 shows the NBO of the three protocols.

Figure 5.7. Normalized Byte Overhead

At the first glance, the NBO of MACAM-1RA is much lower than RRAR. Nevertheless, if we also consider the PDR (the most important metric) performance, it is easy to understand that MACAM-1RA's low overhead is at the cost of its low

reliability. For RRAR, it is mainly the retransmissions of lost Data frames that increase the NBO.

5.3 Adaptive Round-robin Acknowledge and Retransmit (ARAR)

5.3.1 Overview of ARAR

In addition to the round-robin acknowledgement and retransmission employed in RRAR, ARAR also tries to pack multiple upper layer packets into one frame. This is an effective means to reduce control overhead and improve throughput when the traffic load is getting heavier. If the packing generates a long frame, before it is transmitted, RTS/CTS are exchanged between the Data sender and its neighbors, which is explained in detail later. This is to avoid the hidden terminal problem and reduce the possibility of costly retransmissions.

5.3.2 Protocol Description

When there is at least one packet from the upper layer for the MAC layer, ARAR builds the MAC Data frame. It tries to pack multiple broadcast packets into the frame body (see Figure 5.8). Please note that the MAC layer does not wait for multiple packets from the upper layer; only the packets that exist in the queue between the MAC and the upper layer (we assume that there is a queue implemented in-between) are considered for packing. In order to limit the length of the packed frame so as not to broadcast a very long frame through the media, two threshold values are used:

- *MAX_PKT_PER_FRAME_LIMIT*: the maximum number of upper layer packets that can be packed into one frame (this is an upper threshold to ensure that not

too many packets from one node is packed, such that the fairness of transmission is retained), and

- *MAX_PACK_FB_LENGTH_LIMIT*: the maximum length of a frame body that contains multiple packets (in accordance to frame body length defined in the IEEE 802.11 standards, its value is set to 2312 octets)

Figure 5.8. Multiple Packets Packed in One Frame

On the other hand, if the length of the resulting frame body is below *MIN_FB_LENGTH_FOR_RTS* bytes (the minimum length of the frame body above which RTS/CTS exchange is required; similar to the *RTSThreshold* for the unicast as described in the IEEE 802.11 standards), the Data frame is considered as a short frame; otherwise, a long frame. Note that if a single upper layer broadcast packet is longer than *MAX_PACK_FB_LENGTH_LIMIT* bytes, no packing is necessary; it is put directly into the Data frame body and deemed as a long frame. Note that in the implementation, it is rather straightforward to differentiate packed frames. According to the IEEE 802.11 standards, the value of the *Type* field in the *Frame Control* field of the frame header is always set to “10” for Data frames. The *Subtype* field can then be used to distinguish whether the frame body contains multiple upper layer packets or not. If the frame body contains only one upper layer packet, the Subtype field is set to “0000”, as is described in the standards; however, if the frame

contains multiple packets we use “1111” to specify it¹⁵.

The process of broadcasting a short frame is different from that of a long frame as described below.

5.3.2.1 Short Frames

The process of broadcasting a short frame includes the basic collision avoidance phase and a Data/BrAck exchange sequence, which is the same as that in RRAR.

5.3.2.2 Long Frames

The CSMA/CA and RTS/CTS/Data/BrAck exchange are applied for broadcasting a long frame.

After CSMA/CA is performed successfully, ARAR transmits a RTS to the broadcast address. Nodes that receive the RTS correctly and that are in idle state should transmit the CTS back to the initiator of the RTS after waiting for SIFS delay.

The RTS/CTS exchange is to reserve the medium for the subsequent transmission of a long Data frame. The problem is, normally, there are multiple neighbors receiving the RTS and they are able to transmit their own CTS back simultaneously. These CTSs collide at the initiator of the RTS, say node A. The work in [38] assumes that the medium busy status is caused by multiple CTSs when the CTS transmission period starts. However, the sensing of the busy status cannot guarantee that it is the result of CTS collision.

Then how to ensure that the frames colliding at the RTS initiator are indeed CTSs? We solve this problem by using the concept of *Enhanced Carrier Sensing (ECS)* [46]. We assume that the transmission rate is fixed for all frames; various

¹⁵ The Subtype value from “1000” to “1111” is reserved in IEEE 802.11 MAC protocol standards.

control frames (RTS, CTS, BrAck) and Data frames have different lengths¹⁶; and the physical layer is able to sense the start and end of a busy period of the medium accurately, such that frames' respective transmission time or even the collision period in the medium can be observed unambiguously. In ECS, with the help from the *Clear Channel Assessment (CCA)* mechanism [12] of the physical layer, the duration of the medium's busy period can be recorded. The length of the collided frame can consequently be calculated and compared with that of a CTS. If they are equal, it means that the collision is caused by simultaneous CTSs from the neighbors and that the medium is reserved successfully. In this case, node A transmits the long Data frame after SIFS (note here that node A does not have to wait for a delay of EIFS before transmitting the Data although CTS collision has been observed), and waits for the BrAck from one of its neighbors.

Nonetheless, if the calculated length is greater than that of a CTS (CTS is the shortest frame among all), node A concludes that some other node is sending a frame other than the CTS, and this frame collides with the CTSs. The medium reservation is, hence, assumed to have failed. Also, receiving nothing during the CTS receiving period is considered to be a sign of medium reservation failure. In both the above cases, node A should re-initiate the medium reservation process. This procedure is repeated until the *retry limit* is reached. If till then, the long frame has not been transmitted yet, the frame is directly queued in the BcDQ without transmission. The node starts to send new packets or perform retransmission after that.

5.3.2.3 Received Data frame processing

Upon receiving a Data frame, if the value of the Subtype field in the frame header is equal to "1111", a node knows that the frame body contains multiple packets. After

¹⁶ In IEEE 802.11 MAC protocol, the ACK frame has the same length as the CTS. In ECS, CTS is modified to be longer so as to make it different from the ACK frame. Since the ACK frame is not used in broadcast, in our work, we retain the original length of the CTS.

removing the MAC header, the ARAR is able to look into the headers of the upper layer packets. By knowing the packets' lengths, packets are delimited from each other. They are unpacked and forwarded to the upper layer one after another.

5.3.2.4 Retransmissions

If temporarily there is no packet in the queue waiting for the first-time transmission, the sender checks whether there are Data frames that need retransmission (by checking the value of the ReTxFlag for each entry in the BcDQ). If it is the case, the sender performs retransmissions of the Data frames in the BcDQ from the earliest to the latest. If a Data frame is a short frame, the CSMA/CA/Data sequence is performed (the same as in RRAR); if it is a long frame, the CSMA/CA/RTS/CTS/Data sequence is performed. Similar to the retransmission of short frames, no BrAcks are required for long frames' retransmissions.

5.3.3 Performance Evaluation

We evaluate the performance using GloMoSim. In the following performance comparison, we will first compare ARAR with the basic IEEE 802.11 MAC broadcast scheme. After that, a comparison between RRAR and ARAR will be performed.

5.3.3.1 Comparison between ARAR and IEEE 802.11 MAC

Table 5.4 lists the parameters used in our simulation if not mentioned otherwise; Table 5.5 lists two network scenarios; and Table 5.6 illustrates two traffic patterns used for evaluation.

Table 5.4. Simulation Environment Parameters of ARAR

Parameters	Value
Mobility model	Random-Way Point
Nodes' speed	20 m/s
Pause time	5 s
Radio transmission range	250 m
Sensing range	500 m
Propagation path-loss model	Free space
Channel capacity	2 Mbps
Simulation time	600 s
Data packet size	64 octets
ODMRP Join Query interval	3 s
ODMRP FG node lifetime	4 s
Number of sources	3
Broadcast Data queue length	64 frames
MAX_RETX_LIMIT	3
MAX_PKT_PER_FRAME_LIMIT	32
MAX_Pack_FB LENG_LIMIT	2312 octets
MIN_FB_LENGTH_FOR_RTS	256 octets

Table 5.5. Network Scenarios (NwSc-5.1 and NwSc-5.2)

Scenario	Description
NwSc-5.1	An area of 1000 m × 1000 m holds 50 nodes, in which 20 are group members.
NwSc-5.2	An area of 2500 m × 2500 m holds 200 nodes, in which 80 are group members.

Table 5.6. Traffic Patterns (TP-5.1 and TP-5.2)

Traffic Pattern	Description
TP-5.1	CBR, constant packet inter-arrival time ranges from 50 ms to 250 ms, with 50 ms step. Packet length is fixed at 64 octets.
TP-5.2	Packets' arrival obeys the Poisson arrival process, whose average inter-arrival time ranges from 50 ms to 250 ms, with 50 ms step. Packet length is fixed at 64 octets.

Figure 5.9 (a) and (b) show the PDR of NwSc-1 with different traffic patterns. The combination of NwSc-1 and TP-5.1 is a scenario with many collisions, where the three sources start to send data packets with the same interval at almost the same time in a relatively small area. We can see that, in Figure 5.9 (a), ARAR provides

much higher PDR to ODMRP than 802.11 does. Approximately 50% data packets are delivered to the receivers by 802.11. In comparison, ARAR can provide a PDR close to 90% in most of the cases. The PDR obtained by using IEEE 802.11 does not vary a lot with the traffic load. It is because the CSMA/CA mechanism always encounters severe collision problem under this simulation scenario. No matter what the traffic load is, the collisions often result in large percentage of data packets colliding and consequently getting lost. ARAR, however, tries to pack multiple packets into one frame, so as to reduce the discarding of packets in the packet queue. Of course, this results in longer frames. For long frames, RTS/CTS exchanges are applied to provide better protection against collisions. This mechanism is able to reduce retransmission for long frames as much as possible. Furthermore, ARAR is able to perform retransmissions for frames that are lost. With the bitmap included in the BrAcks, a sender is able to identify the frames that are lost and need retransmission.

As a comparison, Figure 5.9 (b) illustrates the performance of the scenario in which not so many collisions are there. As the packet inter-arrival time is exponentially distributed, with CSMA/CA only, IEEE 802.11 is able to perform the job fairly well (close to 90%). The performance gain of ARAR compared to 802.11 in this “small network” is not that dramatic, although the PDR of ARAR is approaching 100%. However, if we observe Figure 5.9 (a) and (b) together, it is obvious that the PDR performance of ARAR under different traffic patterns is much more stable than that of IEEE 802.11, which is a favorable feature in practice.

The PDR of NwSc-5.2 with different traffic patterns is presented in Figure 5.9 (c) and (d). It is obvious that in a large network, ARAR still out-performs IEEE 802.11 significantly in most of the cases. Moreover, similar to the performance under NwSc-5.1, ARAR is relatively more stable than IEEE 802.11 under different traffic patterns.

Nevertheless, we also notice that when data packet (average) interval is 50

milliseconds, ARAR delivers slightly fewer frames than IEEE 802.11 (7% degradation in Figure 5.9 (c) and 5% in Figure 5.9(d)). This is mainly due to the fact that the medium is always very busy under this heavy traffic load. In 802.11, the available capacity is allocated solely to Data frames, while in ARAR, RTS/CTS and BrAck control frames are utilized. These control frames take up a part of the bandwidth, and consequently, the bandwidth allocated to Data is reduced.

Figure 5.9. Reliability Improvement of ARAR over IEEE 802.11 MAC

Figure 5.10 (a) and (b) depict the average throughput each multicast receiver gets under NwSc-5.1. Figure 5.10 (c) and (d) show that under NwSc-5.2, ARAR's packing of upper layer packets into one frame helps achieve higher throughput (in the sense that the MAC layer control overhead for each packet is reduced). Also, as more data packets are delivered to the receivers by introducing ARAR, it is obvious that the receivers enjoy higher throughput compared to the IEEE 802.11.

Figure 5.10. Throughput Improvement of ARAR over IEEE 802.11 MAC

Table 5.7 to Table 5.10 illustrate the average end-to-end delays of ARAR for various scenarios. When the packet inter-arrival time is 50 milliseconds, the average end-to-end delay of ARAR is very high; when the traffic load reduces, the average end-to-end delay also decreases due to less congestion and collisions. However, the performance under heavy traffic load is not acceptable for any interactive, real-time or multimedia communications. We will consider alleviating this problem in Chapter 7.

Table 5.7. Average end-to-end delay (sec), NwSc-1, TP-5.1

	50msec	100msec	150msec	200msec	250msec
802.11	0.022765	0.017967	0.017785	0.018571	0.017931
ARAR	2.662937	0.205772	0.10957	0.359562	0.39684

Table 5.8. Average end-to-end delay (sec), NwSc-1, TP-5.2

	50msec	100msec	150msec	200msec	250msec
802.11	0.038905	0.016767	0.014152	0.01305	0.012665
ARAR	3.01686	0.151273	0.080467	0.070936	0.059009

Table 5.9. Average end-to-end delay (sec), NwSc-2, TP-5.1

	50msec	100msec	150msec	200msec	250msec
802.11	0.153999	0.055865	0.041508	0.047054	0.04541
ARAR	5.406312	1.574007	0.557385	0.275589	0.220379

Table 5.10. Average end-to-end delay (sec), NwSc-2, TP-5.2

	50msec	100msec	150msec	200msec	250msec
802.11	0.151909	0.053863	0.046171	0.041358	0.039965
ARAR	5.853094	2.334039	0.615629	0.338578	0.261337

5.3.3.2 Comparison between ARAR and RRAR

From the discussion in section 5.2.3 and 5.3.3.1, it is obvious that both RRAR and ARAR outperform IEEE 802.11 in terms of the broadcast service's reliability and throughput achieved. Under light traffic loads¹⁷, however, the performance difference between RRAR and ARAR will not be very obvious in that both work fairly well. Consequently, in order to have a deeper insight, in this section, we will compare the performance of RRAR and ARAR under constant and moderate¹⁸ traffic intensity. Two new traffic patterns, TP-5.3 and TP-5.4, are defined in Table 5.11. TP-5.3 will be used with NwSc-5.1 for comparing ARAR and RRAR. TP-5.4 will be used with NwSc-5.2.

The traffic load defined by TP-5.3 is heavier than (doubled) that defined in TP-5.4. The reason that we use different traffic load for different network scenario is as follows. If we refer to Figure 5.9 (c) and (d), it is easy to observe that, when the packet inter-arrival time is 50 milliseconds, the packet length is 64 octets and the NwSc-5.2 is used, ARAR's PDR performance is below that of IEEE 802.11. If we continue to use this traffic pattern to compare ARAR and RRAR, the PDR of both

¹⁷ The "light traffic load" here means that the packet inter-arrival time is long, however, the packets are short. For example, when the packet inter-arrival time is 500 milliseconds, and the corresponding packet length is 64 octets.

¹⁸ We will address the issue of performing multicast under heavy traffic in Chapter 7.

the schemes will be no higher than that of IEEE 802.11. Therefore, we define the TP-5.4 to be moderate so as to easily differentiate the performance between ARAR and RRAR. The performance of IEEE 802.11 under ODMRP is also shown as a reference. In addition, we will use the same simulation environment and the two network scenarios defined in the previous section.

Table 5.11. Traffic Patterns (TP-5.3 and TP-5.4)

Traffic Pattern	Description
TP-5.3	CBR, constant packet inter-arrival time ranges from 50 ms to 250 ms, with 50 ms step. Traffic intensity is fixed. The packet inter-arrival time and corresponding packet lengths are listed in Table 5.12.
TP-5.4	CBR, constant packet inter-arrival time ranges from 50 ms to 250 ms, with 50 ms step. Traffic intensity is fixed. The packet inter-arrival time and corresponding packet lengths are listed in Table 5.13.

Table 5.12. Packet Inter-Arrival Time and Corresponding Packet Length for TP-5.3

Packet Inter-Arrival Time (msec)	Packet Length (Octets)
50	64
100	128
150	192
200	256
250	320

Table 5.13. Packet Inter-Arrival Time and Corresponding Packet Length for TP-5.4

Packet Inter-Arrival Time (msec)	Packet Length (Octets)
50	32
100	64
150	96
200	128
250	160

We will use the metrics “Packet Delivery Ratio” and “Average End-to-end Delay” to compare ARAR and RRAR’s performance.

Figure 5.11 (a) shows the PDR performance of ARAR and RRAR using NwSc-5.1 and TP-5.3. In this scenario, collisions are happening frequently. When the packet inter-arrival time is short (50 and 100 milliseconds) and the packet length is also short, the large performance improvement of ARAR over RRAR is obvious. In these two cases, the packing of multiple packets into one frame makes the Data frames of ARAR moderately longer than those of RRAR. This results in two consequences. First, RTS and CTS control frames are then used by ARAR to reserve the medium, which is able to counteract the hidden terminal problem effectively. Second, with longer frames traveling through the medium, it is easier for neighboring nodes to sense the busyness of the medium and defer accessing the medium accordingly. These two consequences lead to reduced collision of frames. Together with the MAC layer retransmission for lost frames, ARAR is then able to achieve higher PDR than RRAR.

Figure 5.11. Performance Comparison of ARAR and RRAR

With the increase in the packet inter-arrival time and corresponding packet length, the performance difference illustrated in Figure 5.11 (a) between RRAR and ARAR diminishes. This is mainly because the longer packet length defined by the traffic pattern results in reduced collisions and, hence, reduces the need for retransmissions. Therefore, both schemes are able to perform well.

Referring to Figure 5.11 (b), which shows the PDR performance under NwSc-

5.2 TP-5.4, similar observations can be made.

Table 5.14 and Table 5.15 illustrate the performance of average end-to-end delay. For short packet inter-arrival times (50 milliseconds and 100 milliseconds), the decrease in average end-to-end delay in ARAR in comparison to RRAR is noticeable. In RRAR, every time a packet is transmitted, a collision is more likely to occur. Thus, the higher collisions, which lead to more Data frame retransmissions, make the delay in RRAR higher than that in ARAR (ARAR provides better protection for long frames against collisions). On the other hand, when the packet inter-arrival time increases (and the correspondingly packet length increases), the difference in end-to-end delay between ARAR and RRAR decreases. This is because ARAR cannot pack as many packets into one frame as it does when the packet inter-arrival time is short. Therefore, the number of upper layer packets in a frame for ARAR and for RRAR does not vary much. Considering that ARAR also has the overhead of RTS and CTS (provided that the length of a frame after packing is above RTSThreshold), it is not surprising that when the packet inter-arrival increases to 250 millisecond, the delay of ARAR is even slightly higher than RRAR. However, in general, ARAR's performance in terms of the average end-to-end delay is better than RRAR.

Table 5.14. Average end-to-end delay (sec) NwSc-5.1 TP-5.3

	50msec	100msec	150msec	200msec	250msec
802.11	0.027041	0.021705	0.025532	0.02781	0.030898
ARAR	1.299791	1.328253	0.95149	0.722149	0.545759
RRAR	3.579784	2.268575	0.99357	0.564539	0.515424

Table 5.15. Average end-to-end delay (sec) NwSc-5.2 TP-5.4

	50msec	100msec	150msec	200msec	250msec
802.11	0.114856	0.050864	0.054094	0.053795	0.055782
ARAR	2.578994	2.374446	1.751406	1.755601	1.266003
RRAR	4.716151	3.750669	2.691502	1.830662	1.036549

Hence, we can conclude that ARAR improves over RRAR by achieving higher

performance in terms of packet delivery ratio and average end-to-end delay.

5.4 Summary

The MAC layer's reliable broadcast service is important not only for the distribution of control packets, but also for the data delivery in many multicast routing protocols for MANETs. In this chapter, we first proposed a simple and effective extension to the IEEE 802.11 broadcast scheme, called Round-Robin Acknowledge and Retransmit (RRAR), to improve the reliability of broadcasting. RRAR provides novel features such as the round-robin acknowledgement scheme and the MAC layer retransmission. By comparing RRAR with the basic IEEE 802.11 CSMA/CA broadcasting and MACAM under the On-Demand Multicast Routing Protocol (ODMRP), the performance of ODMRP has been shown to improve significantly with the introduction of RRAR.

Noting that there is tremendous overhead for every Data frame transmitted, we then proposed ARAR, which improves RRAR by packing multiple packets into one frame and, consequently, adjusting the length of a frame transmitted. For short frames generated, a simple and effective Data/BrAck (Broadcast Acknowledgment) scenario, as applied in RRAR, is applied; for long frames, contrary to the common belief that potential multiple concurrent control frames (CTS) collide at the RTS sender, we use the RTS/CTS/Data/BrAck four-way handshake to reserve the medium. ARAR provides an even more reliable broadcast service than RRAR to the routing layer, together with the huge decrease in average end-to-end delay when packet inter-arrival times and packet lengths are short.

Although the reliability improvement achieved by ARAR is significant, its performance in end-to-end delay is still not satisfactory. This situation may become even worse when the traffic load continues to increase, which may prevent many applications from operating correctly. Therefore, from the next chapter on, we will

address the problem of performance degradation of ARAR under heavy traffic.

Chapter 6. Literature Review of Congestion Control Schemes in Wireless Networks

In this chapter, we critically review various congestion control schemes for wireless networks.

6.1 Alleviating broadcast storm

A common and straightforward approach to perform broadcasting is *flooding*. However, blind flooding causes redundancy of packets, as well as congestion and collision, which are serious problems in wireless networks. This is termed as the *broadcast storm problem*. In this section, we briefly review some broadcast routing protocols, which reduce excessive rebroadcast and consequently alleviate the congestion.

In [47], *adaptive counter-based*, *adaptive location-based* and *neighbor coverage* schemes are proposed to diminish unnecessary rebroadcast. In adaptive counter-based scheme, if during the initial backoff at the routing layer (i.e., before a packet is passed to the MAC layer, a delay is introduced at the routing layer), the node hears the same packet on the medium $C(n)$ times, the pending packet

transmission will be canceled. $C(n)$ is dependent on the node's current number of neighbors, n .

By assuming that the nodes are equipped with positioning devices, the adaptive location-based scheme allows a node to compute the *additional coverage* (ac) provided by its (possible) rebroadcast upon hearing a broadcast packet during the backoff at the routing layer. If ac is less than $A(n)$, this packet is discarded. Similar to $C(n)$, $A(n)$ also depends on the present number of neighbors.

The neighbor coverage scheme does not depend upon the location information provided by the positioning devices, but on more accurate neighborhood information. By utilizing the *neighbor discovering procedure*, a node gets to know all its one-hop and two-hop neighbors. With this information, during the backoff, by recording the senders' IDs of the redundant packets, a node is able to find whether there are any one-hop neighbors that may not have received the packet. If it is the case, rebroadcast is performed after the backoff timer expires.

[48] proposes three directional broadcast schemes based upon the use of *directional antennas*: *On/Off*, *Relay-Node-Based* and *Location-Based* directional broadcast. In *On/Off* directional broadcast, if a broadcast packet first reaches a certain *angle-of-arrival* (AOA) and is received by one of the four directional antennas¹⁹, say $AOA1$, the packet will only be forwarded by other three directional antennas by setting $AOA1$ to *off* (*passive*). Then, during the backoff prior to rebroadcast the packet, if the same packet is received by another antenna, say $AOA2$, $AOA2$ will also be set to off. Therefore, when the backoff timer expires, the packet will only be forwarded through the antennas that remain switched *on* (*active*).

The *Relay-Node-Based* direction broadcast lets a node designate one *relay node* in each direction (in the message header) only. A relay node in a direction is the one-

¹⁹ In this work, it is assumed that each node is equipped with four identical directional antennas, each of which covers an AOA of 90 degree.

hop neighbor that is the farthest from the designating node²⁰ in that direction. A relay node backs off before forwarding. During the backoff, if the same packet arrives from another direction, the directional antenna of that direction will be set to off. Therefore, after the backoff, the relay node only needs to transmit through the antennas left switched on (active).

In Location-Based directional broadcast, it is assumed that nodes are equipped with positioning devices. Also, the backoff for forwarding a packet in different directions varies, which is based upon the *extra coverage* that can be made if the packet is forwarded along a direction. The larger the extra coverage is, the smaller the backoff will be. If the node receives the same packet from a certain direction during backoff, it does not forward the packet in that direction after backoff.

[49] proposes *Relay Set (RS)*, *Neighbor Coverage (NC)* and *Transmission Order (TO)* algorithms, all of which assume that the nodes are equipped with positioning devices. In RS, when a node receives a broadcast packet for the first time, a subset of one-hop neighbors are computed to be relay nodes, whose radio coverage will cover the rest one-hop neighbors. The set of relay nodes is appended to the forwarded packet. In NC, after receiving a packet for the first time, a node backs off before forwarding it. During the backoff, if multiple copies of the same packet are received and if all the other neighbors are covered, rebroadcasting of the packet will be canceled. The idea of TO is that the further a receiving node is away from the sender of a broadcast packet, the shorter backoff period it should wait for before rebroadcasting, such that nearer nodes may have more chance to detect redundant packets and cancel excessive forwarding.

[50] improves the counter-based and distance-based schemes proposed in [51] by combining the two schemes. A received packet is only rebroadcasted if (a) the distance between the sender (the previous hop node) and the receiver (the node that

²⁰ The distance is calculated by measuring previously received signal strength.

is going to rebroadcast) is above a given threshold D_{TH} ; and (b) the number of times that the same packet is received redundantly is below a given threshold C_{TH} .

Most of the schemes reviewed in this section introduce backoff before a packet is passed to the MAC layer, such that packet's rebroadcast may be canceled if it has already been transmitted enough number of times. However, the backoffs considerably increase the end-to-end delay for the long paths. Moreover, the neighbor coverage scheme [47] uses the neighbor discovery procedure, in which nodes broadcast IDs of their neighbors. This generates a high overhead in dense networks. In Relay-Node-Based direction broadcast [48] and Relay Set [49], the set of relay nodes are included in the forwarded packet, the loss of which will reduce the reliability considerably.

6.2 Reliable ODMRP with Congestion Control

[52] extends the On-Demand Multicast Routing Protocol (ODMRP) to facilitate congestion control in nodes that use Broadcast Medium Window (BMW) MAC protocol (described in Section 4.4). Firstly, each Join Query packet header contains the largest *mean aggregate queue length* (which is defined as the mean length of the queue shared by all sources) of the path thus far traversed. Secondly, although duplicate Join Queries are discarded, the route table entry of the next hop node leading to the source is updated to the upstream node that has transmitted a duplicate Join Query, in which a shorter mean aggregate queue length than the current one is included. Thirdly, Join Query packets are only forwarded (broadcasted) by nodes whose mean aggregate queue length is less than or equal to *QUEUE_LENGTH_THRESHOLD*. By doing so, routes, which have long mean aggregate queue length and thus may cause bottlenecks, are avoided. Upon receiving a Join Query, a member will send a Join Reply to the source, which includes the mean aggregate queue length that it currently experiences. Nodes that are FG nodes,

upon receiving Join Reply, check the mean aggregate queue length in the packet. If it is less than the node's current one, it is replaced with the node's queue length. Once the source node receives the Join Reply, the source adjusts its sending rate based on the longest mean aggregate queue length received to avoid congestion along the paths.

Although duplicate Join Queries with shorter mean aggregate queue lengths are used to update routing table, this actually does not help in finding less congested paths. Suppose a node N has forwarded a Join Query with a shorter queue length than another node M, and its downstream node D1 replaces the next hop address to N (in place of M) for the route back to a multicast source. It is very possible that some other nodes do the same as D1. Therefore, when Join Replies come, all of the downstream nodes will forward them via node N, which becomes a hot-spot for incoming data packets. Furthermore, the source adjusts its sending rate based on the longest mean aggregate queue length on one path, which slows down the traffic on other paths unnecessarily.

6.3 Congestion Bandwidth Algorithm (CBW)

CBW [53] introduces service rate, which works in conjunction with source control²¹ and link algorithm²². It seeks to reduce network congestion by adjusting the service rates of the various output queues in nodes. By increasing the service rate, the congestion decreases and waiting time of the packets reduces.

The service rate of an output queue can be adjusted by tuning transmitter power. For a given bit error rate, an increase in the transmitter power can translate into larger constellation size, yielding a larger number of bits transmitted per symbol time,

²¹ Source control algorithms adjust the source transmission rate in response to a measure of congestion.

²² Link algorithms measure the congestion, signal it to the sources, and sometimes actively manage the queue size as well as other system resources.

and therefore, increases the link data rate and reduces the link congestion.

However, links in wireless networks interact in a complex nonlinear manner. The signal on one link may interfere with another. Increasing the power on one link to alleviate local congestion will probably create congestion elsewhere by increasing interference, and thereby reducing the data rate on other links. Hence, CBW seeks to increase rates on links that have a larger congestion-bandwidth product such that the *overall* system congestion can be efficiently reduced.

CBW is a centralized algorithm requiring knowledge of all the links to compute the optimal link rates. In the distributed version of the algorithm, each node periodically transmits the link information only to the node that is the most interfering one. The interfering node uses the information to locally compute and adjust its transmission power such that the global congestion can be minimized.

On one hand, it is very hard to obtain all the links' information in order to compute the optimal link rates. On the other hand, the CBW's distributed version cannot work very well in a highly dynamic network. This is because the movement of the nodes changes the link condition frequently. Only with information from the complaining nodes, a node cannot perform accurate calculation of its transmission power. As a result, the overall performance may degrade.

6.4 COngestion Detection and Avoidance (CODA)

Designed for sensor networks, CODA [54] comprises three congestion control mechanisms: *receiver-based congestion detection*, *open-loop hop-by-hop backpressure*, and *closed-loop multi-source regulation*.

CODA uses a combination of the present and past channel load conditions, together with the current buffer occupancy to accurately infer congestion status at each node. When a node has packets to forward, it starts channel sensing. During an *epoch period* (which is the minimum time period for performing a set of channel

load measurements), instead of continually listening during the backoff time, a node performs periodic sampling of the channel load. The transmitter can be switched off in-between two consecutive samplings so as to save the energy. The channel load is measured for N consecutive sensing epochs, with a predefined sampling rate to obtain channel state information (busy or idle). If the measured load is above a certain threshold, it infers the congestion is there.

Then, the node detecting the congestion initiates open-loop hop-by-hop backpressure, which is by broadcasting backpressure messages upstream toward the source. Receiving the backpressure signals, nodes may slow down their sending rates or drop packets based on the local congestion alleviation policy. Moreover, based upon their own local media conditions, they also decide whether or not further propagation of the backpressure signals should be performed.

The closed-loop multi-source regulation operates over a slower time scale. When the source's data sending rate is less than some fraction of the maximum theoretical throughput of the channel, the source regulates itself. When this value is exceeded, the source starts to require constant, slow time-scale feedbacks from the destination to maintain its rate. The reception of positive feedbacks at a source serves as self-clocking mechanism, which allows the source to retain its current sending rate; otherwise, negative feedbacks force it to reduce the rate.

CODA senses the channel load by performing periodic samplings, in-between which a sensor node goes into sleep mode, such that energy can be saved. However, the channel load measured in this way is not accurate. On one hand, in order to conserve energy, the sampling rate cannot be high. Therefore, to obtain accurate information, CODA has to take a long time to perform sampling. On the other hand, for bursty traffic, channel load measured over a long history will not reflect the media's current busyness.

6.5 Weighted Load Metric

In [55], the *weighted load metric*, W , is proposed to implement a rate limiting algorithm. W is defined as a weighted summation of the priority values, *precedence* and the *number of transmission attempts*, of all the packets in the queue. Each node also maintains an *activity metric* W_{nb} corresponding to each of its neighbors. W_{nb} indicates the value of W of the neighbor.

When a node is ready to transmit a packet and the next hop's ID is available, it transmits the RTS including extra fields (a) its W metric, and (b) a final destination flag to inform the receiver if it is the final destination of a packet. The next hop node checks its availability of the data channels. In addition, it updates the activity metric W_{nb} by using the W metric in the received RTS.

If the RTS receiving node's queue size has reached a certain high threshold, TH_1 , then a *Negative CTS (NCTS)* is sent to the RTS sender to cancel the transmission of the data packet. If a node's queue size is above the threshold TH_2 ($< TH_1$), and its W and the neighbor's W_{nb} are high, for the incoming *last-hop packets, emergent traffic and system messages*, the receiving node will send a CTS with a *slow down factor* greater than 1, which means a slow down of the transmission should be performed; for other packets, an NCTS is transmitted. If a node's queue size is above a threshold TH_3 ($< TH_2$), for all packets, it replies with a CTS containing a slow down factor greater than 1; otherwise, a CTS with a slow down factor equal to 1, meaning no slow down is required, is sent to start the pending Data transfer.

After receiving a CTS with a slow down factor greater than 1, the RTS sender will increase its transmission interval T_{nh} (i.e. lower the sending rate) according to its W , such that congestion at the next hop can be alleviated.

A problem with this scheme is that it slows down transmission of all the packets regardless of their precedences. Thus, the urgent and important messages are also

affected in the same way as the low priority packets. A simple improvement could be that the low priority packets are delayed or discarded first, such that the transmission of higher priority packets is affected as little as possible.

6.6 Congestion-controlled Adaptive Lightweight Multicast (CALM)

CALM [56] seeks to reduce the multicast delivery rate to what is acceptable to a set of receivers.

CALM transmits data packets at a rate determined by the application until congestion arises. When a certain number of consecutive packets are lost, receivers conclude that congestion has occurred. They indicate this through *Negative Acknowledgements (NACKs)* transferred back to the source. Upon receiving NACKs, the source adds IDs of the complaining receivers to a *Receiver List* and enters congestion control phase.

During this phase, the source selects one receiver from the Receiver List at a time; it reduces the transmission rate and multicasts the next data packet with an indication, which instructs the targeted receiver to reply with an ACK. If the ACK is received before a timeout (based on the measured end-to-end delay between the receiver and the source through the NACK), the source presumes that the receiver no longer experiences congestion. This receiver is removed from the Receiver List and the next node in the Receiver List is chosen for the next ACK reception. When all receivers in the Receiver List acknowledge the source, the source assumes the network is no longer congested. It exits the congestion control phase and reverts to the initial traffic rate.

The receivers' NACKs are not guaranteed to arrive at the source since these messages may well be discarded on the way due to congestion. Even if a NACK has reached the source, in order to know whether the lowered sending rate is acceptable

to the receiver or not, the source has to wait until an Ack comes. It is possible that several dialogues occur before the receiver is finally satisfied with the sending rate. This process is very time consuming. Moreover, during this round-robin process, other receivers are still suffering from packet loss and more NACKs may be issued, which worsens the congestion situation.

6.7 Media Access Delay Regulator (MADR) Algorithm

The MADR [57] algorithm uses the *Proportional-Integral (PI) controller*, which is an *Adaptive Queue Management (AQM)* scheme proposed for wired networks [58]. MADR maintains the *medium access delay* at the *access point (AP)*, which is the time period between a packet arrives from the wired network and the time it is transmitted into the wireless LAN, by dynamically adapting the MAC layer queue size. Since the medium access delay reflects both the node's queue size and the channel collision probability, network congestion can be avoided by controlling the medium access delay within a reasonable range.

For every *COUNT_LIMIT* number of packets arriving from the infrastructure, MADR computes the *target queue size* by applying

$$q_{target} = \frac{T_{mac}}{S_{avg}} \quad (6.1)$$

where T_{mac} is the target average media access delay to the wireless network, S_{avg} is the packet's average service time (measured at the MAC layer by recording the service time of each packet and averaging over a sliding window, in which the service time is defined as the time duration between a packet's first transmission and its acknowledgement from the destination). If the current queue length is above the threshold Q_LIMIT , an incoming packet is discarded, otherwise, an incoming packet is dropped with the following probability:

$$p_{drop} = a \times (q_{current} - q_{target}) - b \times (q_{old} - q_{target}) + p_{drop} \quad (6.2)$$

where a and b are PI parameters, $q_{current}$ and q_{old} are current and previous queue size. In this way, by adjusting the queue length and dropping overflowing packets, congestion can be avoided.

MADR uses the measured average service time to calculate the target queue length for a given target media access delay. However, in cases that the channel error rate is high, there may not be any retransmission for the broadcast packets while unicast packets may be retransmitted several times. Therefore, the average service time is not necessarily an accurate reflection of the actual congestion status. Consequently, q_{target} and p_{drop} cannot be calculated precisely.

6.8 Summary

Controlling or avoiding congestion is important for improving performance. In this chapter, we have reviewed various mechanisms including: (a) broadcast routing protocols that reduce excessive rebroadcast or forwarding to alleviate the broadcast storm problem; (b) a modified version of routing protocol, ODMRP; (c) schemes that regulate the source's transmission rate, together with link algorithms that measure the network's congestion (e.g. through the channel load or queue size), and that perform congestion signaling and/or queue management.

As discussed in [54], the queue length or buffer occupancy does not provide an accurate indication of congestion except in the extreme cases when the queue is empty or about to overflow. The channel load (media's busyness) is a reasonable metric for measuring the congestion, and is used in our congestion control schemes described in the next chapter.

We do not employ Explicit Congestion Notification (ECN) and the source control mechanisms in the MAC or routing layers, as we believe that it is more appropriate to let the upper layers decide the traffic rate according to their specific requirements, if the underlying layers cannot reduce congestion effectively.

Chapter 7. Coping With Congestion

In Chapter 5, the Adaptive Round-robin Acknowledge and Retransmit (ARAR) has been shown to improve the reliability of broadcast at the MAC layer of wireless multi-hop networks. This results in enhanced packet delivery ratio of multicast routing protocols that rely on the MAC layer broadcast service. However, under heavy traffic, the retransmissions can exacerbate the state of congestion in the network, which leads to degraded performance. In this chapter, we address this problem by using an integrated approach where the routing layer and MAC layer issues are considered together. In particular, we propose (a) a Queue Splitting (QS) mechanism, which splits the queue in-between the MAC and routing layers into two queues, the Control Queue and the Default Queue, and a priority scheduling is used at the MAC layer, such that congestion can be alleviated; (b) ADaptive ReTransmission (ADRT) that dynamically adjusts the retransmission limit of the ARAR based on the medium's busyness (channel load). The effect of these modifications on the ARAR scheme is evaluated by using Gateway-Based Multicast Protocol (GBMP) and On-Demand Multicast Routing Protocol (ODMRP) for the network layer, along with the modified ARAR at the MAC layer. Simulations show that compared to the case where only the basic ARAR scheme is used at the MAC layer, the routing layer's performance improves significantly with the proposed

cross-layer enhancements. We also conclude that for different kind of multicast routing protocols, QS alone or “QS and ADRT combination” (termed as QS+ADRT) should be adopted for ARAR's integration with the network layer.

7.1 Motivation

Because of the possibility of congestion by the retransmissions of ARAR and limited storage space, some packets queued for transmission may be discarded. For the routing protocol, this results in several problems. Firstly, discarding the control packets (such as JQ and JR in ODMRP, GJQ, GJR, LJQ and LJR in GBMP) prevents routing protocols from constructing the multicast topology properly. Inferior topology affects the data delivery as the links are more liable to break. Secondly, discarding data packets also degrades the performance of packet delivery. Thirdly, for the protocols that have link repair mechanisms (such as GBMP and ADMR, etc.), discarding packets will make the routing layer protocols generate more control packets. The reason is that these protocols use the loss of a number of consecutive data packets as a sign of link breakage. If due to queue overflow many data packets are lost, receiving nodes may think that link breakages have occurred and thus, link repair mechanisms are initiated. Furthermore, if the repair control packets are lost or arrive late, the routing layer considers that the repair has failed, and reinitiates either the local or global repair. As a result, more and more control packets are generated, making the congestion severe. As shown in Figure 7.1, using the network scenario and heavy traffic configuration presented in Section 7.4, when ODMRP and GBMP are functioning with ARAR, the average end-to-end delay is so high that it may prevent many applications from operating satisfactorily.

Consequently, we need to introduce new mechanisms that are able to smoothly integrate ARAR with the network layer while maintaining satisfactory performance (in terms of, e.g. end-to-end delay). We propose two schemes in this chapter for this

purpose. The first scheme, called Queue-Splitting (QS) mechanism, aims to alleviate the congestion resulting from ARAR under heavy traffic. The second scheme modifies the ARAR itself by introducing the ADaptive ReTransmission (ADRT) such that the retransmissions can be performed adaptively according to the channel load.

Figure 7.1. High delay introduced by ARAR under heavy traffic

7.2 Queue Splitting (QS)

In order to alleviate the congestion generated by excessive frame retransmission in the MAC layer, we need to scrutinize the general characters of the routing protocols. Normally, routing protocols use three categories of packets for their operations: multicast topology construction and maintenance control packets (let us call them to be *ConstPkts*), link repair control packets (*RepairPkts*) and data packets. In most cases, the *ConstPkts* are transmitted regularly to update the multicast topology to deal with nodes' mobility; the *RepairPkts* are initiated if a node finds that a link is broken and smooth data delivery is affected. Although the link breakages caused by nodes' mobility is unavoidable, those caused by inferior topology can be reduced, e.g. by ensuring that *ConstPkts* are delivered reliably even under congestion condition. Therefore, the *ConstPkts* are to be transmitted with a higher priority; however, the transmission of *RepairPkts* and data packets should not be starved.

Taking this into account, we split the queue, which is in-between the Routing and MAC layer (let us call it *inter-MAC-Routing queue*) into two queues (see Figure 7.2): the *Control Queue (CtrlQ)* and the *Default Queue (DftQ)*. The CtrlQ contains control packets of the type ConstPkt solely. The DftQ contains data packets and other control packets (such as RepairPkts). Packets in the CtrlQ have higher priority over those in the DftQ. However, to avoid starvation of the DftQ, we guarantee that in every *MAX_DFQ_PKT_ITVL* interval, the queue is served at least once. Theoretically, the exact value of *MAX_DFQ_PKT_ITVL* should be assigned according to the requirement of the upper layers. However, as we will see through simulations, in most cases, 50ms interval is good enough.

In this arrangement, ConstPkts such as GJQ, GJR, LJQ and LJR in GBMP, JQ and JR in ODMRP are sent to the CtrlQ; other packets are sent to the DftQ by the routing protocol. Since ConstPkts are the most important packets (for the purpose of topology construction and maintenance), they should be sent with a higher priority than the other packets, so that the occurrence of link breakages can be reduced. The other control packets that are used for link repair are given the same priority as the data packets. The reason is, in routing protocols such as GBMP, the loss of several consecutive data packets is deemed as the result of a link breakage. The resulting link repair packets are used for local repairs only, and therefore, if they are not delivered in time, the local link repair may fail and only a few receivers will suffer. Furthermore, link repair packets are large in number. For example, in the simulation using TP-3.2 in Chapter 3, the repair packets are on an average 15.8 % of the number of overall packets issued by GBMP, while the ConstPkts are only 0.5 %. If repair packets are given the same priority as the ConstPkts, the multicast topology construction and maintenance will suffer, affecting the smooth delivery of data packets as well.

Figure 7.2. The Queue-Splitting (QS) mechanism

One may argue that assigning a high priority to the ConstPkts may delay the service of packets in the DfQ. On close scrutiny, we find that it is not so. The reason is that, most of the CtrlQ packets (ConstPkts) are generated at a much lower frequency than the data and other control packets. For instance, in GBMP, GJQs are sent every 60 seconds by the sources and LJQs are sent every 9 seconds by the gateway nodes; in ODMRP, JQ is sent every 3 seconds. Therefore, queue splitting does not affect the service of the DfQ significantly. Even when the CtrlQ holds many packets in some rare cases, the DfQ is served at least once every MAX_DFQ_PKT_ITVL interval, ensuring that the RepairPkts and data packets are not delayed excessively.

The objective of our QS mechanism is very different from the service differentiation in IEEE 802.11e ([60], [61]). In IEEE 802.11e standards, MAC service data units (MSDUs) belonging to different Traffic Categories (TCs) are transmitted through different backoff instances: high priority TCs compete for the medium earlier than the low priority ones; the minimum value of Contention Window (CW) is assigned depending on various TCs. The service differentiation is accomplished in a probabilistic manner. However, the purpose of the QS is to reduce the congestion, which may be caused by ARAR, and to improve the data delivery at the upper layers. This is achieved by giving higher priority to the ConstPkts over the

other packets in a deterministic manner.

7.3 Adaptive Retransmit (ADRT)

In ARAR, a retransmission counter is associated with each stored frame in the BcDQ (Broadcast Data Queue, used to store frames for future retransmissions). If the retransmission counter's value reaches MAX_RETX_LIMIT (the maximum number of times a frame can be retransmitted), the stored frame is discarded. On one hand, excessive retransmissions under heavy traffic congest the network while not achieving much performance gain (sometimes, even causes degradation). On the other hand, if no retransmission is allowed, the reliability cannot be improved. Therefore, retransmissions should be adjusted according to the status of the network.

Our ADaptive ReTransmission (ADRT) scheme is as follows. ADRT continually monitors the medium. Depending upon the duration the medium is busy within the pre-defined time window (sliding time duration), the variable *CURRENT_RETX_LIMIT* is modified. Whenever the retransmission of a stored frame is required, ADRT retransmits it if the retransmission counter of that frame is less than the *CURRENT_RETX_LIMIT*.

A heuristic method is used to modify the retransmission limit as follows. Let us assume that the MAX_RETX_LIMIT is equal to M and suppose the ratio of the busy period in the time window is r , which is expressed as

$$r = \frac{\text{busy_period_in_time_window}}{\text{duration_of_time_window}}.$$

Figure 7.3 exemplifies how the retransmission limit is adjusted when M equals 3. Let us assume that at first, *CURRENT_RETX_LIMIT* equals 3. With the increase of the traffic load, the medium becomes busier and r increases. *CURRENT_RETX_LIMIT* remains 3 until r is above 0.366, at which moment *CURRENT_RETX_LIMIT* is decremented to 2. If the traffic load continues to

increase, CURRENT_RETX_LIMIT will be decremented to 1 if r is above 0.699. However, if the traffic load does not change dramatically, and r varies in the range from 0.300 to 0.633, CURRENT_RETX_LIMIT is not modified. It is incremented to 3 only when r reaches below 0.300. It should be noted that in practice, r is seldom equal to 1 as inter-frame spaces are always there. Therefore, for implementation, when r approaches a value close to 1, e.g. 0.995, CURRENT_RETX_LIMIT is decremented from 1 to 0.

Figure 7.3. Dynamic adjustment of the retransmission limit

In general, $\text{CURRENT_RETX_LIMIT} = M - i$, $0 \leq i < M$, when r ascends above $\frac{i}{M} + \left(\frac{1}{M} \times 10\%\right)$ $1 \leq i < M$ or approaches unity, CURRENT_RETX_LIMIT is decreased by one. Otherwise, when r is descending below $\frac{i}{M} - \left(\frac{1}{M} \times 10\%\right)$ $1 \leq i < M$, CURRENT_RETX_LIMIT is increased by one until it reaches M . Initially, CURRENT_RETX_LIMIT is set to M .

The difference between ADRT and the work reported in [57] is that, rather than adapting the inter-MAC-routing queue size, we adapt the overall traffic intensity by adjusting the retransmission limit. Therefore, by regulating a node's retransmission, a trade-off between reliability and frame redundancy is accomplished. As mentioned in Section 6.8, the sending rate of the "fresh packets" is left to the higher layers to

determine.

7.4 Performance Evaluation

In this section, we study the merit of introducing the QS and ADRT mechanisms.

We evaluate the performance using GloMoSim. Table 7.1 lists the parameters used in our simulation.

Table 7.2 associates various packet inter-arrival times with different packet lengths such that the traffic intensity remains constant for different packet inter-arrival times. There are three sources that start to send data packets at the same time. Group members maintain their membership throughout the simulation period. All the nodes are assumed to be mobile, with mobility parameters given in Table 7.1.

In section 7.4.1, a set of simulation runs are performed to decide the appropriate value for MAX_DFQ_PKT_ITVL used by the QS mechanism. The performance improvements by introducing QS and ADRT are discussed in detail in sections 7.4.2 and 7.4.3 respectively. Section 7.4.4 discusses the performance improvement brought about by the combination of QS and ADRT mechanisms.

The performance is measured using *Bandwidth Efficiency (BE)*, average end-to-end delay and Packet Delivery Ratio (PDR) metrics. BE is defined as

$$\frac{\sum_{i=1}^{N_R} \text{Number of bytes delivered to Receiver } i}{\sum_{j=1}^{N_{NW}} \text{Number of bytes sent (including retransmissions) by Node } j}$$

It reflects the bandwidth efficiency of delivering data packets from the sources to the receivers. A higher value of BE means that the scheme employed is more efficient in terms of bandwidth use for delivering information.

Table 7.1. Simulation Environment Parameters

Parameters	Value
Mobility model	Random-Way Point
Nodes' speed	20 m/s
Pause time	5 s
Radio transmission range	250 m
Sensing range	500 m
Propagation path-loss model	Free space
Channel capacity	2 Mbps
Simulation time	600 s
Number of sources	3
Broadcast DATA queue length	64 frames
Retransmit limit in ARAR	3
Maximum number of packets packed in a frame	32
MAX_FRAME_LENGTH	2312 octets
Number of nodes	50
Number of members	20
Network area	1000m × 1000m

Table 7.2. Packet Inter-Arrival Times and Corresponding Packet Lengths

Packet Inter-Arrival Time (msec)	Packet Length (Octets)
50	160
100	320
150	480
200	640
250	800

Each simulation run was conducted for 10 times with different seed numbers.

The reported data were averaged over these runs.

7.4.1 Value of MAX_DFQ_PKT_ITVL

In order to find a suitable value for the MAX_DFQ_PKT_ITVL, we run a set of simulations using 25, 50, 100, 200, and 400 milliseconds for MAX_DFQ_PKT_ITVL. As demonstrated in Figure 7.4 and Figure 7.5, although the performance in terms of the PDR and BE for ODMRP and GBMP does not vary considerably, difference in average end-to-end delay is noticeable. For both ODMRP

and GBMP, when MAX_DFQ_PKT_ITVL is 400 milliseconds, the delay is the highest. Obviously, longer the maximum interval to serve the DftQ, higher is the average queueing delay. It is also observed that the performance does not vary substantially when the MAX_DFQ_PKT_ITVL is 25, 50 or 100 milliseconds. Therefore, in the rest of the simulations in this chapter, we simply use 50 milliseconds for MAX_DFQ_PKT_ITVL.

Figure 7.4. Performance under Different MAX_DFQ_PKT_ITVL for ODMRP

Figure 7.5. Performance under Different MAX_DFQ_PKT_ITVL for GBMP

7.4.2 Improvement by QS

Table 7.3 shows the average end-to-end delay of ODMRP, when it is working with ARAR, ADRT, QS, ADRT-QS (i.e., applying ADRT and QS together on ARAR) and IEEE 802.11 MAC under different traffic loads; and Table 7.4 shows that under GBMP. Figure 7.6 (a) and (b) show the PDR and BE performance under ODMRP; Figure 7.6 (c) and (d) show the PDR and BE performance under GBMP.

The improvement achieved by introducing QS is noteworthy. Under ODMRP/QS, the end-to-end delay has been reduced approximately to one third of that in ODMRP/ARAR. A similar phenomenon is observed in GBMP/QS. The delay reduction in ODMRP is due to two reasons. The first one is that with the high priority set for the ConstPkt and reliability provided by the underlying ARAR, a robust multicast topology is constructed for the efficient data delivery. The second reason is that serving at least one packet every MAX_DFQ_PKT_ITVL interval

reduces the queueing delay in DftQ, which makes up a large portion of the end-to-end delay. For GBMP, the above two factors further reduce the link breakages, and therefore, less RepairPkts are generated for link repairs (see Table 7.5 for reduction in repair packets issued by GBMP with QS).

Table 7.3. Average end-to-end delay (sec) of ODMRP with various MAC protocols

	50msec	100msec	150msec	200msec	250msec
ARAR	7.760143	9.143806	10.09229	10.45686	10.02068
QS	3.369427	3.359558	3.378103	3.532862	3.375027
ADRT	0.948983	1.142623	0.839099	1.283131	1.397299
ADRT-QS	0.305984	0.39213	0.374348	0.582375	0.749202
802.11	0.220076	0.052996	0.047462	0.054007	0.060929

Table 7.4. Average end-to-end delay (sec) of GBMP with various MAC protocols

	50msec	100msec	150msec	200msec	250msec
ARAR	2.331601	4.41415	2.546938	1.5593	1.634586
QS	1.37432	1.747173	0.904699	0.822629	0.650483
ADRT	0.179279	0.266144	0.25139	0.418787	0.496601
ADRT-QS	0.184195	0.25441	0.255183	0.432751	0.517827
802.11	0.061711	0.037596	0.037917	0.049676	0.05516

Table 7.5. Reduction in Control Packet Issued by GBMP with QS against ARAR

	50msec	100msec	150msec	200msec	250msec
Reduction	5.02%	14.00%	13.07%	11.73%	10.38%

Another major improvement caused by QS is observed in the PDR metric. Although ARAR without the QS mechanism achieves significant improvement in PDR against IEEE 802.11, QS improves it further in heavy traffic load. It is easy to understand that, especially for GBMP, the decrease in RepairPkts leaves more bandwidth for data packets, and consequently the PDR improves.

7.4.3 Improvement by ADRT

ADRT aims to avoid congestion in the network. Thus, it is not surprising that the end-to-end delays recorded in both ODMRP/ADRT and GBMP/ADRT are significantly reduced due to lessening of congestion, and they are even lower than ODMRP/QS and GBMP/QS respectively. For example, when the packet inter-arrival time is 50 milliseconds and each packet is 160 bytes long, the average end-to-end delay in ODMRP/ADRT is 0.949 seconds and in GBMP/ADRT is 0.179 seconds; when the packet inter-arrival time is 250 milliseconds and each packet is 800 bytes long, it is 1.397 seconds in ODMRP/ADRT and 0.497 seconds in GBMP/ADRT.

However, the PDR performance of ODMRP and GBMP under ADRT is totally opposite. The PDR of ODMRP/ADRT degrades compared to that of ODMRP/QS scheme, while that of GBMP/ADRT further improves over GBMP/QS (which are shown in Figure 7.6 (a) and (c)). ODMRP’s PDR degradation is due to the fact that while trying to avoid congestion, ADRT reduces the retransmission of reported lost frames (sometimes when the network is highly congested, there may not be any retransmission at all; see Table 7.6 for reduction in data packet issued by ODMRP with ADRT). This adversely affects the reliability at the MAC layer, which affects not only the data packets, but also the ConstPkts (see Table 7.7 for reduction in control packets issued by ODMRP with ADRT). This makes the multicast topology less robust. The lack of a robust topology together with less retransmissions finally result in decreased PDR in ODMRP.

Table 7.6 Reduction in Data Packets Issued by ODMRP with ADRT against QS

	50msec	100msec	150msec	200msec	250msec
Reduction %	13.44%	15.54%	12.51%	14.13%	13.51%

Table 7.7. Reduction in Control Packets Issued by ODMRP with ADRT against QS

	50msec	100msec	150msec	200msec	250msec
Reduction %	20.08%	22.56%	21.17%	19.32%	19.45%

On the contrary, GBMP’s PDR increases as a consequence of introducing ADRT. With reference to the increased BE shown in Figure 7.6 (d), GBMP obviously benefits from the avoidance of congestion. GBMP employs link repair schemes that initiate RepairPkts. Hence, the retransmission of the RepairPkts by ARAR indeed takes a large portion of the limited bandwidth in GBMP. The bandwidth left for data packets is accordingly reduced. Nevertheless, the introduction of ADRT reduces unnecessary retransmissions, which also reduces bandwidth utilized by RepairPkts. As is shown in Table 7.8 and Table 7.9, the GBMP’s reduction rate in RepairPkts is much more than those in ConstPkts and data packets. Therefore, ConstPkts and data packets get better chance for transmission, resulting in the increased PDR.

Table 7.8. Reduction in ConstPkts and Data Packets Issued by GBMP with ADRT against ARAR

	50msec	100msec	150msec	200msec	250msec
Reduction %	11.94%	8.77%	6.67%	4.28%	5.49%

Table 7.9. Reduction in RepairPkts Issued by GBMP with ADRT against ARAR

	50msec	100msec	150msec	200msec	250msec
Reduction %	47.56%	55.64%	48.34%	42.50%	36.63%

7.4.4 Improvement by combination of QS and ADRT

In this subsection, we examine the performance by applying QS and ADRT together (which is represented by the data series labeled ODMRP/ADRT-QS and GBMP/ADRT-QS in Table 7.3, Table 7.4 and Figure 7.6). For ODMRP, ADRT-QS achieves even lower end-to-end delay than ADRT alone. However, the PDR under ODMRP/ADRT-QS degrades compared to that under ODMRP/QS and ODMRP/ADRT, and so does the BE performance. For GBMP, compared to ADRT, PDR and BE are slightly increased in GBMP/ADRT-QS, however, the average end-to-end delay is almost equal.

The reason for this deviation in the performance of ODMRP and GBMP is as follows. ODMRP relies solely on intermediate nodes' data forwarding, and the receivers are receiving data in a passive way. Therefore, the reduction in retransmission (due to ADRT) greatly affects the data delivery adversely, even though QS is used for constructing robust topology. On the other hand, GBMP employs link repair mechanisms that actively repair broken links and "ask for" data from upstream nodes. The functioning of ADRT and QS in GBMP carries out a nice trade-off between reliability and overall performance.

Figure 7.6. Performance comparison of different schemes

So far, it is clear that for different multicast routing protocols, the adoption of

different techniques into ARAR for its integration with the network layer is necessary. For protocols that do not possess link repair mechanisms and are based thoroughly on topology redundancy for multicast topology construction and data delivery, such as ODMRP, a good performance tradeoff can be achieved by using QS mechanism with ARAR. For protocols that employ link repair mechanisms, such as GBMP, the combination of QS and ADRT should be used with ARAR as they provide robust topology, reduce link breakages and avoid congestion.

7.5 Summary

The interaction between protocols in different layers is an important issue, especially for mobile ad hoc networks because of the physical constraints. One of the examples is that MAC layer's reliable broadcast service is important for many multicast routing protocols in the MANETs. In Adaptive Round-robin Acknowledge and Retransmit (ARAR), retransmission is used for improving the reliability. However, excessive retransmissions may generate congestion in the network, which may prevent applications from operating efficiently. In this chapter, we have analyzed this issue using the Adaptive Round-robin Acknowledge and Retransmit (ARAR) MAC layer protocol with two multicast routing protocols: Gateway-Based Multicast Protocol (GBMP) and On-Demand Multicast Routing Protocol (ODMRP). Based on that, we have proposed two schemes: 1) the Queue Splitting (QS) mechanism, which constructs and maintains robust multicast topology and consequently alleviates congestion; 2) the ADaptive ReTransmit (ADRT) mechanism, which dynamically adjusts the retransmission limit based on the channel load to avoid congestion. Both of these two schemes are proposed to cope with the problem of congestion under heavy traffic load and consequently further improve the network layer's performance. Simulations show that the performance metrics, such as average end-to-end delay, packet delivery ratio and transmission efficiency, improve considerably compared to

using the basic ARAR mechanism for providing broadcast service to the routing protocols. From the simulation results, we also drew the conclusion that for different multicast routing protocols, the adoption of different techniques (QS or ADRT) into ARAR for its integration with the network layer is necessary under heavy traffic.

Consequently, with the GBMP as the multicast routing protocol, the ARAR as the MAC layer enhancement to the IEEE 802.11 protocol for providing reliable broadcast service to the network layer, and QS and ADRT utilized for the smooth integration between the MAC and network layers, we are able to improve the performance of multicasting significantly in MANETs.

Chapter 8. Conclusions and Future Work

In this chapter, we summarize our main contributions and present the future work.

8.1 Main Research Contributions

Performing multicasting in Mobile Ad hoc Networks is a very challenging task because of its dynamic network topology, multi-destination communications, narrow bandwidth, moderate quantity of memory storage, and finite battery power, etc. Also the unique properties of the wireless medium make the design of MAC protocols very different from that of the wire-line networks. Moreover, the highly dependent operations between the MAC and network (routing) layers makes the design even more complex especially when the network is operating under heavy traffic load.

In this thesis, we have mainly focused on improving the performance of multicasting in MANETs by designing protocols at the network and MAC layers. We first proposed a new multicast routing protocol, called the *Gateway-Based Multicast Protocol (GBMP)*. Realizing that the MAC layer's unreliable broadcast is detrimental to the performance of the routing layer, we then developed *Round-Robin Acknowledge and Retransmit (RRAR)*, which modifies IEEE 802.11 MAC to provide

reliable broadcast service to the routing layer. Then, we proposed *Adaptive Round-robin Acknowledge and Retransmit (ARAR)*, which reduces overhead and better adapts to traffic load. Due to the unique features of MANETs, the traditional way of designing different protocol layers independently is not applicable. Therefore, we have then proposed two schemes, the *Queue Splitting (QS)* and *ADaptive ReTransmit (ADRT)*, which aim to integrate ARAR with the GBMP smoothly. Our main contributions in this thesis are as follows:

Gateway-Based Multicast Protocol (GBMP): We have proposed a new multicast protocol for MANET, called the Gateway-Based Multicast Protocol (GBMP). Compared to the existing protocols, GBMP (a) improves the speed and reduces the cost of the multicast tree repair mechanism by using hierarchical loosely structured tree topology; (b) increases the transmission efficiency by using *Passive Receive Mode (PRM)*; and (c) reduces the amount of control overhead by using the *Bi-directional Local Link Repair (BLLR)* strategy, which implements *Upstream-Node-Initiated Local Repair (UNILR)* and *Receiver-Initiated Attach Procedure (RIAP)*, to repair the broken links locally.

Round-Robin Acknowledge and Retransmit (RRAR): The MAC layer's reliable broadcast service is important to many multicast routing protocols in MANETs. Therefore, we have proposed Round-Robin Acknowledge and Retransmit (RRAR) to improve the reliability of broadcasting. RRAR provides features such as (a) the novel round-robin acknowledgement scheme for reporting lost frames yet avoiding the Ack storm problem; and (b) the MAC layer retransmission for lost frames. By comparing RRAR with the basic IEEE 802.11 CSMA/CA broadcasting under the On-Demand Multicast Routing Protocol (ODMRP), the performance of ODMRP has been shown to improve significantly.

Adaptive Round-robin Acknowledge and Retransmit (ARAR): We continue to improve the RRAR by proposing the Adaptive Round-robin Acknowledge and Retransmit (ARAR) scheme. ARAR adjusts the frame length according to the traffic

load; when the traffic load is heavy, multiple packets can be packed into one frame, which reduces control overhead. For short frames, a simple and effective Data/BrAck (Broadcast Acknowledgment) handshake is used; for long frames, the RTS/CTS/Data/BrAck four-way handshake is utilized to avoid hidden terminal problem and to reduce costly retransmissions. These features are easy to implement and they improve the performance considerably.

Queue-Splitting (QS): We observe that simple integration of MAC and network layer protocols in the MANETs may result in inferior performance especially under heavy traffic. In order to have a smooth and efficient integration, we propose a Queue Splitting (QS) mechanism, which splits the queue in-between the MAC and routing layer into two queues, the *Control Queue* and the *Default Queue*. A priority scheduling is used at the MAC layer to serve these two queues, which alleviates the congestion and improves the performance substantially.

ADaptive ReTransmit (ADRT): The limited bandwidth is a critical constraint in MANETs. The ARAR scheme improves the reliability of the MAC layer's broadcast service using retransmissions. However, excessive retransmissions may cause congestion in the network. Therefore, we proposed a modification in the ARAR scheme, called ADaptive ReTransmission (ADRT), which dynamically adjusts the retransmission limit based on the medium's busyness (channel load), to avoid congestion. ARAR using ADRT is thus able to achieve a good trade-off between reliability and other performance metrics.

Therefore, by implementing the GBMP for multicast routing, ARAR at the MAC layer for reliable broadcast, and QS and ADRT for smooth integration between ARAR and GBMP, we are able to provide high performance multicasting in MANETs.

8.2 Future Work

8.2.1 Energy-efficiency and Power-awareness in Multicasting

In MANETs, nodes are usually supported by limited battery power. Designing energy-efficient and power-aware protocols is meaningful not only to prolonging the nodes' lifetime, but also to reducing the interference. Battery power can be saved by reducing transmission power, by letting a node enter sleep mode, and by reducing redundant transmissions, etc. Nonetheless, reducing transmission power shortens the transmission range, which may result in longer routes or link breakages. When the sleep mode is used, the end-to-end delay may degrade; also, in some cases, the power consumption for switching between the sleep mode and active mode may exceed the power saved by the sleep mode operation. Moreover, the retransmissions for lost frames consume extra energy; yet, reducing retransmissions to save power may result in lower reliability. Taking these issues into account, a trade-off must be achieved between energy saving and the requirements of the applications.

8.2.2 Quality of Service (QoS) Support for Multicast Routing Protocol

Due to the increasing popularity of multimedia applications and potential commercial use of MANETs, QoS support for multicasting in MANETs is highly desirable. In order to support QoS, the link state information such as delay, bandwidth, cost, loss rate and error rate etc. should be considered. However, measuring these parameters and sharing them among the nodes in the MANETs is not simple. First, the overheads for a node to update the link state information in a timely manner is very high due to the limited bandwidth. Second, in a dynamic environment like the MANETs, to share link state information in a timely manner is very difficult. As a consequence, to implement complex QoS functionalities for multicast routing is very challenging.

Besides, with the advent of IEEE 802.11e MAC protocol, the issue of “how does the network layer interact and cooperate with the MAC layer for the QoS support” becomes very interesting and challenging.

8.2.3 Impact of High Bit Error Rate

There may be wireless links in the MANETs with high bit error rates that the lower layers cannot provide certain level of service to the upper layers. An interesting task will be to study and evaluate the impact of high bit error rates on the performance of the protocols studied in this thesis. Furthermore, research in the fault-tolerant multicast routing protocols can also be of practical value.

References

- [1] IETF, Mobile Ad Hoc Network Working Group, <http://www.ietf.org/html.charters/manet-charter.html>.
 - [2] S. E. Deering and D. R. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs", *ACM Trans. on Computer Systems*, vol. 8, no. 2, pp. 85--110, 1990.
 - [3] U. Varshney, "Multicast Over Wireless Networks", *Communications of the ACM*, December 2002, Vol. 45, No. 12, pp 31-37.
 - [4] A. Chandra, V. Gummalla, and J. Limb, "Wireless Medium Access Control Protocols," *IEEE Communications Surveys*, Second Quarter 2000.
 - [5] D. Falconer, F. Adachi, and B. Gudmundson, "Time Division Multiple Access Methods for Wireless Personal Communications," *IEEE Communications Surveys*, pp. 50-57, 1995.
 - [6] D. Bertsekas and R. Gallager, "Data Networks," Prentice-Hall International Inc., 1992.
 - [7] R. Kohno, R. Meidan, and L. B. Milstein, "Spread Spectrum Access Methods for Wireless Communications," *IEEE Communications Magazine*, pp. 58-67, Jan. 1995.
 - [8] L. Kleinrock and F. A. Tobagi, "Packet Switching in Radio Channels: Part I –
-

- Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics,” IEEE Transactions on Communications, vol. COM-23, no. 12, December 1975, pp. 1400-1416.
- [9] P. Karn, “MACA – A New Channel Access Method for Packet Radio,” in Proceedings of the 9th ARRL/CRRL Amateur Radio Computer Networking Conference, September 1992.
- [10] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, “MACAW: A Media Access Protocol for Wireless LAN’s,” in Proceedings of ACM SIGCOMM, 1994, pp. 212-225, London, UK, Aug. 31 – Sept. 2, 1994.
- [11] C. Fuller and J.J. Garcia-Luna-Aceves, “Floor Acquisition Multiple Access (FAMA) for packet radio networks,” Computer Communication Review, vol. 25, (no.4).
- [12] Editors of IEEE 802.11, “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification,” Draft Standard IEEE 802.11, 1999.
- [13] Y. Yi, S.-J. Lee, W. Su and M. Gerla, “On-Demand Multicast Routing Protocol (ODMRP) for Ad Hoc Networks”, Internet-Draft, “draft-ietf-manet-odmrp-04.txt”, Nov. 2002.
- [14] J. G. Jetcheva and D. B. Johnson, “Adaptive Demand-Driven Multicast Routing in Multi-Hop Wireless Ad Hoc Networks”, in Proceedings of 2001 ACM International Symposium on Mobile ad hoc networking & computing 2001, Long Beach, CA, USA.
- [15] E. M. Royer and C. E. Perkins, “Multicast Ad hoc On-Demand Distance Vector (MAODV) Routing”, Internet-Draft, “draft-ietf-manet-maodv-00.txt”, July 2000.
- [16] C.-C. Chiang, M. Gerla and L. Zhang, “Forwarding Group Multicast Protocol (FGMP) for Multihop, Mobile Wireless Networks”, ACM Baltzer Journal of Cluster Computing: Special Issue on Mobile Computing, 1(2):187-196, 1998.
- [17] M. Lee, Y. K. Kim, "PatchODMRP: an ad-hoc multicast routing protocol",
-

- Information Networking, 2001. Proceedings. 15th International Conference on, 2001, Page(s): 537 - 543.
- [18] S. K. Das, B. S. Manoj and C. S. R. Murthy, "A Dynamic Core Based Multicast Routing Protocol for Ad hoc Wireless Networks", in Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC), June 9-11, 2002, EPFL Lausanne, Switzerland.
 - [19] S. Lee, C. Kim, "NSMP: Neighbor Supporting Ad Hoc Multicast Routing Protocol", in Proceedings of First Annual Workshop on Mobile Ad Hoc Networking and Computing, (MOBIHOC), Boston, pages 37-50, August 2000
 - [20] P. Sinha R. Sivakumar and V. Bharghavan, "MCEDAR: Multicast Core-Extraction Distributed Ad hoc routing", in Proceedings of the Wireless Communications and Networking Conference, 1999.
 - [21] R. Sivakumar, P. Sinha and V. Bharghavan, "CEDAR: a Core-Extraction Distributed Ad hoc Routing algorithm", IEEE Journal on Selected Areas in Communication, Vol. 17, No. 8, August 1999.
 - [22] V. Devarapalli and D. Sidhu, "MZR: A Multicast Protocol for Mobile Ad Hoc Networks", in Proceedings of IEEE International Conference on Communications (ICC), Helsinki, Finland, June 2001.
 - [23] Z. Haas, M. Pearlman and P. Samar, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks", Internet Draft, draft-ietf-manet-zone-zrp-04.txt, August. 2002.
 - [24] C. Jaikaeo and C-C. Shen, "Adaptive Backbone-Based Multicast for Ad hoc Networks", in Proceedings of IEEE International Conference on Communications, 2002.
 - [25] B. An and S. Papavassiliou, "A Mobility-Based Hybrid Multicast Routing in Mobile Ad-hoc Wireless Networks", Proc. IEEE Military Communications Conference (MILCOM2001), pp. 316-320, Oct. 2001.
 - [26] B. An and S. Papavassiliou, "A mobility-based clustering approach to support
-

- mobility management and multicast routing in mobile ad-hoc wireless networks”, *International Journal of Network Management* 2001; pp 387 – 395.
- [27] <http://www.aero.org/publications/GPSPRIMER/>, July 1999.
- [28] J. Xie, R. R. Talpade, A. Mcauley and M. Liu, “AMRoute: Adhoc Multicast Routing Protocol,” *Mobile Networks and Applications*, Volume 7, Issue 6 (Dec. 2002), pp. 429 - 439.
- [29] C. W. Wu, Y.C. Tay, and C.-K. Toh, “Ad Hoc Multicast Routing Protocol Utilizing Increasing id-numberS (AMRIS) Functional Specification,” *Internet draft*, Nov. 1998.
- [30] L. Ji and M. S. Corson, “A Lightweight Adaptive Multicast Algorithm,” in *Proceedings of IEEE Global Telecommunications Conference* 1998, pp. 1036–1042.
- [31] J. J. Garcia-Luna-Aceves and E.L. Madruga, “The Core-Assisted Mesh Protocol,” *IEEE Journal on Selected Areas in Communications*, August. 1999, pp. 1380–1394.
- [32] C. Gui, and P. Mohapatra, “Efficient Overlay Multicast for Mobile Ad Hoc Networks,” in *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, LA, March 2003.
- [33] M. Mauve, H. Füßler, J. Widmer and T. Lang, “Position-Based Multicast Routing for Mobile Ad-Hoc Networks”, *Accepted as Poster at ACM MobiHoc '03*, Annapolis, MD, USA, June 2003.
- [34] “An Overview of Source-Specific Multicast (SSM) Deployment”, *Internet draft draft-ietf-ssm-overview-02.txt*, Dec. 2001.
- [35] Global Mobile Information Systems Simulation Library, UCLA, <http://pcl.cs.ucla.edu/projects/glomosim>.
- [36] J. Tourrilhes, “Robust Broadcast: Improving the reliability of broadcast transmissions on CSMA/CA,” in *Proceedings of The Ninth IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*

- 98, pp. 1111-1115.
- [37] K. Tang and M. Gerla, "Random Access MAC for Efficient Broadcast Support in Ad Hoc Networks," in Proceedings of IEEE Wireless Communications and Networking Conference (WCNC), pp. 454-459, 2000.
 - [38] S. Banerjee and A. Misra, "Energy Efficient Reliable Communication for Multi-hop Wireless Networks," Accepted for publication to Journal of Wireless Networks (WINET).
 - [39] K. Tang and M. Gerla, "MAC Layer Broadcast Support in 802.11 Wireless Networks," in Proceedings of IEEE Military Communications Conference (MILCOM), pp. 544-548, 2000.
 - [40] K. Tang and M. Gerla, "MAC Reliable Broadcast in Ad Hoc Networks," in Proceedings of IEEE Military Communications Conference (MILCOM), pp. 1008-1013, October 2001.
 - [41] M.-T. Sun, L. Huang A. Arora and T.-H. Lai, "Reliable MAC Layer Multicast in IEEE 802.11 Wireless Networks," in Proceedings of The 2002 International Conference On Parallel Processing (ICPP), pp. 527-532.
 - [42] M. Zorzi and R. R. Rao, "Capture and Retransmission Control in Mobile Radio", IEEE Journal on Selected Areas in Communications, Vol. 12, No. 8, Oct. 1994.
 - [43] S. -T. Sheu, Y. Tsai and J. Chen, "A Highly Reliable Broadcast Scheme for IEEE 802.11 Multi-hop Ad Hoc Networks," in Proceedings of IEEE International Conference on Communications (ICC) 2002.
 - [44] K.-H. Lee and D.-H. Cho, "A Multiple Access Collision Avoidance Protocol for Multicast Service in Mobile Ad Hoc Networks," in Proceedings of the 57th IEEE Semiannual Vehicular Technology Conference VTC 2003-Spring.
 - [45] M. Lott, A. Sitalov, E. Linsky and H. Li, "Performance Analysis of Multicast Transmission in WLAN," in Proceedings of the 57th IEEE Semiannual Vehicular Technology Conference, 2003 (VTC 2003-Spring).
-

- [46] Z. Li, S. Nandi and A. K. Gupta, "Improving MAC Performance in Wireless Ad Hoc Networks Using Enhanced Carrier Sensing (ECS)," in Proceedings of the Third IFIP-TC6 Networking Conference (Networking 2004), Athens, Greece, May 9-14, 2004.
 - [47] Y.-C. Tseng, S.-Y. Ni and E.-Y. Shih, "Adaptive Approaches to Relieving Broadcast Storms in a Wireless Multihop Mobile Ad Hoc Network," IEEE Transaction on Computer, Vol. 52, No. 5, May 2003.
 - [48] C. Hu, Y. Hong and J. Hou, "On Mitigating the Broadcast Storm Problem with Directional Antennas," in Proceedings of IEEE International Conference on Communications 2003 (ICC '03), pp. 104-110.
 - [49] C.-C. Yang and C.-Y. Chen, "A Reachability-Guaranteed Approach for Reducing Broadcast Storms in Mobile Ad Hoc Networks," in Proceedings of IEEE 56th Vehicular Technology Conference, 2002 (VTC 2002-Fall), pp. 1036-1040.
 - [50] D. Kim, C.-K. Toh, J.-C. Cano and P. Manzoni, "A Bounding Algorithm for the Broadcast Storm Problem in Mobile Ad Hoc Networks," in Proceedings of IEEE Wireless Communications and Networking, 2003. (WCNC '03), pp. 1131-1136.
 - [51] S. -Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The Broadcast Storm Problem in a Mobile Ad hoc Network", International Conference on Mobile Computing and Networking (MobiCom'99), pp. 151-162.
 - [52] K. Tang and M. Gerla, "Reliable On-Demand Multicast Routing with Congestion Control in Wireless Ad Hoc Networks," in Proceedings of SPIE 2001, Denver, Colorado, August 2001.
 - [53] D. C. O'Neill, "Adaptive Congestion Control for Wireless Networks Using TCP," in Proceedings of IEEE International Conference on Communications, 2003 (ICC '03), pp. 82-86.
 - [54] C.-Y. Wan, S. B. Eisenman and A. T. Campbell, "CODA: Congestion
-

- Detection and Avoidance in Sensor Networks,” First ACM Conference on Embedded Networked Sensor Systems (SenSys ’03), Nov. 2003.
- [55] S. Z. Ozer, S. Zeng and C. Barker, “Congestion Control and Service Differentiation in Multihop Wireless Networks,” in Proceedings of IEEE Communications Society GlobeCom 2004 Workshops, pp. 62-67.
- [56] K. Tang, K. Obraczka, S.-J. Lee and M. Gerla, “Congestion Controlled Adaptive Lightweight Multicast in Wireless Mobile Ad Hoc Networks,” in Proceedings of the Seventh International Symposium on Computers and Communications, 2002 (ISCC ’02).
- [57] H. Xu, Q. Xue and A. Ganz, “Adaptive Congestion Control in Infrastructure Wireless LANs with Bounded Medium Access Delay,” International Mobility and Wireless Access Workshop (MobiWac’02).
- [58] C. V. Hollot, V. Misra, D. Towsley and W. B. Gong, “Analysis and Design of Controllers for AQM Routers Supporting TCP Flows,” IEEE Transactions on Automatic Control, Vol. 47, No. 6, 2002.
- [59] C. E. Perkins, E. M. Belding-Royer and S. R. Das, “Ad hoc On-Demand Distance Vector (AODV) Routing”, Internet-Draft, “draft-ietf-manet-aodv-13.txt”, Feb., 2003.
- [60] S. Mangold, S. Choi, P. May, O. Klein, G. Hiertz and L. Stibor, “IEEE 802.11e Wireless LAN for Quality of Service,” European Wireless 2002, Florence.
- [61] Editors of IEEE 802.11, “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Enhancements for Quality of Service (QoS),” Draft Supplement to IEEE 802.11 Standards, 1999.
- [62] T. Kunz and E. Cheng, “Multicasting in Ad-Hoc Networks: Comparing MAODV and ODMRP”, the Seventh European Conference on Computer Supported Cooperative Work (ECSCW 2001).
-

Annex: Location of the Source Code of the Simulation Models

The source code for the research work implementations in this thesis is available at <http://www.ntu.edu.sg/home5/pg04570879/GlomoSim/glomosim.zip>. The source code for GBMP is in the files `glomosim\network\gbmp.pc`, `glomosim\network\gbmp.h`, `glomosim\network\gbmp_ext_01.pc` and `glomosim\network\gbmp_ex_01.h`. The source code for ODMRP is in the files `glomosim\network\odmrp.pc` and `glomosim\network\omdrp.h` (ODMRP implementation is from the original GloMoSim package). The source code for ADMR is in the files `glomosim\network\admr.pc` and `glomosim\network\admr.h`.

The implementations of MACAM, RRAR, ARAR, QS, ADRT are included in the file `glomosim\mac\80211.pc` with corresponding macros defined for them.